

The \TeX 3 Package *

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

2021-12-15

Abstract

TODO

*Version 3.0 (last revised 2021-12-15)

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
IV	Implementation	44
15	ST_EX-Basics Implementation	45
15.1	The ST _E XDocument Class	45
15.2	Preliminaries	45
15.3	Messages and logging	46
15.4	Persistence	47
15.5	HTML Annotations	48
15.6	Languages	51
15.7	Activating/Deactivating Macros	51
16	ST_EX-MathHub Implementation	53
16.1	Generic Path Handling	53
16.2	PWD and kpsewhich	55
16.3	File Hooks and Tracking	56
16.4	MathHub Repositories	57

17	sTeX-References Implementation	63
17.1	Document URIs and URLs	63
17.2	Setting Reference Targets	65
17.3	Using References	66
18	sTeX-Modules Implementation	67
18.1	The module environment	70
18.2	Invoking modules	75
19	sTeX-Module Inheritance Implementation	77
19.1	SMS Mode	77
19.2	Inheritance	81
20	sTeX-Symbols Implementation	86
20.1	Symbol Declarations	86
20.2	Notations	92
21	sTeX-Terms Implementation	100
21.1	Symbol Invocations	100
21.2	Terms	102
21.3	Notation Components	109
22	sTeX-Structural Features Implementation	111
22.1	The feature environment	111
22.2	Features	113
23	sTeX-Statements Implementation	118
23.1	Definitions	119
23.2	Assertions	120
23.3	Examples	122
24	sTeX-Others Implementation	123
25	sTeX-Metatheory Implementation	124
26	Tikzinput Implementation	127
27	document-structure.sty Implementation	129
27.1	document-structure-common.sty Implementation	129
27.2	The OMDoc Class	130
27.3	Class Options	130
27.4	Beefing up the document environment	130
27.5	Implementation: OMDoc Package	131
27.6	Package Options	131
27.7	Document Structure	132
27.8	Front and Backmatter	136
27.9	Ignoring Inputs	137

28 MiKoSlides – Implementation	139
28.1 Class and Package Options	139
28.2 Notes and Slides	141
28.3 Header and Footer Lines	143
28.4 Colors and Highlighting	145
28.5 Sectioning	146
28.6 Excursions	148
28.7 Miscellaneous	149

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 _x \int _y \int _z f dx dy dz`?

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

Precedences

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

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

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

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

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

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

For example:

Example 10

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

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

1.1.2 Archives and Imports

Namespaces

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

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

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

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

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

Paths in Import-Statements

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

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

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

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

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

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

Part II

Documentation

Chapter 2

sTeX-Basics

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

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

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

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

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

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

image ($\langle\text{boolean}\rangle$) passed on to tikzinput.

2.1 Macros and Environments

<code>\sTeX</code>	Both print this sTeX logo.
<code>\stex</code>	

<code>\stex_debug:nn</code>	<code>\stex_debug:nn {$\langle\log\text{-}prefix\rangle$} {$\langle\text{message}\rangle$}</code>
-----------------------------	---

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

<code>\stex_add_to_sms:n</code>	Adds the provided code to the <code>.sms</code> -file of the document.
---------------------------------	--

<code>\if@latexml</code>	L ^A T _E X2e and L ^A T _E X3 conditionals for L ^A T _E XML.
<code>\latexml_if_p:</code>	
<code>\latexml_if:T</code>	
<code>\latexml_if:F</code>	
<code>\latexml_if:TF</code>	

We have four macros for annotating generated HTML (via L^AT_EXML or S^CA^LL^AT_EX) with attributes:

<code>\stex_annotate:nnn</code>	<code>\stex_annotate:nnn {⟨property⟩} {⟨resource⟩} {⟨content⟩}</code>
<code>\stex_annotate_invisible:nnn</code>	
<code>\stex_annotate_invisible:n</code>	

Annotates the HTML generated by $\langle content \rangle$ with

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

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

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

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

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

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

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

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

Chapter 3

STEX-MathHub

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

3.1 Macros and Environments

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

3.1.1 Files, Paths, URIs

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

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

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

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

<code>\stex_path_if_absolute_p:N</code>	\star
<code>\stex_path_if_absolute:N\underline{T}</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> <hr/> <code>\inputref:nn</code>	<code>\inputref[<archive-ID>]{<filename>}</code> <code>\inputs</code> the file <code><filename></code> in repository <code><archive-ID></code> .
<hr/> <hr/> <code>\libinput</code>	<code>\libinput{<filename>}</code> Inputs <code><filename>.tex</code> from the <code>lib</code> folders in the current archive and the <code>meta-inf</code> -archive of the current archive group (if existent). Throws an error if no file by that name exists in either folder, includes both if both exist.

Test 2

```

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

```

```

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

```

Chapter 4

sTeX-References

Code related to links and cross-references

4.1 Macros and Environments

Chapter 5

sTeX-Modules

Code related to Modules

5.1 Macros and Environments

`\l_stex_current_module_prop`

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

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

Additionally, it stores:

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

`\l_stex_all_modules_seq`

Stores full URIs for all modules currently in scope.

```
\g_stex_module_files_prop
\g_stex_modules_in_file_seq
```

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

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

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

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

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

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

```
\stex_add_constant_to_current_module:n
```

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

```
\stex_add_import_to_current_module:n
```

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

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

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

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

```
\stex_modules_current_namespace:
```

Computes the current namespace

Test 3

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

```

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

```

.

5.1.1 The module-environment

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

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

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

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

Test 4

```

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

```

```

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

```

.

Test 5

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

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

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

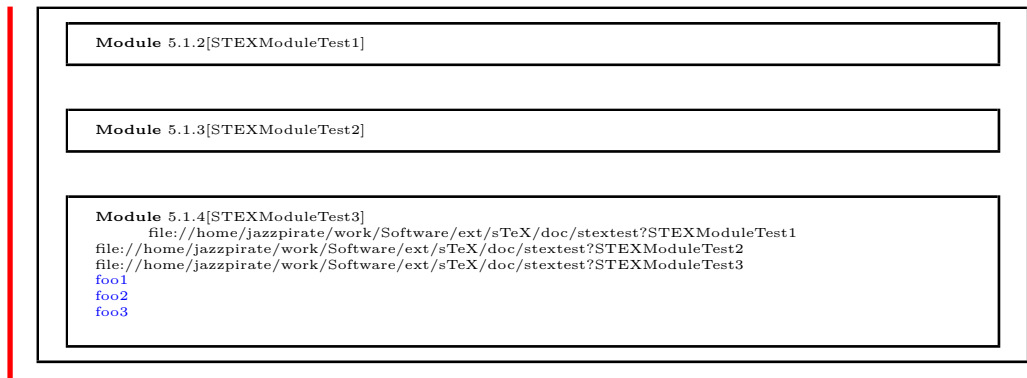
Attempts to find a module whose URI ends with `⟨fragment⟩` in the current scope and passes the full URI on to `\stex_invoke_module:n`.

`\stex_invoke_module:n`

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

Test 6

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



`\stex_activate_module:n`

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

Chapter 6

STEX-Module Inheritance

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

6.1 Macros and Environments

6.1.1 SMS Mode

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

`\g_stex_smsmode_allowedmacros_tl`

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

`\g_stex_smsmode_allowedmacros_escape_tl`

Macros that are executed with the category codes restored.

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

`\g_stex_smsmode_allowedenvs_seq`

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

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

Tests whether SMS mode is currently active.

`\stex_smsmode_set_codes:`

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

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

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

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

Test 7

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

6.1.2 Imports and Inheritance

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

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

Test 8

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

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

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

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

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

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

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

Test 9

```

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

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

```

Module 6.1.4[UseTest1]

Module 6.1.5[UseTest2]

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

Module 6.1.6[UseTest3]

Meaning: `\undefined<`

Meaning: `\macro:~>\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest2?bar}<`

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

Test 10

```

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

```

Circular dependencies:

Module 6.1.7[CircDep1]

`\macro:~>\stex_invoke_symbol:n {http://mathhub.info/tests/Foo/Bar/circular1?Circular1?fooA}<`

`\macro:~>\stex_invoke_symbol:n {http://mathhub.info/tests/Bar/Foo/circular2?Circular2?fooB}<`

`\stex_import_module_uri:nn`

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

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

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

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

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

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

2. Otherwise:

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

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

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

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

`\stex_import_require_module:nnnn`

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

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

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

Chapter 7

STEX-Symbols

Code related to symbol declarations and notations

7.1 Macros and Environments

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

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

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

`\stex_symdecl_do:n`

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

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

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

Test 11

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

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

`\l_stex_all_symbols_seq`

Stores full URIs for all modules currently in scope.

`\stex_get_symbol:n`

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

`\notation`

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

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

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

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

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

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

Test 12

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

Module 7.1.2[NotationTest]

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

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

Test 13

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

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

Chapter 8

STEX-Terms

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

8.1 Macros and Environments

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

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

Test 14

```

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

```

Module 8.1.1[MathTest1]
 $\langle x20x20a^b{}_c \rangle$ and $\langle x20x20a^b{}_c \rangle$.

Test 15

```

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

`\stex_term_custom:nn`

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

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

Test 16

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

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

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

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

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

\bar![bar]

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

\end{module}
```

```
Module 8.1.3[TextTest]
  some a and some b and also some c here.
  some a and some b and also some c here.

  or just some c
  bar
  or first b, then c, and finally a
```

`\stex_highlight_term:nn`

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

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

`\comp`

`\comp{<args>}`

`\compemph`

`\compemph@uri`

`\defemph`

`\defemph@uri`

`\symrefemph`

`\symrefemph@uri`

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

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

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

`\STEXinvisible`

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

`\ellipses`

TODO

Chapter 9

STEX-Structural Features

Code related to structural features

9.1 Macros and Environments

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}
 $\text{\isa{\op ab}\universe}$ 
\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:  $\text{\mM{op}ab}$ 

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

```
Module 9.1.1[StructureTest1]
aob:M
file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?StructureTest1/Magma-feature?universe,file://home/jazzpirate/work/S
feature?op
>macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?StructureTest1?Magma}<
Test: a+b
Test2: (U,+)
```

Chapter 10

TeX-Statements

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

10.1 Macros and Environments

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

Chapter 11

sTeX-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 [sTeXGitHub](#) repository [\[sTeX\]](#).

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

Part IV

Implementation

Chapter 15

ST_EX -Basics Implementation

15.1 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>
```

15.2 Preliminaries

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

```

26 mathhub .tl_set_x:N = \mathhub ,
27 sms .bool_set:N = \c_stex_persist_mode_bool ,
28 image .bool_set:N = \c_tikzinput_image_bool
29 }
30 \ProcessKeysOptions { stex }

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

```

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

Patching `expl3`, if outdated:

```

38 <@@=keys>
39 \cs_if_exist:cF { \c__keys_props_root_str .str_set:N }{
40 \cs_new_protected:cpn { \c__keys_props_root_str .str_set:N } #1
41 { \__keys_variable_set:NnnN #1 { str } { } n }
42 \cs_new_protected:cpn { \c__keys_props_root_str .str_set:c } #1
43 { \__keys_variable_set:cnnN {#1} { str } { } n }
44 \cs_new_protected:cpn { \c__keys_props_root_str .str_set_x:N } #1
45 { \__keys_variable_set:NnnN #1 { str } { } x }
46 \cs_new_protected:cpn { \c__keys_props_root_str .str_set_x:c } #1
47 { \__keys_variable_set:cnnN {#1} { str } { } x }
48 \cs_new_protected:cpn { \c__keys_props_root_str .str_gset:N } #1
49 { \__keys_variable_set:NnnN #1 { str } { g } n }
50 \cs_new_protected:cpn { \c__keys_props_root_str .str_gset:c } #1
51 { \__keys_variable_set:cnnN {#1} { str } { g } n }
52 \cs_new_protected:cpn { \c__keys_props_root_str .str_gset_x:N } #1
53 { \__keys_variable_set:NnnN #1 { str } { g } x }
54 \cs_new_protected:cpn { \c__keys_props_root_str .str_gset_x:c } #1
55 { \__keys_variable_set:cnnN {#1} { str } { g } x }
56 }

```

15.3 Messages and logging

```

57 <@@=stex_log>
    Warnings and error messages
58 \msg_new:nnn{stex}{error/unknownlanguage}{
59   Unknown~language:~#1
60 }
61 \msg_new:nnn{stex}{warning/nomathhub}{
62   MATHHUB~system~variable~not~found~and~no~
63   \detokenize{\mathhub}~value~set!
64 }
65 \msg_new:nnn{stex}{error/deactivated-macro}{
66   The~\detokenize{#1}~command~is~only~allowed~in~#2!
67 }

```

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

```

68 \cs_new_protected:Nn \stex_debug:nn {
69   \clist_if_in:NnTF \c_stex_debug_clist { all } {
70     \exp_args:Nnnx\msg_set:nnn{stex}{debug / #1}{
71       \\Debug~#1:~#2\\
72     }
73     \msg_none:nn{stex}{debug / #1}
74   }{
75     \clist_if_in:NnT \c_stex_debug_clist { #1 } {
76       \exp_args:Nnnx\msg_set:nnn{stex}{debug / #1}{
77         \\Debug~#1:~#2\\
78       }
79       \msg_none:nn{stex}{debug / #1}
80     }
81   }
82 }

```

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

Redirecting messages:

```

83 \clist_if_in:NnTF \c_stex_debug_clist {all} {
84   \msg_redirect_module:nnn{ stex }{ none }{ term }
85 }{
86   \clist_map_inline:Nn \c_stex_debug_clist {
87     \msg_redirect_name:nnn{ stex }{ debug / ##1 }{ term }
88   }
89 }
90
91 \stex_debug:nn{log}{debug-mode-on}

```

15.4 Persistence

```

92 <@=stex_persist>

```

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

```

93 \iow_new:N \c__stex_persist_sms_iow
94 \AddToHook{begindocument}{
95   \bool_if:NTF \c_stex_persist_mode_bool {
96     \ExplSyntaxOn \input{\jobname.sms} \ExplSyntaxOff
97   } {
98     \iow_open:Nn \c__stex_persist_sms_iow {\jobname.sms}
99   }
100 }
101 \AddToHook{enddocument}{
102   \bool_if:NF \c_stex_persist_mode_bool {
103     \iow_close:N \c__stex_persist_sms_iow
104   }
105 }

```

(End definition for `\c__stex_persist_sms_iow`.)

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

```

106 \cs_new_protected:Nn \stex_add_to_sms:n {
107   \bool_if:NF \c_stex_persist_mode_bool {
108     \iow_now:Nn \c__stex_persist_sms_iow { #1 }

```

```

109 }
110 }

```

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

15.5 HTML Annotations

```

111 <@=stex_annotate>
112 \RequirePackage{scalatex}

```

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

```

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

```

```

\if@latexml Conditional for LATEXML:
\latexml_if_p: 114 \ifcsname if@latexml\endcsname\else
\latexml_if:TF 115 \expandafter\newif\csname if@latexml\endcsname\@latexmlfalse
116 \fi
117
118 \prg_new_conditional:Nnn \latexml_if: {p, T, F, TF} {
119   \if@latexml
120     \prg_return_true:
121   \else:
122     \prg_return_false:
123   \fi:
124 }

```

(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
125 \tl_new:N \l__stex_annotate_arg_tl
126 \tl_const:Nx \c__stex_annotate_emptyarg_tl {
127   \scalatex_if:TF {
128     \scalatex_direct_HTML:n { \c_ampsand_str lrm; }
129   }{-}
130 }

```

(End definition for `\l__stex_annotate_arg_tl` and `\c__stex_annotate_emptyarg_tl`.)

```

\__stex_annotate_checkempty:n
131 \cs_new_protected:Nn \__stex_annotate_checkempty:n {
132   \tl_set:Nn \l__stex_annotate_arg_tl { #1 }
133   \tl_if_empty:NT \l__stex_annotate_arg_tl {
134     \tl_set_eq:NN \l__stex_annotate_arg_tl \c__stex_annotate_emptyarg_tl
135   }
136 }

```

(End definition for `__stex_annotate_checkempty:n`.)

```

\l_stex_html_do_output_bool Whether to (locally) produce HTML output
\stex_if_do_html:
137 \bool_new:N \l_stex_html_do_output_bool
138 \bool_set_true:N \l_stex_html_do_output_bool
139 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
140   \bool_if:nTF \l_stex_html_do_output_bool
141     \prg_return_true: \prg_return_false:
142 }

```

(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

```

143 \cs_new_protected:Nn \stex_suppress_html:n {
144   \exp_args:Nne \use:nn {
145     \bool_set_false:N \l_stex_html_do_output_bool
146     #1
147   }{
148     \stex_if_do_html:T {
149       \bool_set_true:N \l_stex_html_do_output_bool
150     }
151   }
152 }
```

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

`\stex_annotate:env`

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).

`\stex_annotate_invisible:n`

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.

`\stex_annotate_invisible:nnn`

```

153 \scalatex_if:TF{
154   \cs_new_protected:Nn \stex_annotate:nnn {
155     \__stex_annotate_checkempty:n { #3 }
156     \scalatex_annotate_HTML:nn {
157       property="stex:#1" ~
158       resource="#2"
159     } {
160       \tl_use:N \l__stex_annotate_arg_tl
161     }
162   }
163   \cs_new_protected:Nn \stex_annotate_invisible:n {
164     \__stex_annotate_checkempty:n { #1 }
165     \scalatex_annotate_HTML:nn {
166       stex:visible="false" ~
167       style:display="none"
168     } {
169       \tl_use:N \l__stex_annotate_arg_tl
170     }
171   }
172   \cs_new_protected:Nn \stex_annotate_invisible:nnn {
173     \__stex_annotate_checkempty:n { #3 }
174     \scalatex_annotate_HTML:nn {
175       property="stex:#1" ~
176       resource="#2" ~
177       stex:visible="false" ~
178       style:display="none"
179     } {
180       \tl_use:N \l__stex_annotate_arg_tl
181     }
182   }
183   \NewDocumentEnvironment{stex_annotate_env} { m m } {
184     \par
185     \scalatex_annotate_HTML_begin:n {
```

```

186     property="stex:#1" ~
187     resource="#2"
188   }
189 }{
190   \scalatex_annotate_HTML_end:
191 }
192 }{
193   \latexml_if:TF {
194     \cs_new_protected:Nn \stex_annotate:nnn {
195       \__stex_annotate_checkempty:n { #3 }
196       \mode_if_math:TF {
197         \cs:w latexml@annotate@math\cs_end:{#1}{#2}{
198           \tl_use:N \l__stex_annotate_arg_tl
199         }
200       }{
201         \cs:w latexml@annotate@text\cs_end:{#1}{#2}{
202           \tl_use:N \l__stex_annotate_arg_tl
203         }
204       }
205     }
206     \cs_new_protected:Nn \stex_annotate_invisible:n {
207       \__stex_annotate_checkempty:n { #1 }
208       \mode_if_math:TF {
209         \cs:w latexml@invisible@math\cs_end:{
210           \tl_use:N \l__stex_annotate_arg_tl
211         }
212       } {
213         \cs:w latexml@invisible@text\cs_end:{
214           \tl_use:N \l__stex_annotate_arg_tl
215         }
216       }
217     }
218     \cs_new_protected:Nn \stex_annotate_invisible:nnn {
219       \__stex_annotate_checkempty:n { #3 }
220       \cs:w latexml@annotate@invisible\cs_end:{#1}{#2}{
221         \tl_use:N \l__stex_annotate_arg_tl
222       }
223     }
224     \NewDocumentEnvironment{stex_annotate_env} { m m } {
225       \par\begin{latexml@annotateenv}{#1}{#2}
226     }{
227       \end{latexml@annotateenv}
228     }
229   }{
230     \cs_new_protected:Nn \stex_annotate:nnn {#3}
231     \cs_new_protected:Nn \stex_annotate_invisible:n {}
232     \cs_new_protected:Nn \stex_annotate_invisible:nnn {}
233     \NewDocumentEnvironment{stex_annotate_env} { m m } {\par}{\par}
234   }
235 }

```

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

15.6 Languages

236 `<@@=stex_language>`

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

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

```

237 \prop_const_from_keyval:Nn \c_stex_languages_prop {
238   en = english ,
239   de = ngerman ,
240   ar = arabic ,
241   bg = bulgarian ,
242   ru = russian ,
243   fi = finnish ,
244   ro = romanian ,
245   tr = turkish ,
246   fr = french
247 }
248
249 \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop {
250   english   = en ,
251   ngerman   = de ,
252   arabic    = ar ,
253   bulgarian = bg ,
254   russian   = ru ,
255   finnish   = fi ,
256   romanian  = ro ,
257   turkish   = tr ,
258   french    = fr
259 }
260 % todo: chinese simplified (zhs)
261 %       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:

```

262 \clist_if_empty:NF \c_stex_languages_clist {
263   \clist_clear:N \l_tmpa_clist
264   \clist_map_inline:Nn \c_stex_languages_clist {
265     \prop_get:NnNTF \c_stex_languages_prop { #1 } \l_tmpa_str {
266       \clist_put_right:No \l_tmpa_clist \l_tmpa_str
267     } {
268       \msg_error:nxx{stex}{error/unknownlanguage}{\l_tmpa_str}
269     }
270   }
271   \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
272   \RequirePackage[\clist_use:Nn \l_tmpa_clist ,]{babel}
273 }

```

15.7 Activating/Deactivating Macros

`\stex_deactivate_macro:Nn`

```

274 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
275   \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
276   \def#1{

```



```

277     \msg_error:nnnn{stex}{error/deactivated-macro}{#1}{#2}
278   }
279 }

```

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

\stex_reactivate_macro:N

```

280 \cs_new_protected:Nn \stex_reactivate_macro:N {
281   \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname
282 }

```

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

```

283 \</package>

```

Chapter 16

STEX -MathHub Implementation

```
284 <*package>
285
286 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
287
288 <@@=stex_path>
289
290 Warnings and error messages
291 \msg_new:nnn{stex}{error/norepository}{
292   No~archive~#1~found~in~#2
293 }
294 \msg_new:nnn{stex}{error/notinarchive}{
295   Not~currently~in~an~archive,~but~\detokenize{#1}~
296   needs~one!
297 }
298 \msg_new:nnn{stex}{error/nofile}{
299   \detokenize{#1}~could~not~find~file~#2
300 }
```

16.1 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
299 \cs_new_protected:Nn \stex_path_from_string:Nn {
300   \str_set:Nx \l_tmpa_str { #2 }
301   \str_if_empty:NTF \l_tmpa_str {
302     \seq_clear:N #1
303   }{
304     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
305     \sys_if_platform_windows:T{
306       \seq_clear:N \l_tmpa_tl
307       \seq_map_inline:Nn #1 {
308         \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
309         \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl

```

```

310     }
311     \seq_set_eq:NN #1 \l_tmpa_tl
312   }
313   \stex_path_canonicalize:N #1
314 }
315 }
316 \cs_generate_variant:Nn \stex_path_from_string:Nn
317 { 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
318 \cs_new_protected:Nn \stex_path_to_string:NN {
319   \exp_args:NNe \str_set:Nn #2 { \seq_use:Nn #1 / }
320 }
321
322 \cs_new:Nn \stex_path_to_string:N {
323   \seq_use:Nn #1 /
324 }

```

(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
325 \str_const:Nn \c__stex_path_dot_str {.}
326 \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.

```

327 \cs_new_protected:Nn \stex_path_canonicalize:N {
328   \seq_if_empty:NF #1 {
329     \seq_clear:N \l_tmpa_seq
330     \seq_get_left:NN #1 \l_tmpa_tl
331     \str_if_empty:NT \l_tmpa_tl {
332       \seq_put_right:Nn \l_tmpa_seq {}
333     }
334     \seq_map_inline:Nn #1 {
335       \str_set:Nn \l_tmpa_tl { ##1 }
336       \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_dot_str {} {
337         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
338           \seq_if_empty:NTF \l_tmpa_seq {
339             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
340               \c__stex_path_up_str
341             }
342           }{
343             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
344             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
345               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
346                 \c__stex_path_up_str
347               }
348             }{
349               \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
350             }

```

```

351     }
352   }{
353     \str_if_empty:NF \l_tmpa_tl {
354       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
355     }
356   }
357 }
358 }
359 \seq_gset_eq:NN #1 \l_tmpa_seq
360 }
361 }

```

(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`

```

362 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
363   \seq_if_empty:NTF #1 {
364     \prg_return_false:
365   }{
366     \seq_get_left:NN #1 \l_tmpa_tl
367     \str_if_empty:NTF \l_tmpa_tl {
368       \prg_return_true:
369     }{
370       \prg_return_false:
371     }
372   }
373 }

```

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

16.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

374 \str_new:N\l_stex_kpsewhich_return_str
375 \cs_new_protected:Nn \stex_kpsewhich:n {
376   \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
377   \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
378   \tl_trim_spaces:N \l_stex_kpsewhich_return_str
379 }

```

(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`

```

380 \sys_if_platform_windows:TF{
381   \stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
382 }{
383   \stex_kpsewhich:n{-var-value~PWD}
384 }
385
386 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
387 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
388 \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.)

16.3 File Hooks and Tracking

389 `<@@=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

390 `\seq_gclear_new:N\g__stex_files_stack`

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`

`\c_stex_mainfile_str`

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

392 `\stex_path_from_string:Nn \c_stex_mainfile_seq`

393 `\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`.

394 `\seq_gclear_new:N\g_stex_currentfile_seq`

395 `\AddToHook{file/before}{`

396 `\stex_path_from_string:Nn\g_stex_currentfile_seq{\CurrentFilePath}`

397 `\stex_path_if_absolute:NTF\g_stex_currentfile_seq{`

398 `\exp_args:NNe\seq_put_right:Nn\g_stex_currentfile_seq{\CurrentFile}`

399 `}{`

400 `\stex_path_from_string:Nn\g_stex_currentfile_seq{`

401 `\c_stex_pwd_str/\CurrentFilePath/\CurrentFile`

402 `}`

403 `}`

404 `\seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq`

405 `\exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq`

406 `}`

407 `\AddToHook{file/after}{`

408 `\seq_if_empty:NF\g__stex_files_stack{`

409 `\seq_gpop:Nn\g__stex_files_stack\l_tmpa_seq`

410 `}`

411 `\seq_if_empty:NTF\g__stex_files_stack{`

412 `\seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq`

413 `}{`

414 `\seq_get:NN\g__stex_files_stack\l_tmpa_seq`

415 `\seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq`

416 `}`

417 `}`

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

16.4 MathHub Repositories

```

418 <@@=stex_mathhub>

\mathhub
\c_stex_mathhub_seq
\c_stex_mathhub_str
419 \str_if_empty:NTF\mathhub{
420   \stex_kpsewhich:n{-var-value~MATHHUB}
421   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
422
423   \str_if_empty:NTF\c_stex_mathhub_str{
424     \msg_warning:nn{stex}{warning/nomathhub}
425   }{
426     \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
427     \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
428   }
429 }{
430   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
431   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
432     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
433       \c_stex_pwd_str/\mathhub
434     }
435   }
436   \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
437   \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
438 }

```

(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
439 \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
440   \str_set:Nx \l_tmpa_str { #1 }
441   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
442     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
443     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
444     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
445     \__stex_mathhub_find_manifest:N \l_tmpa_seq
446     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
447       \msg_error:nnnn{stex}{error/norepository}{#1}{
448         \stex_path_to_string:N \c_stex_mathhub_str
449       }
450     } {
451       \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
452     }
453   }
454 }

```

(End definition for `__stex_mathhub_do_manifest:n`.)

```

\l__stex_mathhub_manifest_file_seq
455 \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`:

```

456 \cs_new_protected:Nn \_stex_mathhub_find_manifest:N {
457   \seq_set_eq:NN \l_tmpa_seq #1
458   \bool_set_true:N \l_tmpa_bool
459   \bool_while_do:Nn \l_tmpa_bool {
460     \seq_if_empty:NTF \l_tmpa_seq {
461       \bool_set_false:N \l_tmpa_bool
462     }{
463       \file_if_exist:nTF{
464         \stex_path_to_string:N \l_tmpa_seq/MANIFEST.MF
465       }{
466         \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
467         \bool_set_false:N \l_tmpa_bool
468       }{
469         \file_if_exist:nTF{
470           \stex_path_to_string:N \l_tmpa_seq/META-INF/MANIFEST.MF
471         }{
472           \seq_put_right:Nn \l_tmpa_seq{META-INF}
473           \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
474           \bool_set_false:N \l_tmpa_bool
475         }{
476           \file_if_exist:nTF{
477             \stex_path_to_string:N \l_tmpa_seq/meta-inf/MANIFEST.MF
478           }{
479             \seq_put_right:Nn \l_tmpa_seq{meta-inf}
480             \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
481             \bool_set_false:N \l_tmpa_bool
482           }{
483             \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
484           }
485         }
486       }
487     }
488   }
489   \seq_set_eq:NN \l__stex_mathhub_manifest_file_seq \l_tmpa_seq
490 }

```

(End definition for `_stex_mathhub_find_manifest:N`.)

`\c_stex_mathhub_manifest_ior` File variable used for MANIFEST-files

```

491 \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:

```

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

```

```

500 \exp_args:NNe \str_set:Nn \l_tmpb_tl {
501 \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
502 }
503 \exp_args:No \str_case:nnTF \l_tmpa_tl {
504 {id} {
505 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
506 { id } \l_tmpb_tl
507 }
508 {narration-base} {
509 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
510 { narr } \l_tmpb_tl
511 }
512 {url-base} {
513 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
514 { docurl } \l_tmpb_tl
515 }
516 {source-base} {
517 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
518 { ns } \l_tmpb_tl
519 }
520 {ns} {
521 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
522 { ns } \l_tmpb_tl
523 }
524 {dependencies} {
525 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
526 { deps } \l_tmpb_tl
527 }
528 }{}{}
529 }{}
530 }
531 \ior_close:N \c__stex_mathhub_manifest_ior
532 }

```

(End definition for `__stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

533 \cs_new_protected:Nn \stex_set_current_repository:n {
534 \stex_require_repository:n { #1 }
535 \prop_set_eq:Nc \l_stex_current_repository_prop {
536 c_stex_mathhub_#1_manifest_prop
537 }
538 }

```

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

`\stex_require_repository:n`

```

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

```



```

547     narr = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { narr } ,
548     deps = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { deps }
549   }
550 }
551 }
552 }

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

553 \prop_new:N \l_stex_current_repository_prop
554
555 \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
556 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
557   \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
558 } {
559   \__stex_mathhub_parse_manifest:n { main }
560   \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
561   \l_tmpa_str
562   \prop_set_eq:cN { c_stex_mathhub_ \l_tmpa_str _manifest_prop }
563   \c_stex_mathhub_main_manifest_prop
564   \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
565   \stex_debug:nn{mathhub}{Current~repository:~
566     \prop_item:Nn \l_stex_current_repository_prop {id}
567   }
568 }

```

(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.

```

569 \cs_new_protected:Nn \stex_in_repository:nn {
570   \str_set:Nx \l_tmpa_str { #1 }
571   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
572   \str_if_empty:NTF \l_tmpa_str {
573     \exp_args:Ne \l_tmpa_cs{
574       \prop_item:Nn \l_stex_current_repository_prop { id }
575     }
576   }{
577     \stex_require_repository:n \l_tmpa_str
578     \str_set:Nx \l_tmpa_str { #1 }
579     \exp_args:Nne \use:nn {
580       \stex_set_current_repository:n \l_tmpa_str
581       \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
582     }{
583       \stex_set_current_repository:n {
584         \prop_item:Nn \l_stex_current_repository_prop { id }
585       }
586     }
587   }
588 }

```

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

\inputref
\inputref:nn

```

589 \newif \ifinputref \inputreffalse
590
591 \cs_new_protected:Nn \inputref:nn {
592   \stex_in_repository:nn {#1} {
593     \ifinputref
594       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
595     \else
596       \inputreftrue
597       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
598     \inputreffalse
599   \fi
600 }
601 }
602 \NewDocumentCommand \inputref { 0{} m}{
603   \inputref:nn{ #1 }{ #2 }
604 }

```

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

\mhp

```

605 \def \mhp #1 #2 {
606   \exp_args:Ne \str_if_eq:nnTF{#1}{}{
607     \c_stex_mathhub_str /
608     \prop_item:Nn \l_stex_current_repository_prop { id }
609     / source / #2
610   }{
611     \c_stex_mathhub_str / #1 / source / #2
612   }
613 }

```

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

\libinput

```

614 \cs_new_protected:Npn \libinput #1 {
615   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
616     \msg_error:nnn{stex}{error/notinarchive}\libinput
617   }
618   \bool_set_false:N \l_tmpa_bool
619   \tl_clear:N \l_tmpa_tl
620   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
621   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
622   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str
623   \seq_pop_left:NNT \l_tmpb_seq \l_tmpb_str {
624     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
625     \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
626       / meta-inf / lib / #1.tex}{
627       \bool_set_true:N \l_tmpa_bool
628       \tl_put_right:Nx \l_tmpa_tl {
629         \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
630           / meta-inf / lib / #1.tex}
631       }
632     }{}
633 }

```

```

634 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
635 / \l_tmpa_str / lib / #1.tex
636 }{
637   \bool_set_true:N \l_tmpa_bool
638   \tl_put_right:Nx \l_tmpa_tl {
639     \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
640       / \l_tmpa_str / lib / #1.tex}
641   }
642 }{}
643 \bool_if:NF \l_tmpa_bool {
644   \msg_error:nnnn{stex}{error/nofile}\libinput{#1.tex}
645 }
646 \l_tmpa_tl
647 }

```

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

```

648 </package>

```

Chapter 17

STEX -References Implementation

```
649 <*package>
650
651 %%%%%%%%%% references.dtx %%%%%%%%%%
652
653 %\RequirePackage{hyperref}
654 %\RequirePackage{cleveref}
655 <@@=stex_refs>
656
657 Warnings and error messages
658
659 \iow_new:N \c__stex_refs_refs_iow
660 \AddToHook{begindocument}{
661   \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
662 }
663 \AddToHook{enddocument}{
664   \iow_close:N \c__stex_refs_refs_iow
665 }
666
667 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
668
669 \NewDocumentCommand \STEXreftitle { m } {
670   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
671 }
```

17.1 Document URIs and URLs

```
670 \seq_new:N \g__stex_refs_all_refs_seq
671
672 \str_new:N \l_stex_current_docns_str
673
674 \cs_new_protected:Nn \stex_get_document_uri: {
675   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
676   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
677   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
678   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
```

```

679 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
680
681 \str_clear:N \l_tmpa_str
682 \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
683   \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
684 }
685
686 \str_if_empty:NTF \l_tmpa_str {
687   \str_set:Nx \l_stex_current_docns_str {
688     file:/\stex_path_to_string:N \l_tmpa_seq
689   }
690 }{
691   \bool_set_true:N \l_tmpa_bool
692   \bool_while_do:Nn \l_tmpa_bool {
693     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
694     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
695       {source} { \bool_set_false:N \l_tmpa_bool }
696     }{}{
697       \seq_if_empty:NT \l_tmpa_seq {
698         \bool_set_false:N \l_tmpa_bool
699       }
700     }
701   }
702
703   \seq_if_empty:NTF \l_tmpa_seq {
704     \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
705   }{
706     \str_set:Nx \l_stex_current_docns_str {
707       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
708     }
709   }
710 }
711 }
712
713 \str_new:N \l_stex_current_docurl_str
714 \cs_new_protected:Nn \stex_get_document_url: {
715   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
716   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
717   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
718   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
719   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
720
721   \str_clear:N \l_tmpa_str
722   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
723     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
724       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
725     }
726   }
727
728   \str_if_empty:NTF \l_tmpa_str {
729     \str_set:Nx \l_stex_current_docurl_str {
730       file:/\stex_path_to_string:N \l_tmpa_seq
731     }
732   }{
733     \bool_set_true:N \l_tmpa_bool

```

```

733 \bool_while_do:Nn \l_tmpa_bool {
734   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
735   \exp_args:No \str_case:nnTF { \l_tmpb_str } {
736     {source} { \bool_set_false:N \l_tmpa_bool }
737   }{}{
738     \seq_if_empty:NT \l_tmpa_seq {
739       \bool_set_false:N \l_tmpa_bool
740     }
741   }
742 }
743
744 \seq_if_empty:NTF \l_tmpa_seq {
745   \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
746 }{
747   \str_set:Nx \l_stex_current_docurl_str {
748     \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
749   }
750 }
751 }
752 }

```

17.2 Setting Reference Targets

```

753 \str_const:Nn \c__stex_refs_url_str{URL}
754 \str_const:Nn \c__stex_refs_ref_str{REF}
755 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
756   \stex_get_document_uri:
757   \str_set:Nx \l_tmpa_str { #1 }
758   \str_if_empty:NT \l_tmpa_str {
759     \int_zero:N \l_tmpa_int
760     \bool_set_true:N \l_tmpa_bool
761     \bool_while_do:Nn \l_tmpa_bool {
762       \cs_if_exist:cTF {
763         sref_\l_stex_current_docns_str\c_hash_str REF\int_use:N \l_tmpa_int _type
764       }{
765         \int_incr:N \l_tmpa_int
766       }{}
767       \str_set:Nx \l_tmpa_str { REF\int_use:N \l_tmpa_int }
768       \bool_set_false:N \l_tmpa_bool
769     }
770   }
771 }
772 \str_set:Nx \l_tmpa_str {
773   \l_stex_current_docns_str\c_hash_str\l_tmpa_str
774 }
775 \seq_gput_right:No \g__stex_refs_all_refs_seq \l_tmpa_str
776 \stex_if_smsmode:TF {
777   \stex_get_document_url:
778   \str_gset_eq:cN {sref_url_\l_tmpa_str _str}\l_stex_current_docurl_str
779   \str_gset_eq:cN {sref_\l_tmpa_str _type}\c__stex_refs_url_str
780 }{
781   \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~::~\expandafter{\@currentlabel~in~\exp_a
782   \exp_after:wN\label\exp_after:wN{sref_\l_tmpa_str}
783   \str_gset:cn {sref_\l_tmpa_str _type}\c__stex_refs_ref_str

```

```

784 }
785 }
786 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
787   \str_gset_eq:cN {sref_sym_#1_uri} \l_stex_current_docns_str
788 }

```

17.3 Using References

```

789 \keys_define:nn { stex / sref } {
790   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
791   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
792   pre           .tl_set:N = \l__stex_refs_pre_tl ,
793   post          .tl_set:N = \l__stex_refs_post_tl ,
794   indoc         .str_set_x:N = \l__stex_refs_repo_str ,
795 }
796
797 \cs_new_protected:Nn \__stex_refs_args:n {
798   \tl_clear:N \l__stex_refs_linktext_tl
799   \tl_clear:N \l__stex_refs_fallback_tl
800   \tl_clear:N \l__stex_refs_pre_tl
801   \tl_clear:N \l__stex_refs_post_tl
802   \str_clear:N \l__stex_refs_repo_str
803   \keys_set:nn { stex / sref } { #1 }
804 }
805
806 </package>

```

Chapter 18

STEX -Modules Implementation

```
807 <*package>
808
809 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
810
811 <@@=stex_modules>
      Warnings and error messages
812 \msg_new:nnn{stex}{error/unknownmodule}{
813   No~module~#1~found
814 }
815 \msg_new:nnn{stex}{error/syntax}{
816   Syntax~error:~#1
817 }
818 \msg_new:nnn{stex}{error/siglanguage}{
819   Module~#1~declares~signature~#2,~but~does~not~
820   declare~its~language
821 }
```

`\l_stex_current_module_prop` The current module:

```
822 \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

```
823 \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`

```
824 \seq_new:N \g_stex_modules_in_file_seq
825 \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
826 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
827   \prop_if_empty:NTF \l_stex_current_module_prop
828   \prg_return_false: \prg_return_true:
829 }

```

(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
830 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
831   \prop_if_exist:cTF { c_stex_module_#1_prop }
832   \prg_return_true: \prg_return_false:
833 }

```

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

\stex_add_to_current_module:n Only allowed within modules:

```

\STEXexport
834 \cs_new_protected:Nn \stex_add_to_current_module:n {
835   \prop_get:NnN \l_stex_current_module_prop { content } \l_tmpa_tl
836   \tl_put_right:Nn \l_tmpa_tl { #1 }
837   \prop_put:Nno \l_stex_current_module_prop { content } { \l_tmpa_tl }
838 }
839 \cs_new_protected:Npn \STEXexport #1 {
840   #1
841   \stex_add_to_current_module:n { #1 }
842   \stex_smsmode_set_codes:
843 }
844 \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
845 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
846   \str_set:Nx \l_tmpa_str { #1 }
847   \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
848   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
849   \prop_put:Nno \l_stex_current_module_prop { constants } \l_tmpa_seq
850 }

```

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

```

\stex_add_import_to_current_module:n
851 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
852   \str_set:Nx \l_tmpa_str { #1 }
853   \prop_get:NnN \l_stex_current_module_prop { imports } \l_tmpa_seq
854   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
855   \prop_put:Nno \l_stex_current_module_prop { imports } \l_tmpa_seq
856 }

```

(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).

```

857 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
858   \str_set:Nx \l_tmpa_str { #1 }
859   \seq_set_eq:NN \l_tmpa_seq #2
860   % split off file extension
861   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
862   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
863   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
864   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
865
866   \bool_set_true:N \l_tmpa_bool
867   \bool_while_do:Nn \l_tmpa_bool {
868     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
869     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
870       {source} { \bool_set_false:N \l_tmpa_bool }
871     }{}{
872       \seq_if_empty:NT \l_tmpa_seq {
873         \bool_set_false:N \l_tmpa_bool
874       }
875     }
876   }
877
878   \seq_if_empty:NTF \l_tmpa_seq {
879     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
880   }{
881     \str_set:Nx \l_stex_modules_ns_str {
882       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
883     }
884   }
885 }

```

(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`

```

886 \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.

```

887 \cs_new_protected:Nn \stex_modules_current_namespace: {
888   \prop_get:NnTF \l_stex_current_repository_prop { ns } \l_tmpa_str {
889     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
890   }{
891     % split off file extension
892     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
893     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
894     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
895     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
896     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
897     \str_set:Nx \l_stex_modules_ns_str {
898       file:/\stex_path_to_string:N \l_tmpa_seq

```

```

899     }
900   }
901 }

```

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

18.1 The module environment

module arguments:

```

902 \keys_define:nn { stex / module } {
903   title      .str_set_x:N = \l_stex_module_title_str ,
904   ns         .str_set_x:N = \l_stex_module_ns_str ,
905   lang       .str_set_x:N = \l_stex_module_lang_str ,
906   sig        .str_set_x:N = \l_stex_module_sig_str ,
907   creators   .str_set_x:N = \l_stex_module_creators_str ,
908   contributors .str_set_x:N = \l_stex_module_contributors_str ,
909   meta       .str_set_x:N = \l_stex_module_meta_str
910 }
911
912 \cs_new_protected:Nn \__stex_modules_args:n {
913   \str_clear:N \l_stex_module_title_str
914   \str_clear:N \l_stex_module_ns_str
915   \str_clear:N \l_stex_module_lang_str
916   \str_clear:N \l_stex_module_sig_str
917   \str_clear:N \l_stex_module_creators_str
918   \str_clear:N \l_stex_module_contributors_str
919   \str_clear:N \l_stex_module_meta_str
920   \keys_set:nn { stex / module } { #1 }
921 }
922
923 % module parameters here? In the body?
924

```

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

```

925 \cs_new_protected:Nn \stex_module_setup:nn {
926   \str_set:Nx \l_stex_module_name_str { #2 }
927   \__stex_modules_args:n { #1 }
928
929   First, we set up the name and namespace of the module.
930   Are we in a nested module?
931
932   \stex_if_in_module:TF {
933     % Nested module
934     \prop_get:NnN \l_stex_current_module_prop
935       { ns } \l_stex_module_ns_str
936     \str_set:Nx \l_stex_module_name_str {
937       \prop_item:Nn \l_stex_current_module_prop
938         { name } / \l_stex_module_name_str
939     }
940   }{
941     % not nested:
942     \str_if_empty:NT \l_stex_module_ns_str {
943       \stex_modules_current_namespace:
944       \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
945     }
946   }
947 }

```

```

941 \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
942 / {\l_stex_module_ns_str}
943 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
944 \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
945   \str_set:Nx \l_stex_module_ns_str {
946     \stex_path_to_string:N \l_tmpa_seq
947   }
948 }
949 }
950 }

```

Next, we determine the language of the module:

```

951 \str_if_empty:NT \l_stex_module_lang_str {
952   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
953   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
954   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
955   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
956   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
957     \stex_debug:nn{modules} {Language-\l_stex_module_lang_str~
958       inferred~from~file~name}
959     \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
960   }
961 }
962
963 \str_if_empty:NF \l_stex_module_lang_str {
964   \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
965   \l_tmpa_str {
966     \ltx@ifpackageloaded{babel}{
967       \exp_args:Nx \selectlanguage { \l_tmpa_str }
968     }{}
969   } {
970     \msg_error:nnn{stex}{error/unknownlanguage}{\l_tmpa_str}
971   }
972 }

```

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

```

973 \str_if_empty:NTF \l_stex_module_sig_str {
974   \str_clear:N \l_tmpa_str
975   \seq_clear:N \l_tmpa_seq
976   \tl_clear:N \l_tmpa_tl
977   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
978     name      = \l_stex_module_name_str ,
979     ns        = \l_stex_module_ns_str ,
980     imports   = \exp_not:o { \l_tmpa_seq } ,
981     constants = \exp_not:o { \l_tmpa_seq } ,
982     content   = \exp_not:o { \l_tmpa_tl } ,
983     file      = \exp_not:o { \g_stex_currentfile_seq } ,
984     lang      = \l_stex_module_lang_str ,
985     sig       = \l_stex_module_sig_str ,
986     meta      = \l_stex_module_meta_str
987   }
988 }{
989   \str_if_empty:NT \l_stex_module_lang_str {

```

```

990     \msg_error:nnnn{stex}{error/siglanguage}{
991       \l_stex_module_ns_str?\l_stex_module_name_str
992     }{\l_stex_module_sig_str}
993   }
994
995   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
996   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
997   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
998   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
999   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1000   \str_set:Nx \l_tmpa_str {
1001     \stex_path_to_string:N \l_tmpa_seq /
1002     \l_tmpa_str . \l_stex_module_sig_str .tex
1003   }
1004   \IfFileExists \l_tmpa_str {
1005     \exp_args:No \stex_in_smsmode:nn { \l_tmpa_str } {
1006       \seq_clear:N \l_stex_all_modules_seq
1007       \prop_clear:N \l_stex_current_module_prop
1008       \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1009       \input { \l_tmpa_str }
1010     }
1011   }{
1012     \msg_error:nnn{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1013   }
1014   \stex_activate_module:n {
1015     \l_stex_module_ns_str ? \l_stex_module_name_str
1016   }
1017   \prop_set_eq:Nc \l_stex_current_module_prop {
1018     c_stex_module_
1019     \l_stex_module_ns_str ?
1020     \l_stex_module_name_str
1021     _prop
1022   }
1023 }

```

We load the metatheory:

```

1024 \str_if_empty:NT \l_stex_module_meta_str {
1025   \str_set:Nx \l_stex_module_meta_str {
1026     \c_stex_metatheory_ns_str ? Metatheory
1027   }
1028 }
1029 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1030   \exp_args:Nx \stex_add_to_current_module:n {
1031     \stex_activate_module:n {\l_stex_module_meta_str}
1032   }
1033   \stex_activate_module:n {\l_stex_module_meta_str}
1034 }
1035 }

```

(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}`

```

1036 \cs_new_protected:Nn \__stex_modules_begin_module:nn {
1037   \stex_reactivate_macro:N \STEXexport
1038   \stex_reactivate_macro:N \importmodule
1039   \stex_reactivate_macro:N \symdecl
1040   \stex_reactivate_macro:N \notation
1041   \stex_reactivate_macro:N \symdef
1042   \stex_module_setup:nn{#1}{#2}
1043
1044   \stex_debug:nn{modules}{
1045     New~module:\\
1046     Namespace:~\l_stex_module_ns_str\\
1047     Name:~\l_stex_module_name_str\\
1048     Language:~\l_stex_module_lang_str\\
1049     Signature:~\l_stex_module_sig_str\\
1050     Metatheory:~\l_stex_module_meta_str\\
1051     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1052   }
1053
1054   \seq_put_right:Nx \l_stex_all_modules_seq {
1055     \l_stex_module_ns_str ? \l_stex_module_name_str
1056   }
1057
1058   \seq_gput_right:Nx \g_stex_modules_in_file_seq
1059     { \l_stex_module_ns_str ? \l_stex_module_name_str }
1060
1061   \stex_if_smsmode:TF {
1062     \stex_smsmode_set_codes:
1063   } {
1064     \begin{stex_annotate_env} {theory} {
1065       \l_stex_module_ns_str ? \l_stex_module_name_str
1066     }
1067
1068     \stex_annotate_invisible:nnn{header}{} {
1069       \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1070       \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1071       \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1072         \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1073       }
1074     }
1075   }
1076   % TODO: Inherit metatheory for nested modules?
1077 }
1078 \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}

```

1079 \cs_new_protected:Nn \__stex_modules_end_module: {
1080   \str_set:Nx \l_tmpa_str {
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 %^^A \prop_new:c { \l_tmpa_str }
1087 \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
1088 \stex_debug:nn{modules}{Closing~module~\prop_item:Nn \l_stex_current_module_prop { name }}
1089 }

```

(End definition for `_stex_modules_end_module:.`)

@module The core environment, with no header

```

1090 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1091 \NewDocumentEnvironment { @module } { 0{} m } {
1092   \par
1093   \_stex_modules_begin_module:nn{#1}{#2}
1094 } {
1095   \_stex_modules_end_module:
1096   \stex_if_smsmode:TF {
1097     \exp_args:Nx \stex_add_to_sms:n {
1098       \prop_gset_from_keyval:cn {
1099         c_stex_module_
1100         \prop_item:Nn \l_stex_current_module_prop { ns } ?
1101         \prop_item:Nn \l_stex_current_module_prop { name }
1102         _prop
1103       } {
1104         name      = \prop_item:cn { \l_tmpa_str } { name } ,
1105         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
1106         imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
1107         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
1108         content   = \prop_item:cn { \l_tmpa_str } { content } ,
1109         file      = \prop_item:cn { \l_tmpa_str } { file } ,
1110         lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
1111         sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
1112         meta      = \prop_item:cn { \l_tmpa_str } { meta }
1113       }
1114     }
1115   }{
1116     \end{stex_annotate_env}
1117   }
1118 }

```

\stex_modules_heading: Code for document headers

```

1119 \cs_if_exist:NTF \thesection {
1120   \newcounter{module}[section]
1121 }{
1122   \newcounter{module}
1123 }
1124
1125 \bool_if:NT \c_stex_showmods_bool {
1126   \latexml_if:F { \RequirePackage{mdframed} }
1127 }
1128
1129 \cs_new_protected:Nn \stex_modules_heading: {
1130   \stepcounter{module}
1131   \par
1132   \bool_if:NT \c_stex_showmods_bool {
1133     \noindent{\textbf{Module}} ~

```

```

1134     \cs_if_exist:NT \thesection {\thesection.}
1135     \themodule ~ [\l_stex_module_name_str]
1136   }
1137   \str_if_empty:NTF \l_stex_module_title_str {
1138   }{
1139     \quad(\l_stex_module_title_str)\hfill
1140   }\par
1141 }
1142 \edef\@currentlabel{Module~\thesection.\themodule~[\l_stex_module_name_str]}
1143 % TODO
1144 \stex_ref_new_doc_target:n \l_stex_module_name_str
1145 }

```

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

Finally:

```

1146 \NewDocumentEnvironment { module } { 0{} m } {
1147   \bool_if:NT \c_stex_showmods_bool {
1148     \begin{mdframed}
1149   }
1150   \begin{@module} [#1] {#2}
1151   \stex_modules_heading:
1152 }{
1153   \end{@module}
1154   \bool_if:NT \c_stex_showmods_bool {
1155     \end{mdframed}
1156   }
1157 }

```

18.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n
1158 \NewDocumentCommand \STEXModule { m } {
1159   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1160   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1161   \tl_set:Nn \l_tmpa_tl {
1162     \msg_error:nnn{stex}{error/unknownmodule}{#1}
1163   }
1164   \seq_map_inline:Nn \l_stex_all_modules_seq {
1165     \str_set:Nn \l_tmpb_str { ##1 }
1166     \str_if_eq:eeT { \l_tmpa_str } {
1167       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1168     } {
1169       \seq_map_break:n {
1170         \tl_set:Nn \l_tmpa_tl {
1171           \stex_invoke_module:n { ##1 }
1172         }
1173       }
1174     }
1175   }
1176   \l_tmpa_tl
1177 }
1178
1179 \cs_new_protected:Nn \stex_invoke_module:n {

```



```

1180 \stex_debug:nn{modules}{Invoking~module~#1}
1181 \peek_charcode_remove:NTF ! {
1182   \__stex_modules_invoke_uri:nN { #1 }
1183 } {
1184   \peek_charcode_remove:NTF ? {
1185     \__stex_modules_invoke_symbol:nn { #1 }
1186   } {
1187     \msg_error:nnn{stex}{error/syntax}{
1188       ?~or~!~expected~after~
1189       \c_backslash_str STEXModule{#1}
1190     }
1191   }
1192 }
1193 }
1194
1195 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1196   \str_set:Nn #2 { #1 }
1197 }
1198
1199 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1200   \stex_invoke_symbol:n{#1?#2}
1201 }

```

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

`\stex_activate_module:n`

```

1202 \cs_new_protected:Nn \stex_activate_module:n {
1203   \stex_debug:nn{modules}{Activating~module~#1}
1204   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1205     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1206     \prop_item:cn { c_stex_module_#1_prop } { content }
1207   }
1208 }

```

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

```

1209 </package>

```

Chapter 19

STEX -Module Inheritance Implementation

```
1210 <*package>
1211
1212 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1213
```

19.1 SMS Mode

```
1214 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq
1215 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1216 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1217 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1218
1219 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1220   \makeatletter
1221   \makeatother
1222   \ExplSyntaxOn
1223   \ExplSyntaxOff
1224 }
1225
1226 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1227   \symdef
1228   \importmodule
1229   \notation
1230   \symdecl
1231   \STEXexport
1232 }
1233
1234 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1235   \tl_to_str:n {
1236     module,
1237     @module
```

```

1238 }
1239 }

```

(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

```

```

1240 \bool_new:N \g__stex_smsmode_bool
1241 \bool_set_false:N \g__stex_smsmode_bool
1242 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1243   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1244 }

```

(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

```

```

1245 \bool_new:N \g__stex_smsmode_catcode_bool
1246 \bool_set_false:N \g__stex_smsmode_catcode_bool
1247 \prg_new_conditional:Nnn \__stex_smsmode_if_catcodes: { p, T, F, TF } {
1248   \bool_if:NTF \g__stex_smsmode_catcode_bool
1249   \prg_return_true: \prg_return_false:
1250 }

```

(End definition for `__stex_smsmode_if_catcodes:TF`.)

```

\stex_smsmode_set_codes:

```

```

1251 \cs_new_protected:Nn \stex_smsmode_set_codes: {
1252   \stex_if_smsmode:T {
1253     \__stex_smsmode_if_catcodes:F {
1254       \bool_gset_true:N \g__stex_smsmode_catcode_bool
1255       \exp_after:wN \char_gset_active_eq:NN
1256       \c_backslash_str \__stex_smsmode_cs:
1257       \tex_global:D \char_set_catcode_active:N \
1258       \tex_global:D \char_set_catcode_other:N $
1259       \tex_global:D \char_set_catcode_other:N ^
1260       \tex_global:D \char_set_catcode_other:N _
1261       \tex_global:D \char_set_catcode_other:N &
1262       \tex_global:D \char_set_catcode_other:N ##
1263     }
1264   }
1265 } \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.

```

1266 \cs_new_protected:Nn \__stex_smsmode_unset_codes: {
1267   \__stex_smsmode_if_catcodes:T {
1268     \bool_gset_false:N \g__stex_smsmode_catcode_bool
1269     \exp_after:wN \tex_global:D \exp_after:wN
1270     \char_set_catcode_escape:N \c_backslash_str
1271     \tex_global:D \char_set_catcode_math_toggle:N $
1272     \tex_global:D \char_set_catcode_math_superscript:N ^
1273     \tex_global:D \char_set_catcode_math_subscript:N _
1274     \tex_global:D \char_set_catcode_alignment:N &
1275     \tex_global:D \char_set_catcode_parameter:N ##
1276   }
1277 } \iffalse $ \fi % to make syntax highlighting work again

```

(End definition for `_stex_smsmode_unset_codes:`.)

`\stex_in_smsmode:nn`

```

1278 \cs_new_protected:Nn \stex_in_smsmode:nn {
1279   \vbox_set:Nn \l_tmpa_box {
1280     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1281     \bool_gset_true:N \g__stex_smsmode_bool
1282     \stex_smsmode_set_codes:
1283     #2
1284     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1285     \stex_if_smsmode:F {
1286       \__stex_smsmode_unset_codes:
1287     }
1288   }
1289   \box_clear:N \l_tmpa_box
1290 }

```

(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:

```

1291 \cs_new_protected:Nn \_stex_smsmode_cs: {
1292   \str_clear:N \l_tmpa_str
1293   \peek_analysis_map_inline:n {
1294     % #1: token (one expansion)
1295     % #2: charcode
1296     % #3 catcode
1297     \token_if_eq_charcode:NNTF ##3 B {
1298       % token is a letter
1299       \exp_args:NNNo \str_put_right:Nn \l_tmpa_str { ##1 }
1300     } {
1301       \str_if_empty:NTF \l_tmpa_str {
1302         % we don't allow (or need) single non-letter CSs
1303         % for now
1304         \peek_analysis_map_break:
1305       }{
1306         \str_if_eq:onTF \l_tmpa_str { begin } {
1307           \peek_analysis_map_break:n {
1308             \exp_after:wN \_stex_smsmode_checkbegin:n ##1
1309           }
1310         } {
1311           \str_if_eq:onTF \l_tmpa_str { end } {
1312             \peek_analysis_map_break:n {
1313               \exp_after:wN \_stex_smsmode_checkend:n ##1
1314             }
1315           } {
1316             \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
1317             \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1318               \g_stex_smsmode_allowedmacros_tl
1319               { \use:c{\l_tmpa_str} } {
1320               \stex_debug:nn{modules}{Executing-1:~\l_tmpa_str}
1321               \peek_analysis_map_break:n {
1322                 \exp_after:wN \l_tmpa_tl ##1
1323               }

```

```

1324     } {
1325         \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1326         \g_stex_smsmode_allowedmacros_escape_tl
1327         { \use:c{\l_tmpa_str} } {
1328             \__stex_smsmode_unset_codes:
1329             \stex_debug:nn{modules}{Executing~2:~\l_tmpa_str}
1330             % TODO \__stex_smsmode_rescan_cs:
1331             % \int_compare:nNnTF {##2} = {92} {
1332             %     \peek_analysis_map_break:n {
1333             %         \__stex_smsmode_unset_codes:
1334             %         \__stex_smsmode_rescan_cs:
1335             %     }
1336             % } {
1337             %     \peek_analysis_map_break:n {
1338             %         \exp_after:wN \l_tmpa_tl ##1
1339             %     }
1340             % }
1341             } {
1342                 \int_compare:nNnTF {##2} = {92} {
1343                     \peek_analysis_map_break:n { \__stex_smsmode_cs: }
1344                 }{
1345                     \peek_analysis_map_break:n { \exp_after:wN\relax ##1 }
1346                 }
1347             }
1348         }
1349     }
1350 }
1351 }
1352 }
1353 }
1354 }

```

(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:

```

1355 \cs_new_protected:Nn \__stex_smsmode_rescan_cs: {
1356     \str_clear:N \l_tmpb_str
1357     \peek_analysis_map_inline:n {
1358         \token_if_eq_charcode:NNTF ##3 B {
1359             % token is a letter
1360             \exp_args:NNNo \str_put_right:Nn \l_tmpb_str { ##1 }
1361         } {
1362             \peek_analysis_map_break:n {
1363                 \exp_after:wN \use:c \exp_after:wN {
1364                     \exp_after:wN \l_tmpa_str\exp_after:wN
1365                 } \use:c { \l_tmpb_str \exp_after:wN } ##1
1366             }
1367         }
1368     }
1369 }

```

(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.

```

1370 \cs_new_protected:Nn \__stex_smsmode_checkbegin:n {
1371   \str_set:Nn \l_tmpa_str { #1 }
1372   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1373     \__stex_smsmode_unset_codes:
1374     \begin{#1}
1375   }
1376 }

```

(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.

```

1377 \cs_new_protected:Nn \__stex_smsmode_checkend:n {
1378   \str_set:Nn \l_tmpa_str { #1 }
1379   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1380     \end{#1}
1381   }
1382 }

```

(End definition for `__stex_smsmode_checkend:n`.)

19.2 Inheritance

1383 `\@@=stex_importmodule`

`\stex_import_module_uri:nn`

```

1384 \cs_new_protected:Nn \stex_import_module_uri:nn {
1385   \str_set:Nx \l__stex_importmodule_archive_str { #1 }
1386   \str_set:Nn \l__stex_importmodule_path_str { #2 }
1387   \str_if_empty:NT \l__stex_importmodule_archive_str {
1388     \prop_if_empty:NF \l_stex_current_repository_prop {
1389       \prop_get:NnN \l_stex_current_repository_prop { id } \l__stex_importmodule_archive_str
1390     }
1391   }
1392
1393   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l__stex_importmodule_path_str }
1394   \seq_pop_right:NN \l_tmpb_seq \l__stex_importmodule_name_str
1395   \str_set:Nx \l__stex_importmodule_path_str { \seq_use:Nn \l_tmpb_seq ? }
1396
1397   \str_if_empty:NTF \l__stex_importmodule_archive_str {
1398     \stex_modules_current_namespace:
1399     \str_if_empty:NF \l__stex_importmodule_path_str {
1400       \str_set:Nx \l_stex_module_ns_str {
1401         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1402       }
1403     }
1404   }{
1405     \stex_require_repository:n \l__stex_importmodule_archive_str
1406     \prop_get:cnN { c_stex_mathhub\l__stex_importmodule_archive_str _manifest_prop } { ns }
1407     \l_stex_module_ns_str
1408     \str_if_empty:NF \l__stex_importmodule_path_str {
1409       \str_set:Nx \l_stex_module_ns_str {
1410         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1411       }

```

```

1412     }
1413   }
1414 }

```

(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 1415 \str_new:N \l__stex_importmodule_name_str
\l_stex_importmodule_path_str 1416 \str_new:N \l__stex_importmodule_archive_str
\l_stex_importmodule_file_str 1417 \str_new:N \l__stex_importmodule_path_str
1418 \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>}
1419 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1420   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1421
1422     % archive
1423     \str_set:Nx \l_tmpa_str { #2 }
1424     \str_if_empty:NTF \l_tmpa_str {
1425       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1426     } {
1427       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1428       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1429       \seq_put_right:Nn \l_tmpa_seq { source }
1430     }
1431
1432     % path
1433     \str_set:Nx \l_tmpb_str { #3 }
1434     \str_if_empty:NTF \l_tmpb_str {
1435       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1436
1437       \ltx@ifpackageloaded{babel} {
1438         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1439           { \language } \l_tmpb_str {
1440           \msg_error:nnn{stex}{error/unknownlanguage}{\language}
1441         }
1442       } {
1443         \str_clear:N \l_tmpb_str
1444       }
1445
1446       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1447       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1448         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1449       }{
1450         \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1451         \IfFileExists{ \l_tmpa_str.tex }{
1452           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1453         }{
1454           % try english as default
1455           \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1456           \IfFileExists{ \l_tmpa_str.en.tex }{
1457             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }

```

```

1458         }{
1459             \msg_error:nnn{stex}{error/unknownmodule}{#1?#4}
1460         }
1461     }
1462 }
1463
1464 } {
1465     \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1466     \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1467
1468     \ltx@ifpackageloaded{babel} {
1469         \exp_args:NnX \prop_get:NnNF \c_stex_language_abbrevs_prop
1470             { \language } \l_tmpb_str {
1471             \msg_error:nnn{stex}{error/unknownlanguage}{\language}
1472         }
1473     } {
1474         \str_clear:N \l_tmpb_str
1475     }
1476
1477     \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1478
1479     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1480     \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1481         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1482     }{
1483         \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1484         \IfFileExists{ \l_tmpa_str/#4.tex }{
1485             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1486         }{
1487             % try english as default
1488             \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1489             \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1490                 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1491             }{
1492                 \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1493                 \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1494                     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1495                 }{
1496                     \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1497                     \IfFileExists{ \l_tmpa_str.tex }{
1498                         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1499                     }{
1500                         % try english as default
1501                         \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1502                         \IfFileExists{ \l_tmpa_str.en.tex }{
1503                             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1504                         }{
1505                             \msg_error:nnn{stex}{error/unknownmodule}{#1?#4}
1506                         }
1507                     }
1508                 }
1509             }
1510         }
1511     }

```



```

1512     }
1513
1514     \seq_set_eq:NN \l_tmpa_seq \g_stex_modules_in_file_seq
1515     \seq_clear:N \g_stex_modules_in_file_seq
1516     % \exp_args:Nnx \use:nn {
1517         \exp_args:No \stex_in_smsmode:nn { \g__stex_importmodule_file_str } {
1518             \seq_clear:N \l_stex_all_modules_seq
1519             \prop_clear:N \l_stex_current_module_prop
1520             \str_set:Nx \l_tmpb_str { #2 }
1521             \str_if_empty:NF \l_tmpb_str {
1522                 \stex_set_current_repository:n { #2 }
1523             }
1524             \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1525             \input { \g__stex_importmodule_file_str }
1526         }
1527     % }{
1528
1529     % }
1530     \prop_gput:Noo \g_stex_module_files_prop
1531     \g__stex_importmodule_file_str \g_stex_modules_in_file_seq
1532     \seq_set_eq:NN \g_stex_modules_in_file_seq \l_tmpa_seq
1533
1534     \stex_if_module_exists:nF { #1 ? #4 } {
1535         \msg_error:nnn{stex}{error/unknownmodule}{
1536             #1?#4~(in~file~\g__stex_importmodule_file_str)
1537         }
1538     }
1539 }
1540 \stex_activate_module:n { #1 ? #4 }
1541 }

```

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

`\importmodule`

```

1542 \NewDocumentCommand \importmodule { O{} m } {
1543     \stex_import_module_uri:nn { #1 } { #2 }
1544     \stex_debug:nn{modules}{Importing~module:~
1545         \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1546     }
1547     \stex_if_smsmode:F {
1548         \stex_import_require_module:nnnn
1549         { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1550         { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1551         \stex_annotate_invisible:nnn
1552         {import} { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } {}
1553     }
1554     \exp_args:Nx \stex_add_to_current_module:n {
1555         \stex_import_require_module:nnnn
1556         { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1557         { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1558     }
1559     \exp_args:Nx \stex_add_import_to_current_module:n {
1560         \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1561     }

```

```

1562 \stex_smsmode_set_codes:
1563 }
1564 \stex_deactivate_macro:Nn \importmodule {module-environments}

```

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

\usemodule

```

1565 \NewDocumentCommand \usemodule { 0{} m } {
1566   \stex_if_smsmode:F {
1567     \stex_import_module_uri:nn { #1 } { #2 }
1568     \stex_import_require_module:nnnn
1569     { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1570     { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1571     \stex_annotate_invisible:nnn
1572     {usemodule} {\l_stex_module_ns_str ? \l__stex_importmodule_name_str} {}
1573   }
1574   \stex_smsmode_set_codes:
1575 }

```

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

```

1576 \endpackage

```

Chapter 20

STEX -Symbols Implementation

```
1577 <*package>
1578
1579 %%%%%%%%%%%%% symbols.dtx %%%%%%%%%%%%%
1580
Warnings and error messages
1581
```

20.1 Symbol Declarations

```
1582 <@@=stex_symdecl>

\l_stex_all_symbols_seq Stores all available symbols
1583 \seq_new:N \l_stex_all_symbols_seq

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

\STEXsymbol

1584 \NewDocumentCommand \STEXsymbol { m } {
1585   \stex_get_symbol:n { #1 }
1586   \exp_args:No
1587   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1588 }

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

1589 \keys_define:nn { stex / symdecl } {
1590   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1591   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1592   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1593   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
1594   align     .str_set:N    = \l_stex_symdecl_align_str , % TODO(?)
1595   gfc       .str_set:N    = \l_stex_symdecl_gfc_str , % TODO(?)
1596   specializes .str_set:N  = \l_stex_symdecl_specializes_str , % TODO(?)
1597   def       .tl_set:N    = \l_stex_symdecl_definiens_tl
1598 }
```

```

1599
1600 \bool_new:N \l_stex_symdecl_make_macro_bool
1601
1602 \cs_new_protected:Nn \__stex_symdecl_args:n {
1603   \str_clear:N \l_stex_symdecl_name_str
1604   \str_clear:N \l_stex_symdecl_args_str
1605   \bool_set_false:N \l_stex_symdecl_local_bool
1606   \tl_clear:N \l_stex_symdecl_type_tl
1607   \tl_clear:N \l_stex_symdecl_definiens_tl
1608
1609   \keys_set:nn { stex / symdecl } { #1 }
1610 }

```

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

```

1611
1612 \NewDocumentCommand \symdecl { s O{} m } {
1613   \__stex_symdecl_args:n { #2 }
1614   \IfBooleanTF #1 {
1615     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1616   } {
1617     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1618   }
1619   \stex_symdecl_do:n { #3 }
1620   \stex_smsmode_set_codes:
1621 }
1622 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

\stex_symdecl_do:n

```

1623 \cs_new_protected:Nn \stex_symdecl_do:n {
1624   \stex_if_in_module:F {
1625     % TODO throw error? some default namespace?
1626   }
1627
1628   \str_if_empty:NT \l_stex_symdecl_name_str {
1629     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1630   }
1631
1632   \prop_if_exist:cT { g_stex_symdecl_
1633     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1634     \prop_item:Nn \l_stex_current_module_prop {name} ?
1635     \l_stex_symdecl_name_str
1636     _prop
1637   }{
1638     % TODO throw error (beware of circular dependencies)
1639   }
1640
1641   \prop_clear:N \l_tmpa_prop
1642   \prop_put:Nnx \l_tmpa_prop { module } {
1643     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1644     \prop_item:Nn \l_stex_current_module_prop {name}
1645   }

```

```

1646 \seq_clear:N \l_tmpa_seq
1647 \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
1648 \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1649 \prop_put:Nno \l_tmpa_prop { local } \l_stex_symdecl_local_bool
1650 \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1651
1652 \exp_args:No \stex_add_constant_to_current_module:n {
1653   \l_stex_symdecl_name_str
1654 }
1655
1656 % arity/args
1657 \int_zero:N \l_tmpb_int
1658
1659 \bool_set_true:N \l_tmpa_bool
1660 \str_map_inline:Nn \l_stex_symdecl_args_str {
1661   \token_case_meaning:NnF ##1 {
1662     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1663     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1664     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1665     {\tl_to_str:n a} {
1666       \bool_set_false:N \l_tmpa_bool
1667       \int_incr:N \l_tmpb_int
1668     }
1669     {\tl_to_str:n B} {
1670       \bool_set_false:N \l_tmpa_bool
1671       \int_incr:N \l_tmpb_int
1672     }
1673   }{
1674     \msg_set:nnn{stex}{error/wrongargs}{
1675       args~value~in~symbol~declaration~for~
1676       \prop_item:Nn \l_stex_current_module_prop {ns} ?
1677       \prop_item:Nn \l_stex_current_module_prop {name} ?
1678       \l_stex_symdecl_name_str ~
1679       needs~to~be~
1680       i,~a,~b~or~B,~but~##1~given
1681     }
1682     \msg_error:nn{stex}{error/wrongargs}
1683   }
1684 }
1685 \bool_if:NTF \l_tmpa_bool {
1686   % possibly numeric
1687   \str_if_empty:NTF \l_stex_symdecl_args_str {
1688     \prop_put:Nnn \l_tmpa_prop { args } {}
1689     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1690   }{
1691     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1692     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1693     \str_clear:N \l_tmpa_str
1694     \int_step_inline:nn \l_tmpa_int {
1695       \str_put_right:Nn \l_tmpa_str i
1696     }
1697     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1698   }
1699 } {

```

```

1700     \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1701     \prop_put:Nnx \l_tmpa_prop { arity }
1702         { \str_count:N \l_stex_symdecl_args_str }
1703 }
1704 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
1705
1706
1707 % semantic macro
1708
1709 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1710     \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1711         \prop_item:Nn \l_tmpa_prop { module } ?
1712         \prop_item:Nn \l_tmpa_prop { name }
1713     } }
1714
1715     \bool_if:NF \l_stex_symdecl_local_bool {
1716         \exp_args:Nx \stex_add_to_current_module:n {
1717             \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1718                 \prop_item:Nn \l_tmpa_prop { module } ?
1719                 \prop_item:Nn \l_tmpa_prop { name }
1720             } }
1721         }
1722     }
1723 }
1724
1725 % add to all symbols
1726
1727 \bool_if:NF \l_stex_symdecl_local_bool {
1728     \exp_args:Nx \stex_add_to_current_module:n {
1729         \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1730             \prop_item:Nn \l_tmpa_prop { module } ?
1731             \prop_item:Nn \l_tmpa_prop { name }
1732         }
1733     }
1734 }
1735
1736 \stex_debug:nn{symbols}{New~symbol:~
1737     \prop_item:Nn \l_tmpa_prop { module } ?
1738     \prop_item:Nn \l_tmpa_prop { name } ^^J
1739     Type:~\exp_not:o { \l_stex_symdecl_type_tl } ^^J
1740     Args:~\prop_item:Nn \l_tmpa_prop { args }
1741 }
1742
1743 % circular dependencies require this:
1744
1745 \prop_if_exist:cF {
1746     g_stex_symdecl_
1747     \prop_item:Nn \l_tmpa_prop { module } ?
1748     \prop_item:Nn \l_tmpa_prop { name }
1749     _prop
1750 } {
1751     \prop_gset_eq:cN {
1752         g_stex_symdecl_
1753         \prop_item:Nn \l_tmpa_prop { module } ?

```

```

1754     \prop_item:Nn \l_tmpa_prop { name }
1755     _prop
1756   } \l_tmpa_prop
1757 }
1758
1759 \stex_if_smsmode:TF {
1760   \bool_if:NF \l_stex_symdecl_local_bool {
1761     \exp_args:Nx \stex_add_to_sms:n {
1762       \prop_gset_from_keyval:cn {
1763         g_stex_symdecl_
1764         \prop_item:Nn \l_tmpa_prop { module } ?
1765         \prop_item:Nn \l_tmpa_prop { name }
1766         _prop
1767       } {
1768         name      = \prop_item:Nn \l_tmpa_prop { name }      ,
1769         module    = \prop_item:Nn \l_tmpa_prop { module }    ,
1770         notations = \prop_item:Nn \l_tmpa_prop { notations } ,
1771         local     = \prop_item:Nn \l_tmpa_prop { local }     ,
1772         type      = \prop_item:Nn \l_tmpa_prop { type }      ,
1773         args      = \prop_item:Nn \l_tmpa_prop { args }      ,
1774         arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
1775         assocs    = \prop_item:Nn \l_tmpa_prop { assocs }
1776       }
1777       \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1778         \prop_item:Nn \l_tmpa_prop { module } ?
1779         \prop_item:Nn \l_tmpa_prop { name }
1780       }
1781     }
1782   }
1783 }{
1784   \exp_args:NNx \seq_put_right:Nn \l_stex_all_symbols_seq {
1785     \prop_item:Nn \l_tmpa_prop { module } ?
1786     \prop_item:Nn \l_tmpa_prop { name }
1787   }
1788   \stex_if_do_html:T {
1789     \stex_annotate_invisible:nnn {symdecl} {
1790       \prop_item:Nn \l_tmpa_prop { module } ?
1791       \prop_item:Nn \l_tmpa_prop { name }
1792     } {
1793       \stex_annotate_invisible:nnn{type}{}{\l_stex_symdecl_type_tl$}
1794       \stex_annotate_invisible:nnn{args}{}{
1795         \prop_item:Nn \l_tmpa_prop { args }
1796       }
1797       \stex_annotate_invisible:nnn{macroname}{}{#1}
1798       \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
1799         \stex_annotate_invisible:nnn{definiens}{}
1800         {\l_stex_symdecl_definiens_tl$}
1801       }
1802     }
1803   }
1804 }
1805 }

```

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

`\stex_get_symbol:n`

```
1806 \str_new:N \l_stex_get_symbol_uri_str
1807
1808 \cs_new_protected:Nn \stex_get_symbol:n {
1809   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
1810     \__stex_symdecl_get_symbol_from_cs:n { #1 }
1811   }{
1812     % argument is a string
1813     % is it a command name?
1814     \cs_if_exist:cTF { #1 }{
1815       \cs_set_eq:Nc \l_tmpa_tl { #1 }
1816       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
1817       \str_if_empty:NTF \l_tmpa_str {
1818         \exp_args:Nx \cs_if_eq:NNTF {
1819           \tl_head:N \l_tmpa_tl
1820         } \stex_invoke_symbol:n {
1821           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
1822         }{
1823           \__stex_symdecl_get_symbol_from_string:n { #1 }
1824         }
1825       } {
1826         \__stex_symdecl_get_symbol_from_string:n { #1 }
1827       }
1828     }{
1829       % argument is not a command name
1830       \__stex_symdecl_get_symbol_from_string:n { #1 }
1831       % \l_stex_all_symbols_seq
1832     }
1833   }
1834 }
1835
1836 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
1837   \str_set:Nn \l_tmpa_str { #1 }
1838   \bool_set_false:N \l_tmpa_bool
1839   \stex_if_in_module:T {
1840     \prop_get:NnN \l_stex_current_module_prop
1841     { constants } \l_tmpa_seq
1842     \exp_args:NNo \seq_if_in:NnT \l_tmpa_seq { \l_tmpa_str } {
1843       \bool_set_true:N \l_tmpa_bool
1844       \str_set:Nx \l_stex_get_symbol_uri_str {
1845         \prop_item:Nn \l_stex_current_module_prop { ns } ?
1846         \prop_item:Nn \l_stex_current_module_prop { name } ? #1
1847       }
1848     }
1849   }
1850   \bool_if:NF \l_tmpa_bool {
1851     \tl_set:Nn \l_tmpa_tl {
1852       \msg_set:nnn{stex}{error/unknownsymbol}{
1853         No~symbol~#1~found!
1854       }
1855     }
1856     \msg_error:nn{stex}{error/unknownsymbol}
1857   }
1858   \str_set:Nn \l_tmpa_str { #1 }
1859   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
```



```

1859 \seq_map_inline:Nn \l_stex_all_symbols_seq {
1860   \str_set:Nn \l_tmpb_str { ##1 }
1861   \str_if_eq:eeT { \l_tmpa_str } {
1862     \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1863   } {
1864     \seq_map_break:n {
1865       \tl_set:Nn \l_tmpa_tl {
1866         \str_set:Nn \l_stex_get_symbol_uri_str {
1867           ##1
1868         }
1869       }
1870     }
1871   }
1872 }
1873 \l_tmpa_tl
1874 }
1875 }
1876
1877 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
1878   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
1879   { \tl_tail:N \l_tmpa_tl }
1880   \tl_if_single:NTF \l_tmpa_tl {
1881     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
1882       \exp_after:wN \str_set:Nn \exp_after:wN
1883       \l_stex_get_symbol_uri_str \l_tmpa_tl
1884     }{
1885       % TODO
1886       % tail is not a single group
1887     }
1888   }{
1889     % TODO
1890     % tail is not a single group
1891   }
1892 }

```

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

20.2 Notations

```

1893 <@@=stex_notation>
1894 notation arguments:
1895 \keys_define:nn { stex / notation } {
1896   lang .tl_set_x:N = \l__stex_notation_lang_str ,
1897   variant .tl_set_x:N = \l__stex_notation_variant_str ,
1898   prec .tl_set_x:N = \l__stex_notation_prec_str ,
1899   op .tl_set:N = \l__stex_notation_op_tl ,
1900   unknown .code:n = \str_set:Nx
1901     \l__stex_notation_variant_str \l_keys_key_str
1902 }
1903
1904 \cs_new_protected:Nn \__stex_notation_args:n {
1905   \str_clear:N \l__stex_notation_lang_str
1906   \str_clear:N \l__stex_notation_variant_str

```

```

1906 \str_clear:N \l__stex_notation_prec_str
1907 \tl_clear:N \l__stex_notation_op_tl
1908
1909 \keys_set:nn { stex / notation } { #1 }
1910 }

```

\notation

```

1911 \NewDocumentCommand \notation { 0{ } m } {
1912   \__stex_notation_args:n { #1 }
1913   \tl_clear:N \l_stex_symdecl_definiens_tl
1914   \stex_get_symbol:n { #2 }
1915   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
1916 }
1917 \stex_deactivate_macro:Nn \notation {module~environments}

```

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

\stex_notation_do:nn

```

1918 \cs_new_protected:Nn \stex_notation_do:nn {
1919   \prop_set_eq:Nc \l_tmpa_prop {
1920     g_stex_symdecl_ #1 _prop
1921   }
1922
1923   \prop_clear:N \l_tmpb_prop
1924   \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
1925   \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
1926   \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
1927
1928   % precedences
1929   \seq_clear:N \l_tmpb_seq
1930   \exp_args:NNno
1931   \str_if_empty:NTF \l__stex_notation_prec_str {
1932     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1933     \int_compare:nNnTF \l_tmpa_str = 0 {
1934       \exp_args:NNnx
1935       \prop_put:Nno \l_tmpb_prop { opprec }
1936       { \neginfprec }
1937     }{
1938       \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
1939     }
1940   } {
1941     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
1942       \exp_args:NNnx
1943       \prop_put:Nno \l_tmpb_prop { opprec }
1944       { \neginfprec }
1945       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1946       \int_step_inline:nn { \l_tmpa_str } {
1947         \exp_args:NNx
1948         \seq_put_right:Nn \l_tmpb_seq { \infprec }
1949       }
1950     }{
1951       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
1952       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
1953         \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
1954         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {

```

```

1955         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
1956         \l_tmpa_seq {\tl_to_str:n{x} } { \l_tmpa_str }
1957         \seq_map_inline:Nn \l_tmpa_seq {
1958             \seq_put_right:Nn \l_tmpb_seq { ##1 }
1959         }
1960     }
1961     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1962 }{
1963     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1964     \int_compare:nNnTF \l_tmpa_str = 0 {
1965         \exp_args:NNnx
1966         \prop_put:Nno \l_tmpb_prop { opprec }
1967         { \infprec }
1968     }{
1969         \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
1970     }
1971 }
1972 }
1973 }
1974
1975 \seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
1976 \int_step_inline:nn { \l_tmpa_str } {
1977     \seq_pop_left:NnF \l_tmpa_seq \l_tmpb_str {
1978         \exp_args:NNx
1979         \seq_put_right:Nn \l_tmpb_seq {
1980             \prop_item:Nn \l_tmpb_prop { opprec }
1981         }
1982     }
1983 }
1984
1985 \prop_put:Nno \l_tmpb_prop { argprec } \l_tmpb_seq
1986 \tl_clear:N \l_tmpa_tl
1987
1988 \int_compare:nNnTF \l_tmpa_str = 0 {
1989     \exp_args:NNe
1990     \cs_set:Npn \l__stex_notation_macrocode_cs {
1991         \_stex_term_math_oms:nnnn { #1 }
1992         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1993         { \prop_item:Nn \l_tmpb_prop { opprec } }
1994         { \exp_not:n { #2 } }
1995     }
1996     \__stex_notation_final:
1997 }{
1998     \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
1999     \str_if_in:NnTF \l_tmpb_str b {
2000         \exp_args:Nne \use:nn
2001         {
2002             \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2003             \cs_set:Npn \l_tmpa_str { {
2004                 \_stex_term_math_omb:nnnn { #1 }
2005                 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2006                 { \prop_item:Nn \l_tmpb_prop { opprec } }
2007                 { \exp_not:n { #2 } }
2008             }}

```

```

2009   }{
2010     \str_if_in:NnTF \l_tmpb_str B {
2011       \exp_args:Nne \use:nn
2012       {
2013         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2014         \cs_set:Npn \l_tmpa_str } { {
2015           \stex_term_math_omb:nnnn { #1 }
2016           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2017           { \prop_item:Nn \l_tmpb_prop { opprec } }
2018           { \exp_not:n { #2 } }
2019         } }
2020       }{
2021         \exp_args:Nne \use:nn
2022         {
2023           \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2024           \cs_set:Npn \l_tmpa_str } { {
2025             \stex_term_math_oma:nnnn { #1 }
2026             { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2027             { \prop_item:Nn \l_tmpb_prop { opprec } }
2028             { \exp_not:n { #2 } }
2029           } }
2030         }
2031       }
2032
2033       \int_zero:N \l_tmpa_int
2034       \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2035       \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2036       \__stex_notation_arguments:
2037     }
2038   }

```

(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

```

2039 \cs_new_protected:Nn \__stex_notation_arguments: {
2040   \int_incr:N \l_tmpa_int
2041   \str_if_empty:NnTF \l_tmpa_str {
2042     \__stex_notation_final:
2043   }{
2044     \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2045     \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2046     \str_if_eq:VnTF \l_tmpb_str a {
2047       \__stex_notation_argument_assoc:n
2048     }{
2049       \str_if_eq:VnTF \l_tmpb_str B {
2050         \__stex_notation_argument_assoc:n
2051       }{
2052         \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2053         \tl_put_right:Nx \l_tmpa_tl {
2054           { \stex_term_math_arg:nnn
2055             { \int_use:N \l_tmpa_int }
2056             { \l_tmpb_str }
2057             { ####\int_use:N \l_tmpa_int }
2058           }

```

```

2059     }
2060     \__stex_notation_arguments:
2061   }
2062 }
2063 }
2064 }

```

(End definition for __stex_notation_arguments:.)

__stex_notation_argument_assoc:n

```

2065 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
2066   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2067   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
2068   \tl_put_right:Nx \l_tmpa_tl {
2069     { \stex_term_math_assoc_arg:nnnn
2070       { \int_use:N \l_tmpa_int }
2071       { \l_tmpb_str }
2072       \exp_args:No \exp_not:n
2073       {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
2074       { ####\int_use:N \l_tmpa_int }
2075     }
2076   }
2077   \__stex_notation_arguments:
2078 }

```

(End definition for __stex_notation_argument_assoc:n.)

__stex_notation_final: Called after processing all notation arguments

```

2079 \cs_new_protected:Nn \__stex_notation_final: {
2080   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
2081   \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
2082   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2083   \exp_args:Nne \use:nn
2084   {
2085     \cs_generate_from_arg_count:cNnn {
2086       stex_notation_ \l_tmpa_str \c_hash_str
2087       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2088       _cs
2089     }
2090     \cs_gset:Npn \l_tmpb_str } { {
2091       \exp_after:wN \exp_after:wN \exp_after:wN
2092       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2093       { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2094     } }
2095
2096   \tl_if_empty:NF \l__stex_notation_op_tl {
2097     \cs_gset:cpx {
2098       stex_op_notation_ \l_tmpa_str \c_hash_str
2099       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2100       _cs
2101     } {
2102       \stex_term_oms:nnn {
2103         \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2104         \l__stex_notation_lang_str

```

```

2105     }{
2106         \l_tmpa_str
2107     }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2108 }
2109 }
2110
2111
2112
2113 \stex_debug:nn{symbols}{
2114     Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2115     ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
2116     Operator~precedence:~
2117     \prop_item:Nn \l_tmpb_prop { opprec }^^J
2118     Argument~precedences:~
2119     \seq_use:Nn \l_tmpa_seq {,~}^^J
2120     Notation: \cs_meaning:c {
2121         stex_notation_ \l_tmpa_str \c_hash_str
2122         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2123         _cs
2124     }
2125 }
2126
2127 \prop_gset_eq:cN {
2128     g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2129     \c_hash_str \l__stex_notation_lang_str _prop
2130 } \l_tmpb_prop
2131
2132 \exp_args:Nx
2133 \stex_add_to_current_module:n {
2134     \prop_get:cnN {
2135         g_stex_symdecl_
2136         \prop_item:Nn \l_tmpb_prop { symbol }
2137         _prop
2138     } { notations } \exp_not:N \l_tmpa_seq
2139     \seq_put_right:Nn \exp_not:N \l_tmpa_seq {
2140         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2141     }
2142     \prop_put:cno {
2143         g_stex_symdecl_
2144         \prop_item:Nn \l_tmpb_prop { symbol }
2145         _prop
2146     } { notations } \exp_not:N \l_tmpa_seq
2147 }
2148
2149 \stex_if_smsmode:TF {
2150     \stex_smsmode_set_codes:
2151     \exp_args:Nx \stex_add_to_sms:n {
2152         \prop_gset_from_keyval:cn {
2153             g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2154             \c_hash_str \l__stex_notation_lang_str _prop
2155         } {
2156             symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2157             language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2158             variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,

```

```

2159         opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2160         argprec     = \prop_item:Nn \l_tmpb_prop { argprec }     ,
2161     }
2162 }
2163 }{
2164   \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
2165   \seq_put_right:Nx \l_tmpa_seq {
2166     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2167   }
2168   \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
2169   \prop_set_eq:cN {
2170     g_stex_symdecl_ \l_tmpa_str _prop
2171   } \l_tmpa_prop
2172
2173   % HTML annotations
2174   \stex_if_do_html:T {
2175     \stex_annotate_invisible:nnn { notation }
2176     { \prop_item:Nn \l_tmpb_prop { symbol } } {
2177       \stex_annotate_invisible:nnn { notationfragment }
2178       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
2179       \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2180       \stex_annotate_invisible:nnn { precedence }
2181       { \prop_item:Nn \l_tmpb_prop { opprec } ;
2182         \seq_use:Nn \l_tmpa_seq { x }
2183       }{}
2184
2185       \int_zero:N \l_tmpa_int
2186       \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2187       \tl_clear:N \l_tmpa_tl
2188       \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{}{
2189         \int_incr:N \l_tmpa_int
2190         \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2191         \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2192         \str_if_eq:VnTF \l_tmpb_str a {
2193           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2194             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2195             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2196           } }
2197         }{
2198           \str_if_eq:VnTF \l_tmpb_str B {
2199             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2200               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2201               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2202             } }
2203           }{
2204             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2205               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2206             } }
2207           }
2208         }
2209       }
2210       \stex_annotate_invisible:nnn { notationcomp }{}{
2211         $ \exp_args:Nno \use:nn { \use:c {
2212           stex_notation_ \prop_item:Nn \l_tmpb_prop { symbol }

```

```

2213         \c_hash_str \l__stex_notation_variant_str
2214         \c_hash_str \l__stex_notation_lang_str _cs
2215     } } { \l_tmpa_tl } $
2216   }
2217 }
2218 }
2219 }
2220 }

```

(End definition for _stex_notation_final:.)

\symdef

```

2221 \keys_define:nn { stex / symdef } {
2222   name .str_set_x:N = \l_stex_symdecl_name_str ,
2223   local .bool_set:N = \l_stex_symdecl_local_bool ,
2224   args .str_set_x:N = \l_stex_symdecl_args_str ,
2225   type .tl_set:N = \l_stex_symdecl_type_tl ,
2226   def .tl_set:N = \l_stex_symdecl_definiens_tl ,
2227   op .tl_set:N = \l__stex_notation_op_tl ,
2228   lang .str_set_x:N = \l__stex_notation_lang_str ,
2229   variant .str_set_x:N = \l__stex_notation_variant_str ,
2230   prec .str_set_x:N = \l__stex_notation_prec_str ,
2231   unknown .code:n = \str_set:Nx
2232     \l__stex_notation_variant_str \l_keys_key_str
2233 }
2234
2235 \cs_new_protected:Nn \_stex_notation_symdef_args:n {
2236   \str_clear:N \l_stex_symdecl_name_str
2237   \str_clear:N \l_stex_symdecl_args_str
2238   \bool_set_false:N \l_stex_symdecl_local_bool
2239   \tl_clear:N \l_stex_symdecl_type_tl
2240   \tl_clear:N \l_stex_symdecl_definiens_tl
2241   \str_clear:N \l__stex_notation_lang_str
2242   \str_clear:N \l__stex_notation_variant_str
2243   \str_clear:N \l__stex_notation_prec_str
2244   \tl_clear:N \l__stex_notation_op_tl
2245
2246   \keys_set:nn { stex / symdef } { #1 }
2247 }
2248
2249 \NewDocumentCommand \symdef { 0{} m } {
2250   \_stex_notation_symdef_args:n { #1 }
2251   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2252   \stex_symdecl_do:n { #2 }
2253   \exp_args:Nx \stex_notation_do:nn {
2254     \prop_item:Nn \l_tmpa_prop { module } ?
2255     \prop_item:Nn \l_tmpa_prop { name }
2256   }
2257 }
2258 \stex_deactivate_macro:Nn \symdef {module~environments}

```

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

```

2259 \endpackage

```


Chapter 21

STEX -Terms Implementation

```
2260 <*package>
2261
2262 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2263
2264 <@@=stex_terms>
2265
2266 Warnings and error messages
2267 \msg_new:nnn{stex}{error/nonotation}{
2268   Symbol~#1~invoked,~but~has~no~notation#2!
2269 }
2270 \msg_new:nnn{stex}{error/notationarg}{
2271   Error~in~parsing~notation~#1
2272 }
2273
```

21.1 Symbol Invocations

Arguments:

```
2272 \keys_define:nn { stex / terms } {
2273   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2274   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2275   unknown .code:n = \str_set:Nx
2276     \l__stex_terms_variant_str \l_keys_key_str
2277 }
2278
2279 \cs_new_protected:Nn \__stex_terms_args:n {
2280   \str_clear:N \l__stex_terms_lang_str
2281   \str_clear:N \l__stex_terms_variant_str
2282   \str_clear:N \l__stex_terms_prec_str
2283   \tl_clear:N \l__stex_terms_op_tl
2284
2285   \keys_set:nn { stex / terms } { #1 }
2286 }
```

`\stex_invoke_symbol:n` Invokes a semantic macro

```

2287 \cs_new_protected:Nn \stex_invoke_symbol:n {
2288   \if_mode_math:
2289     \exp_after:wN \__stex_terms_invoke_math:n
2290   \else:
2291     \exp_after:wN \__stex_terms_invoke_text:n
2292   \fi: { #1 }
2293 }

```

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

`__stex_terms_invoke_math:n`

```

2294 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2295   \peek_charcode_remove:NTF ! {
2296     \peek_charcode:NTF [ {
2297       \__stex_terms_invoke_op:nw { #1 }
2298     }{
2299       \__stex_terms_invoke_op:nw { #1 } []
2300     }
2301   }{
2302     \peek_charcode_remove:NTF * {
2303       \__stex_terms_invoke_text:n { #1 }
2304     }{
2305       \peek_charcode:NTF [ {
2306         \__stex_terms_invoke_math:nw { #1 }
2307       }{
2308         \__stex_terms_invoke_math:nw { #1 } []
2309       }
2310     }
2311   }
2312 }

```

(End definition for `__stex_terms_invoke_math:n`.)

`__stex_terms_invoke_op:nw`

```

2313 \cs_new_protected:Npn \__stex_terms_invoke_op:nw #1 [#2] {
2314   \__stex_terms_args:n { #2 }
2315   \cs_if_exist:cTF {
2316     stex_op_notation_ #1 \c_hash_str
2317     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str_cs
2318   }{
2319     \csname stex_op_notation_ #1 \c_hash_str
2320     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str_cs
2321     \endcsname
2322   }{
2323     % TODO throw error
2324   }
2325 }

```

(End definition for `__stex_terms_invoke_op:nw`.)

`__stex_terms_invoke_math:nw`

```

2326 \cs_new_protected:Npn \__stex_terms_invoke_math:nw #1 [#2] {
2327   \__stex_terms_args:n { #2 }
2328   \prop_set_eq:Nc \l_tmpa_prop {
2329     g_stex_symdecl_ #1 _prop

```

```

2330 }
2331 \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
2332 \seq_if_empty:NTF \l_tmpa_seq {
2333   \msg_error:nnnn{stex}{error/nonotation}{#1}{s}
2334 } {
2335   \seq_if_in:NxTF \l_tmpa_seq
2336   { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
2337     \use:c{
2338       stex_notation_ #1 \c_hash_str
2339       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2340       _cs
2341     }
2342   }{
2343     \str_if_empty:NTF \l__stex_terms_variant_str {
2344       \str_if_empty:NTF \l__stex_terms_lang_str {
2345         \seq_get_left:NN \l_tmpa_seq \l_tmpa_str
2346         \use:c{
2347           stex_notation_ #1 \c_hash_str \l_tmpa_str
2348           _cs
2349         }
2350       }{
2351         \msg_error:nn{stex}{error/nonotation}{#1}{
2352           ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2353         }
2354       }
2355     }{
2356       \msg_error:nn{stex}{error/nonotation}{#1}{
2357         ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2358       }
2359     }
2360   }
2361 }
2362 }

```

(End definition for `__stex_terms_invoke_math:nw`.)

`__stex_terms_invoke_text:n`

```

2363 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
2364   \peek_charcode_remove:NTF ! {
2365     \stex_term_custom:nn { #1 } { }
2366   }{
2367     \prop_set_eq:Nc \l_tmpa_prop {
2368       g_stex_symdecl_ #1 _prop
2369     }
2370     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2371     \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2372   }
2373 }

```

(End definition for `__stex_terms_invoke_text:n`.)

21.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
2374 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2375 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2376 \int_new:N \l__stex_terms_downprec
2377 \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
2378 \tl_set:Nn \l__stex_terms_left_bracket_str (
2379 \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

```

2380 \cs_new_protected:Nn \_stex_terms_maybe_brackets:nn {
2381 \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
2382 \bool_if:NTF \l_stex_inarray_bool { #2 }{
2383 \dobrackets { #2 }
2384 }
2385 }{ #2 }
2386 }

(End definition for \_stex_terms_maybe_brackets:nn.)

```

\dobrackets

```

2387 %\RequirePackage{scalerel}
2388 \cs_new_protected:Npn \dobrackets #1 {
2389 %\ThisStyle{\if D\m@switch
2390 % \exp_args:Nnx \use:nn
2391 % { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
2392 % { \exp_not:N\right\l__stex_terms_right_bracket_str }
2393 % \else
2394 \exp_args:Nnx \use:nn
2395 { \l__stex_terms_left_bracket_str #1 }
2396 { \l__stex_terms_right_bracket_str }
2397 %\fi}
2398 }

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

```

\withbrackets

```

2399 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2400 \exp_args:Nnx \use:nn
2401 {
2402 \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
2403 \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
2404 #3
2405 }
2406 {
2407 \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
2408 {\l__stex_terms_left_bracket_str}
2409 \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str

```

```

2410     {\l__stex_terms_right_bracket_str}
2411   }
2412 }

```

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

`\STEXinvisible`

```

2413 \cs_new_protected:Npn \STEXinvisible #1 {
2414   \stex_annotate_invisible:n { #1 }
2415 }

```

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

OMDoc terms:

`_stex_term_math_oms:nnnn`

```

2416 \cs_new_protected:Nn \_stex_term_oms:nnn {
2417   \stex_annotate:nnn{ OMID }{ #2 }{
2418     \stex_highlight_term:nn { #1 } { #3 }
2419   }
2420 }
2421
2422 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2423   \__stex_terms_maybe_brackets:nn { #3 }{
2424     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2425   }
2426 }

```

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

`_stex_term_math_oma:nnnn`

```

2427 \cs_new_protected:Nn \_stex_term_oma:nnn {
2428   \stex_annotate:nnn{ OMA }{ #2 }{
2429     \stex_highlight_term:nn { #1 } { #3 }
2430   }
2431 }
2432
2433 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2434   \__stex_terms_maybe_brackets:nn { #3 }{
2435     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2436   }
2437 }

```

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

`_stex_term_math_omb:nnnn`

```

2438 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2439   \stex_annotate:nnn{ OMBIND }{ #2 }{
2440     \stex_highlight_term:nn { #1 } { #3 }
2441   }
2442 }
2443
2444 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2445   \__stex_terms_maybe_brackets:nn { #3 }{
2446     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2447   }
2448 }

```

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

`_stex_term_math_arg:nnn`

```

2449 \cs_new_protected:Nn \_stex_term_arg:nn {
2450   \stex_unhighlight_term:n {
2451     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2452   }
2453 }
2454 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
2455   \exp_args:Nnx \use:nn
2456     { \int_set:Nn \l__stex_terms_downprec { #2 }
2457       \_stex_term_arg:nn { #1 }{ #3 }
2458     }
2459   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2460 }

```

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

`_stex_term_math_assoc_arg:nnnn`

```

2461 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
2462   \seq_set_split:Nnn \l_tmpa_seq , { #4 }
2463   \int_compare:nNnTF { \seq_count:N \l_tmpa_seq } < 2 {
2464     \tl_set:Nn \l_tmpa_tl { #4 }
2465   }{
2466     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
2467     \seq_reverse:N \l_tmpa_seq
2468     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_tl
2469     \tl_set:No \l_tmpa_tl { \l_tmpb_tl }
2470
2471     \seq_map_inline:Nn \l_tmpa_seq {
2472       \exp_args:Nno \tl_set:No \l_tmpa_tl {
2473         \exp_args:Nno
2474           \l_tmpa_cs { ##1 } \l_tmpa_tl
2475       }
2476     }
2477
2478   }
2479   \exp_args:Nnno
2480   \_stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2481 }

```

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

`\stex_term_custom:nn`

```

2482 \cs_new_protected:Nn \stex_term_custom:nn {
2483   \str_set:Nn \l__stex_terms_custom_uri { #1 }
2484   \str_set:Nn \l_tmpa_str { #2 }
2485   \tl_clear:N \l_tmpa_tl
2486   \int_zero:N \l_tmpa_int
2487   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2488   \__stex_terms_custom_loop:
2489 }

```

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

_stex_terms_custom_loop:

```

2490 \cs_new_protected:Nn \_stex_terms_custom_loop: {
2491   \bool_set_false:N \l_tmpa_bool
2492   \bool_while_do:nn {
2493     \str_if_eq_p:ee X {
2494       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2495     }
2496   }{
2497     \int_incr:N \l_tmpa_int
2498   }
2499
2500   \peek_charcode:NTF [ {
2501     % notation/text component
2502     \_stex_terms_custom_component:w
2503   } {
2504     \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2505       % all arguments read => finish
2506       \_stex_terms_custom_final:
2507     } {
2508       % arguments missing
2509       \peek_charcode_remove:NTF * {
2510         % invisible, specific argument position or both
2511         \peek_charcode:NTF [ {
2512           % visible specific argument position
2513           \_stex_terms_custom_arg:wn
2514         } {
2515           % invisible
2516           \peek_charcode_remove:NTF * {
2517             % invisible specific argument position
2518             \_stex_terms_custom_arg_inv:wn
2519           } {
2520             % invisible next argument
2521             \_stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2522           }
2523         } {
2524           % next normal argument
2525           \_stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
2526         }
2527       }
2528     }
2529   }
2530 }

```

(End definition for _stex_terms_custom_loop:.)

_stex_terms_custom_arg_inv:wn

```

2531 \cs_new_protected:Npn \_stex_terms_custom_arg_inv:wn [ #1 ] #2 {
2532   \bool_set_true:N \l_tmpa_bool
2533   \_stex_terms_custom_arg:wn [ #1 ] { #2 }
2534 }

```

(End definition for _stex_terms_custom_arg_inv:wn.)

_stex_terms_custom_arg:wn

```

2535 \cs_new_protected:Npn \__stex_terms_custom_arg:wn [ #1 ] #2 {
2536   \str_set:Nx \l_tmpb_str {
2537     \str_item:Nn \l_tmpa_str { #1 }
2538   }
2539   \str_case:VnTF \l_tmpb_str {
2540     { X } {
2541       \msg_error:nnn{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2542     }
2543     { i } { \__stex_terms_custom_set_X:n { #1 } }
2544     { b } { \__stex_terms_custom_set_X:n { #1 } }
2545     { a } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
2546     { B } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
2547   }{}{
2548     \msg_error:nnn{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2549   }
2550
2551   \bool_if:nTF \l_tmpa_bool {
2552     \tl_put_right:Nx \l_tmpa_tl {
2553       \stex_annotate_invisible:n {
2554         \stex_term_arg:nn { \int_eval:n { #1 } }
2555         \exp_not:n { { #2 } }
2556       }
2557     }
2558   } {
2559     \tl_put_right:Nx \l_tmpa_tl {
2560       \stex_term_arg:nn { \int_eval:n { #1 } }
2561       \exp_not:n { { #2 } }
2562     }
2563   }
2564
2565   \__stex_terms_custom_loop:
2566 }

```

(End definition for __stex_terms_custom_arg:wn.)

__stex_terms_custom_set_X:n

```

2567 \cs_new_protected:Nn \__stex_terms_custom_set_X:n {
2568   \str_set:Nx \l_tmpa_str {
2569     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
2570     X
2571     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
2572   }
2573 }

```

(End definition for __stex_terms_custom_set_X:n.)

__stex_terms_custom_component:

```

2574 \cs_new_protected:Npn \__stex_terms_custom_component:w [ #1 ] {
2575   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
2576   \__stex_terms_custom_loop:
2577 }

```

(End definition for __stex_terms_custom_component:.)

`_stex_terms_custom_final:`

```

2578 \cs_new_protected:Nn \_stex_terms_custom_final: {
2579   \int_compare:nNnTF \l_tmpb_int = 0 {
2580     \exp_args:Nnno \_stex_term_oms:nnn
2581   }{
2582     \str_if_in:NnTF \l_tmpa_str {b} {
2583       \exp_args:Nnno \_stex_term_ombind:nnn
2584     } {
2585       \exp_args:Nnno \_stex_term_oma:nnn
2586     }
2587   }
2588   { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
2589 }

```

(End definition for _stex_terms_custom_final:.)

\symref

\symname

```

2590 \NewDocumentCommand \symref { m m }{
2591   \let\compemph_uri_prev:\compemph@uri
2592   \let\compemph@uri\symrefemph@uri
2593   \STEXsymbol{#1}![#2]
2594   \let\compemph@uri\compemph_uri_prev:
2595 }
2596
2597 \keys_define:nn { stex / symname } {
2598   post      .str_set_x:N      = \l_stex_symname_post_str
2599 }
2600
2601 \cs_new_protected:Nn \stex_symname_args:n {
2602   \str_clear:N \l_stex_symname_post_str
2603   \keys_set:nn { stex / symname } { #1 }
2604 }
2605
2606 \NewDocumentCommand \symname { 0{} m }{
2607   \stex_symname_args:n { #1 }
2608   \stex_get_symbol:n { #2 }
2609   \str_set:Nx \l_tmpa_str {
2610     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
2611   }
2612   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
2613
2614   \let\compemph_uri_prev:\compemph@uri
2615   \let\compemph@uri\symrefemph@uri
2616   \exp_args:NNx \use:nn
2617   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }![
2618     \l_tmpa_str \l_stex_symname_post_str
2619   ] }
2620   \let\compemph@uri\compemph_uri_prev:
2621 }

```

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

21.3 Notation Components

2622 <@@=stex_notationcomps>

`\stex_highlight_term:nn`

```

2623
2624 \str_new:N \l__stex_notationcomps_highlight_uri_str
2625 \cs_new_protected:Nn \stex_highlight_term:nn {
2626   \exp_args:Nnx
2627   \use:nn {
2628     \str_set:Nx \l__stex_notationcomps_highlight_uri_str { #1 }
2629     #2
2630   } {
2631     \str_set:Nx \exp_not:N \l__stex_notationcomps_highlight_uri_str
2632     { \l__stex_notationcomps_highlight_uri_str }
2633   }
2634 }
2635
2636 \cs_new_protected:Nn \stex_unhighlight_term:n {
2637   % \latexml_if:TF {
2638   %   #1
2639   % } {
2640   %   \scalatex_if:TF {
2641   %     #1
2642   %   } {
2643     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
2644   % }
2645 % }
2646 }
```

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

```

\comp
\compemph@uri 2647 \cs_new_protected:Npn \comp #1 {
\compemph 2648   \str_if_empty:NF \l__stex_notationcomps_highlight_uri_str {
\defemph 2649     \scalatex_if:TF {
\defemph@uri 2650       \stex_annotate:nnn { comp }{ \l__stex_notationcomps_highlight_uri_str }{ #1 }
\symrefemph 2651     }{
\symrefemph@uri 2652       \exp_args:Nnx \compemph@uri { #1 } { \l__stex_notationcomps_highlight_uri_str }
2653     }
2654   }
2655 }
2656
2657 \cs_new_protected:Npn \compemph@uri #1 #2 {
2658   \compemph{ #1 }
2659 }
2660
2661
2662 \cs_new_protected:Npn \compemph #1 {
2663   \textcolor{blue}{#1}
2664 }
2665
2666 \cs_new_protected:Npn \defemph@uri #1 #2 {
2667   \defemph{#1}
2668 }
```

```

2669
2670 \cs_new_protected:Npn \defemph #1 {
2671   \textbf{#1}
2672 }
2673
2674 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
2675   \symrefemph{#1}
2676 }
2677
2678 \cs_new_protected:Npn \symrefemph #1 {
2679   \textbf{#1}
2680 }

```

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

`\ellipses`

```

2681 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix
\parrayline
\parraylineh
\parraycell
2682 \bool_new:N \l_stex_inarray_bool
2683 \bool_set_false:N \l_stex_inarray_bool
2684 \NewDocumentCommand \parray { m m } {
2685   \begingroup
2686   \bool_set_true:N \l_stex_inarray_bool
2687   \begin{array}{#1}
2688     #2
2689   \end{array}
2690   \endgroup
2691 }
2692
2693 \NewDocumentCommand \prmatrix { m } {
2694   \begingroup
2695   \bool_set_true:N \l_stex_inarray_bool
2696   \begin{matrix}
2697     #1
2698   \end{matrix}
2699   \endgroup
2700 }
2701
2702 \def \parrayline #1 #2 {
2703   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
2704 }
2705
2706 \def \parraylineh #1 #2 {
2707   #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
2708 }
2709
2710 \def \parraycell #1 {
2711   #1 \bool_if:NT \l_stex_inarray_bool {\&}
2712 }

```

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

```

2713 \endpackage

```

Chapter 22

STEX -Structural Features Implementation

```
2714 <*package>
2715
2716 %%%%%%%%%%% features.dtx %%%%%%%%%%%
2717
2718 <@@=stex_features>
      Warnings and error messages
2719
```

22.1 The feature environment

structural@feature

```
2720
2721 \NewDocumentEnvironment{structural@feature}{ m m m }{
2722   \stex_if_in_module:F {
2723     \msg_set:nnn{stex}{error/nomodule}{
2724       Structural~Feature~has~to~occur~in~a~module:\\
2725       Feature~#2~of~type~#1\\
2726       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
2727     }
2728     \msg_error:nn{stex}{error/nomodule}
2729   }
2730
2731   \str_set:Nx \l_stex_module_name_str {
2732     \prop_item:Nn \l_stex_current_module_prop
2733       { name } / #2 - feature
2734   }
2735
2736   \str_set:Nx \l_stex_module_ns_str {
2737     \prop_item:Nn \l_stex_current_module_prop
2738       { ns }
2739   }
2740
```

```

2741
2742 \str_clear:N \l_tmpa_str
2743 \seq_clear:N \l_tmpa_seq
2744 \tl_clear:N \l_tmpa_tl
2745 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
2746   origname = #2,
2747   name     = \l_stex_module_name_str ,
2748   ns       = \l_stex_module_ns_str ,
2749   imports  = \exp_not:o { \l_tmpa_seq } ,
2750   constants = \exp_not:o { \l_tmpa_seq } ,
2751   content  = \exp_not:o { \l_tmpa_tl } ,
2752   file     = \exp_not:o { \g_stex_currentfile_seq } ,
2753   lang     = \l_stex_module_lang_str ,
2754   sig      = \l_tmpa_str ,
2755   meta     = \l_tmpa_str ,
2756   feature  = #1 ,
2757 }
2758
2759 \stex_if_smsmode:TF {
2760   \stex_smsmode_set_codes:
2761 } {
2762   \begin{stex_annotate_env}{ feature:#1 }{}
2763   \stex_annotate_invisible:nnn{header}{}{ #3 }
2764 }
2765 }{
2766   \str_set:Nx \l_tmpa_str {
2767     c_stex_feature_
2768     \prop_item:Nn \l_stex_current_module_prop { ns } ?
2769     \prop_item:Nn \l_stex_current_module_prop { name }
2770     _prop
2771   }
2772   \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
2773   \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
2774   \stex_if_smsmode:TF {
2775     \exp_args:Nx \stex_add_to_sms:n {
2776       \prop_gset_from_keyval:cn {
2777         c_stex_feature_
2778         \prop_item:Nn \l_stex_current_module_prop { ns } ?
2779         \prop_item:Nn \l_stex_current_module_prop { name }
2780         _prop
2781       } {
2782         origname = #2,
2783         name     = \prop_item:cn { \l_tmpa_str } { name } ,
2784         ns       = \prop_item:cn { \l_tmpa_str } { ns } ,
2785         imports  = \prop_item:cn { \l_tmpa_str } { imports } ,
2786         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
2787         content  = \prop_item:cn { \l_tmpa_str } { content } ,
2788         file     = \prop_item:cn { \l_tmpa_str } { file } ,
2789         lang     = \prop_item:cn { \l_tmpa_str } { lang } ,
2790         sig      = \prop_item:cn { \l_tmpa_str } { sig } ,
2791         meta     = \prop_item:cn { \l_tmpa_str } { meta } ,
2792         feature  = \prop_item:cn { \l_tmpa_str } { feature }
2793       }
2794     }

```

```

2795 } {
2796     \end{stex_annotate_env}
2797 }
2798 }
2799

```

22.2 Features

structure

```

2800
2801 \prop_new:N \l_stex_all_structures_prop
2802
2803 \keys_define:nn { stex / features / structure } {
2804     name .str_set_x:N = \l__stex_features_structure_name_str ,
2805 }
2806
2807 \cs_new_protected:Nn \__stex_features_structure_args:n {
2808     \str_clear:N \l__stex_features_structure_name_str
2809     \keys_set:nn { stex / features / structure } { #1 }
2810 }
2811
2812 %\stex_new_feature:nnnn { structure } { 0{ } m } {
2813 % \__stex_features_structure_args:n { ##1 }
2814 % \str_if_empty:NT \l__stex_features_structure_name_str {
2815 %     \str_set:Nx \l__stex_features_structure_name_str { ##2 }
2816 % }
2817 %} {
2818 %
2819 %}
2820
2821 \NewDocumentEnvironment{mathstructure}{ 0{ } m }{
2822     \__stex_features_structure_args:n { #1 }
2823     \str_if_empty:NT \l__stex_features_structure_name_str {
2824         \str_set:Nx \l__stex_features_structure_name_str { #2 }
2825     }
2826     \exp_args:Nnnx
2827     \begin{structural@feature}{ structure }
2828         { \l__stex_features_structure_name_str }{}
2829         \seq_clear:N \l_tmpa_seq
2830         \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
2831     }{
2832     }{
2833         \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
2834         \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
2835         \str_set:Nx \l_tmpa_str {
2836             \prop_item:Nn \l_stex_current_module_prop { ns } ?
2837             \prop_item:Nn \l_stex_current_module_prop { name }
2838         }
2839         \seq_map_inline:Nn \l_tmpa_seq {
2840             \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
2841         }
2842         \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
2843         \exp_args:Nnx

```

```

2844 \AddToHookNext { env / mathstructure / after }{
2845 \syndec1[type = \exp_not:N\collection,def={\STEXsymbol{module-type}{
2846 \_stex_term_math_oms:nnnn { \l_tmpa_str }{}{0}{}}
2847 }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
2848 \STEXexport {
2849 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2850 {\prop_item:Nn \l_stex_current_module_prop { origname }}
2851 {\l_tmpa_str}
2852 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2853 {#2}{\l_tmpa_str}
2854 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
2855 % \prop_item:Nn \l_stex_current_module_prop { origname },
2856 % \l_tmpa_str
2857 % }
2858 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
2859 % #2,\l_tmpa_str
2860 % }
2861 % \tl_set:cx { #2 } {
2862 % \stex_invoke_structure:n { \l_tmpa_str }
2863 }
2864 }
2865
2866 \end{structural@feature}
2867 % \g_stex_last_feature_prop
2868 }

```

\instantiate

```

2869 \seq_new:N \l__stex_features_structure_field_seq
2870 \str_new:N \l__stex_features_structure_field_str
2871 \str_new:N \l__stex_features_structure_def_tl
2872 \prop_new:N \l__stex_features_structure_prop
2873 \NewDocumentCommand \instantiate { m O{} m }{
2874 \stex_smsmode_set_codes:
2875 \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
2876 \prop_set_eq:Nc \l__stex_features_structure_prop {
2877 c_stex_feature_\l_tmpa_str _prop
2878 }
2879 \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
2880 \seq_map_inline:Nn \l__stex_features_structure_field_seq {
2881 \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
2882 \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
2883 \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
2884 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
2885 {!} \l_tmpa_tl
2886 \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
2887 \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
2888 \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
2889 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
2890 }{
2891 \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
2892 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
2893 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
2894 \l_tmpa_tl
2895 \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {

```

```

2896         \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
2897         \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
2898     }{
2899         \tl_clear:N \l_tmpb_tl
2900     }
2901 }
2902 }{
2903     \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
2904     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
2905         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
2906         \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
2907         \tl_clear:N \l_tmpa_tl
2908     }{
2909         % TODO throw error
2910     }
2911 }
2912 % \l_tmpa_str: name
2913 % \l_tmpa_tl: definiens
2914 % \l_tmpb_tl: notation
2915 \tl_if_empty:NT \l__stex_features_structure_field_str {
2916     % TODO throw error
2917 }
2918 \str_clear:N \l_tmpb_str
2919
2920 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
2921 \seq_map_inline:Nn \l_tmpa_seq {
2922     \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
2923     \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
2924     \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
2925         \seq_map_break:n {
2926             \str_set:Nn \l_tmpb_str { ####1 }
2927         }
2928     }
2929 }
2930 \prop_get:cnN { g_stex_symdecl_ \l_tmpb_str _prop } {args}
2931     \l_tmpb_str
2932
2933 \tl_if_empty:NTF \l_tmpb_tl {
2934     \tl_if_empty:NF \l_tmpa_tl {
2935         \exp_args:Nx \use:n {
2936             \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
2937         }
2938     }
2939 }{
2940     \tl_if_empty:NTF \l_tmpa_tl {
2941         \exp_args:Nx \use:n {
2942             \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\
2943         }
2944     }
2945 }{
2946     \exp_args:Nx \use:n {
2947         \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
2948         \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
2949     }

```



```

2950     }
2951   }
2952   % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
2953   % \prop_item:Nn \l_stex_current_module_prop {name} ?
2954   % #3/\l_stex_features_structure_field_str
2955   % \par
2956   % \expandafter\present\csname
2957   %   g_stex_symdecl_
2958   %   \prop_item:Nn \l_stex_current_module_prop {ns} ?
2959   %   \prop_item:Nn \l_stex_current_module_prop {name} ?
2960   %   #3/\l_stex_features_structure_field_str
2961   %   _prop
2962   % \endcsname
2963 }
2964
2965 \tl_clear:N \l__stex_features_structure_def_tl
2966
2967 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
2968 \seq_map_inline:Nn \l_tmpa_seq {
2969   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
2970   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
2971   \exp_args:Nx \use:n {
2972     \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
2973
2974     }
2975   }
2976
2977   \prop_if_exist:cF {
2978     g_stex_symdecl_
2979     \prop_item:Nn \l_stex_current_module_prop {ns} ?
2980     \prop_item:Nn \l_stex_current_module_prop {name} ?
2981     #3/\l_tmpa_str
2982     _prop
2983   }{
2984     \prop_get:cnN { g_stex_symdecl_ ##1 _prop } {args}
2985     \l_tmpb_str
2986     \exp_args:Nx \use:n {
2987       \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
2988     }
2989   }
2990 }
2991
2992 \symdecl*[type={\STEXsymbol{module-type}}{
2993   \_stex_term_math_oms:nnnn {
2994     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
2995     \prop_item:Nn \l__stex_features_structure_prop {name}
2996     }{}{0}{}
2997   }{}{#3}
2998
2999 % TODO: -> sms file
3000
3001 \tl_set:cx{ #3 }{
3002   \stex_invoke_structure:nnn {
3003     \prop_item:Nn \l_stex_current_module_prop {ns} ?

```

```

3004     \prop_item:Nn \l_stex_current_module_prop {name} ? #3
3005   } {
3006     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3007     \prop_item:Nn \l__stex_features_structure_prop {name}
3008   }
3009 }
3010
3011 }

```

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

\stex_invoke_structure:nnn

```

3012 % #1: URI of the instance
3013 % #2: URI of the instantiated module
3014 \cs_new_protected:Nn \stex_invoke_structure:nnn {
3015   \tl_if_empty:nTF{ #3 }{
3016     \prop_set_eq:Nc \l__stex_features_structure_prop {
3017       c_stex_feature_ #2 _prop
3018     }
3019     \tl_clear:N \l_tmpa_tl
3020     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3021     \seq_map_inline:Nn \l_tmpa_seq {
3022       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3023       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3024       \cs_if_exist:cT {
3025         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
3026       }{
3027         \tl_if_empty:NF \l_tmpa_tl {
3028           \tl_put_right:Nn \l_tmpa_tl {,}
3029         }
3030         \tl_put_right:Nx \l_tmpa_tl {
3031           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
3032         }
3033       }
3034     }
3035     \exp_args:No \mathstrut \l_tmpa_tl
3036   }{
3037     \stex_invoke_symbol:n{#1/#3}
3038   }
3039 }

```

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

```

3040 </package>

```

Chapter 23

STEX -Statements Implementation

```
3041 <*package>
3042
3043 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3044
3045 <@@=stex_statements>
3046
3047 Warnings and error messages
3048
3049
3050 symboldoc
3051
3052 \NewDocumentEnvironment{symboldoc}{m}{
3053   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3054   \seq_clear:N \l_tmpb_seq
3055   \seq_map_inline:Nn \l_tmpa_seq {
3056     \stex_get_symbol:n { ##1 }
3057     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3058       \l_stex_get_symbol_uri_str
3059     }
3060   }
3061   \par
3062   \exp_args:Nnnx
3063   \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
3064 }{
3065   \end{stex_annotate_env}
3066 }
3067
3068 \seq_new:N \g_stex_statements_patched_seq
3069
3070 \cs_new_protected:Nn \stex_statements_set_patched:n {
3071   \seq_put_right:Nn \g_stex_statements_patched_seq {#1}
3072 }
3073
3074 \cs_new_protected:Nn \stex_statements_patch:nn {
3075   \seq_if_in:NnF \g_stex_statements_patched_seq {#1} {
3076     \AddToHook{begindocument}{
3077       \cs_if_exist:cTF{end#1}{

```

```

3072     \AddToHook{env/#1/before}[stex]{\use:c{__stex_statements_#2_begin:n}{}}
3073     \AddToHook{env/#1/after}[stex]{\use:c{__stex_statements_#2_end:}}
3074   }{
3075     \NewDocumentEnvironment{#1}{0{}}{
3076       \use:c{__stex_statements_#2_begin:n}{}
3077     }{
3078       \use:c{__stex_statements_#2_end:}
3079     }
3080   }
3081 }
3082 }
3083 }

```

23.1 Definitions

definition

```

3084
3085 \NewDocumentCommand \definiendum { 0{ } m m } {
3086   \stex_get_symbol:n { #2 }
3087   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3088   \scalatex_if:TF {
3089     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { #3 }
3090   } {
3091     \exp_args:Nnx \defemph@uri { #3 } { \l_stex_get_symbol_uri_str }
3092   }
3093 }
3094 \stex_deactivate_macro:Nn \definiendum {definition~environments}
3095 \NewDocumentCommand \definame { 0{ } m } {
3096   % TODO: root
3097   \stex_get_symbol:n { #2 }
3098   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3099   \str_set:Nx \l_tmpa_str {
3100     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3101   }
3102   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {-}
3103   \scalatex_if:TF {
3104     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3105       \l_tmpa_str
3106     }
3107   } {
3108     \defemph@uri {
3109       \l_tmpa_str
3110     } { \l_stex_get_symbol_uri_str }
3111   }
3112 }
3113 \stex_deactivate_macro:Nn \definame {definition~environments}
3114
3115 \cs_new_protected:Nn \__stex_statements_defi_begin:n {
3116   \stex_reactivate_macro:N \definiendum
3117   \stex_reactivate_macro:N \definame
3118   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3119   \seq_clear:N \l_tmpb_seq
3120   \seq_map_inline:Nn \l_tmpa_seq {

```

```

3121 \stex_get_symbol:n { ##1 }
3122 \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3123   \l_stex_get_symbol_uri_str
3124 }
3125 }
3126 \stex_smsmode_set_codes:
3127 \exp_args:Nnnx
3128 \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
3129 }
3130
3131 \cs_new_protected:Nn \__stex_statements_defi_end: {
3132   \end{stex_annotate_env}
3133 }

Hook:
3134 \stex_statements_patch:nn{definition}{defi}

```

23.2 Assertions

assertion

```

3135 \cs_new_protected:Nn \__stex_statements_assertion_begin:n {
3136   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3137   \seq_clear:N \l_tmpb_seq
3138   \seq_map_inline:Nn \l_tmpa_seq {
3139     \stex_get_symbol:n { ##1 }
3140     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3141       \l_stex_get_symbol_uri_str
3142     }
3143   }
3144   \stex_smsmode_set_codes:
3145   \exp_args:Nnnx
3146   \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
3147 }
3148
3149 \cs_new_protected:Nn \__stex_statements_assertion_end: {
3150   \end{stex_annotate_env}
3151 }

Hook:
3152 \stex_statements_patch:nn{assertion}{assertion}

```

theorem

```

3153 \cs_new_protected:Nn \__stex_statements_theorem_begin:n {
3154   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3155   \seq_clear:N \l_tmpb_seq
3156   \seq_map_inline:Nn \l_tmpa_seq {
3157     \stex_get_symbol:n { ##1 }
3158     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3159       \l_stex_get_symbol_uri_str
3160     }
3161   }
3162   \stex_smsmode_set_codes:

```

```

3163 \exp_args:Nnnx
3164 \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
3165 }
3166
3167 \cs_new_protected:Nn \__stex_statements_theorem_end: {
3168 \end{stex_annotate_env}
3169 }

```

Hook:

```

3170 \stex_statements_patch:nn{theorem}{theorem}

```

lemma

```

3171 \cs_new_protected:Nn \__stex_statements_lemma_begin:n {
3172 \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3173 \seq_clear:N \l_tmpb_seq
3174 \seq_map_inline:Nn \l_tmpa_seq {
3175 \stex_get_symbol:n { ##1 }
3176 \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3177 \l_stex_get_symbol_uri_str
3178 }
3179 }
3180 \stex_smsmode_set_codes:
3181 \exp_args:Nnnx
3182 \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
3183 }
3184
3185 \cs_new_protected:Nn \__stex_statements_lemma_end: {
3186 \end{stex_annotate_env}
3187 }

```

Hook:

```

3188 \stex_statements_patch:nn{lemma}{lemma}

```

axiom

```

3189 \cs_new_protected:Nn \__stex_statements_axiom_begin:n {
3190 \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3191 \seq_clear:N \l_tmpb_seq
3192 \seq_map_inline:Nn \l_tmpa_seq {
3193 \stex_get_symbol:n { ##1 }
3194 \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3195 \l_stex_get_symbol_uri_str
3196 }
3197 }
3198 \stex_smsmode_set_codes:
3199 \exp_args:Nnnx
3200 \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
3201 }
3202
3203 \cs_new_protected:Nn \__stex_statements_axiom_end: {
3204 \end{stex_annotate_env}
3205 }

```

Hook:

```

3206 \stex_statements_patch:nn{axiom}{axiom}

```

23.3 Examples

example

```
3207 \cs_new_protected:Nn \__stex_statements_example_begin:n {  
3208   \seq_set_split:Nnn \l_tmpa_seq , { #1 }  
3209   \seq_clear:N \l_tmpb_seq  
3210   \seq_map_inline:Nn \l_tmpa_seq {  
3211     \stex_get_symbol:n { ##1 }  
3212     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {  
3213       \l_stex_get_symbol_uri_str  
3214     }  
3215   }  
3216   \stex_smsmode_set_codes:  
3217   \exp_args:Nnnx  
3218   \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpb_seq {,}}  
3219 }  
3220  
3221 \cs_new_protected:Nn \__stex_statements_example_end: {  
3222   \end{stex_annotate_env}  
3223 }
```

Hook:

```
3224 \stex_statements_patch:nn{example}{example}  
3225 \</package>
```

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

Chapter 24

STEX -Others Implementation

```
3226 <*package>
3227
3228 %%%%%%%%%% others.dtx %%%%%%%%%%
3229
3230 <@@=stex_others>
    Warnings and error messages
3231 % None

\MSC Math subject classifier

3232 \NewDocumentCommand \MSC {m} {
3233 % TODO
3234 }

(End definition for \MSC. This function is documented on page 10.)
    Patching tikzinput, if loaded
3235 \@ifpackageloaded{tikzinput}{
3236 \RequirePackage{stex-tikzinput}
3237 }{}
3238 </package>
```


Chapter 25

STEX -Metatheory Implementation

```
3239 \*package>
3240 \@@=stex_modules>
3241
3242 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
3243
3244 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
3245 \begingroup
3246 \stex_module_setup:nn{
3247   ns=\c_stex_metatheory_ns_str,
3248   meta=NONE
3249 }{Metatheory}
3250 \stex_reactivate_macro:N \symdecl
3251 \stex_reactivate_macro:N \notation
3252 \stex_reactivate_macro:N \symdef
3253 \ExplSyntaxOff
3254 \csname stex_suppress_html:n\endcsname{
3255   % is-a (a:A, a \in A, a is an A, etc.)
3256   \symdecl[args=ai]{isa}
3257   \notation[typed]{isa}{#1 \comp{:} #2}{#1 \comp, #2}
3258   \notation[in]{isa}{#1 \comp\in #2}{#1 \comp, #2}
3259   \notation[pred]{isa}{#2\comp(#1 \comp)}{#1 \comp, #2}
3260
3261   % bind (\forall, \Pi, \lambda etc.)
3262   \symdecl[args=Bi]{bind}
3263   \notation[forall]{bind}{\comp\forall #1.\;#2}{#1 \comp, #2}
3264   \notation[\Pi]{bind}{\comp\prod_{#1}#2}{#1 \comp, #2}
3265   \notation[deffun]{bind}{\comp( #1 \comp)\; \to\;}{#1 \comp, #2}
3266
3267   % dummy variable
3268   \symdecl{dummyvar}
3269   \notation[underscore]{dummyvar}{\comp\_}
3270   \notation[dot]{dummyvar}{\comp\cdot}
3271   \notation[dash]{dummyvar}{\comp{\rm --}}
3272
3273   %fromto (function space, Hom-set, implication etc.)
```

```

3274 \symdecl[args=ai]{fromto}
3275 \notation[xarrow]{fromto}{#1 \comp\to #2}{#1 \comp\times #2}
3276 \notation[arrow]{fromto}{#1 \comp\to #2}{#1 \comp\to #2}
3277
3278 % mapto (lambda etc.)
3279 %\symdecl[args=Bi]{mapto}
3280 %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
3281 %\notation[lambda]{mapto}{\comp\lambda #1 \comp. \; #2}{#1 \comp, #2}
3282 %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp. \; #2}{#1 \comp, #2}
3283
3284 % function/operator application
3285 \symdecl[args=ia]{apply}
3286 \notation[prec=0;0x\neginfprec,parens]{apply}{#1 \comp( #2 \comp)}{#1 \comp, #2}
3287 \notation[prec=0;0x\neginfprec,lambda]{apply}{#1 \; #2 }{#1 \; #2}
3288
3289 % ‘type’ of all collections (sets, classes, types, kinds)
3290 \symdecl{collection}
3291 \notation[U]{collection}{\comp{\mathcal{U}}{}}
3292 \notation[set]{collection}{\comp{\textsf{Set}}{}}
3293
3294 % sequences
3295 \symdecl[args=1]{seqtype}
3296 \notation[kleene]{seqtype}{#1^{\comp\ast}}
3297
3298 \symdef[args=2,li]{sequence-index}{#1_{#2}}
3299 \notation[ui]{sequence-index}{#1^{\#2}}
3300
3301 %\symdef[args=3,li]{sequence-from-to}{#1_{#2}\comp{\,\ellipses,}#1_{#3}}
3302 %\notation[ui]{sequence-from-to}{#1^{\#2}\comp{\,\ellipses,}#1^{\#3}}
3303 % ^ superceded by \aseqfromto and \livar/\uivar
3304
3305 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{\,\ellipses}}{#1\comp,#2}
3306 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{\,\ellipses\comp,}#2 }{#1\comp,#2}
3307 \symdef[args=aui,prec=nobrackets]{aseqfromtovia}{#1\comp{\,\ellipses\comp,}#2\comp{\,\ellipses\comp,}#3}{#1\comp,#2}
3308
3309 % letin (‘let’, local definitions, variable substitution)
3310 \symdecl[args=bii]{letin}
3311 \notation[let]{letin}{\comp{\rm let}}{\rm let};#1\comp{=}#2\; \comp{\rm in}}{\rm in};#3}
3312 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
3313 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
3314
3315 % structures
3316 \symdecl*[args=1]{module-type}
3317 \notation{module-type}{\mathtt{MOD} #1}
3318 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
3319 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{#1 \comp, #2}
3320
3321 }
3322 \ExplSyntaxOn
3323 \stex_add_to_current_module:n{
3324   \let\nappa\apply
3325   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
3326   \def\livar{\csname sequence-index\endcsname[li]}
3327   \def\uivar{\csname sequence-index\endcsname[ui]}

```

```

3328     \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
3329     \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
3330   }
3331   \__stex_modules_end_module:
3332 \endgroup
3333 </package>

```

Chapter 26

Tikzinput Implementation

```
3334 <*package>
3335
3336 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
3337
3338 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
3339 \RequirePackage{l3keys2e}
3340
3341 \keys_define:nn { tikzinput } {
3342   image .bool_set:N = \c_tikzinput_image_bool,
3343   image .default:n = false ,
3344 }
3345
3346 \ProcessKeysOptions { tikzinput }
3347
3348 \bool_if:NTF \c_tikzinput_image_bool {
3349   \RequirePackage{graphicx}
3350
3351   \providecommand\usetikzlibrary[]{}
3352   \newcommand\tikzinput[2][]{\includegraphics[#1]{#2}}
3353 }{
3354   \RequirePackage{tikz}
3355   \RequirePackage{standalone}
3356
3357   \newcommand \tikzinput [2] [] {
3358     \setkeys{Gin}{#1}
3359     \ifx \Gin@ewidth \Gin@exclamation
3360       \ifx \Gin@eheight \Gin@exclamation
3361         \input { #2 }
3362       \else
3363         \resizebox{!}{ \Gin@eheight }{
3364           \input { #2 }
3365         }
3366       \fi
3367     \else
3368       \ifx \Gin@eheight \Gin@exclamation
3369         \resizebox{ \Gin@ewidth }{!}{
3370           \input { #2 }
3371         }
3372       \fi
3373     \fi
3374   }
3375 }
```

```

3372     \else
3373     \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
3374     \input { #2 }
3375     }
3376     \fi
3377 \fi
3378 }
3379 }
3380
3381 \newcommand \ctikzinput [2] [] {
3382 \begin{center}
3383 \tikzinput [1] {#2}
3384 \end{center}
3385 }
3386
3387 \@ifpackageloaded{stex}{
3388 \RequirePackage{stex-tikzinput}
3389 }{}
3390
3391 </package>
3392 <*stex>
3393 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
3394 \RequirePackage{stex}
3395 \RequirePackage{tikzinput}
3396
3397 \newcommand\mhtikzinput[2] [] {%
3398 \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
3399 \stex_in_repository:nn\Gin@mhrepos{
3400 \tikzinput[#1]{\mhp{##1}{#2}}
3401 }
3402 }
3403 \newcommand\cmhtikzinput[2] [] {\begin{center}\mhtikzinput[#1]{#2}\end{center}}
3404 </stex>

```

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

Chapter 27

document-structure.sty Implementation

27.1 document-structure-common.sty Implementation

Common code for package and class: needs to be processed *before* loading the package in the class file:

```
3405 <*common>
3406 \ProvidesExplPackage{document-structure-common}{2020/10/19}{1.4}{OMDoc document Structure (c
```

Patching expl3, if outdated:

```
3407 <@@=keys>
3408 \cs_if_exist:cF { \c__keys_props_root_str .str_set:N }{
3409   \cs_new_protected:cpn { \c__keys_props_root_str .str_set:N } #1
3410   { \__keys_variable_set:NnnN #1 { str } { } n }
3411   \cs_new_protected:cpn { \c__keys_props_root_str .str_set:c } #1
3412   { \__keys_variable_set:cnnN {#1} { str } { } n }
3413   \cs_new_protected:cpn { \c__keys_props_root_str .str_set_x:N } #1
3414   { \__keys_variable_set:NnnN #1 { str } { } x }
3415   \cs_new_protected:cpn { \c__keys_props_root_str .str_set_x:c } #1
3416   { \__keys_variable_set:cnnN {#1} { str } { } x }
3417   \cs_new_protected:cpn { \c__keys_props_root_str .str_gset:N } #1
3418   { \__keys_variable_set:NnnN #1 { str } { g } n }
3419   \cs_new_protected:cpn { \c__keys_props_root_str .str_gset:c } #1
3420   { \__keys_variable_set:cnnN {#1} { str } { g } n }
3421   \cs_new_protected:cpn { \c__keys_props_root_str .str_gset_x:N } #1
3422   { \__keys_variable_set:NnnN #1 { str } { g } x }
3423   \cs_new_protected:cpn { \c__keys_props_root_str .str_gset_x:c } #1
3424   { \__keys_variable_set:cnnN {#1} { str } { g } x }
3425 }
3426 <@@=document_structure>
      keys:
3427 \keys_define:nn{ document-structure / clspkg }{
3428   class      .initial:n      = {article},
3429   class      .str_set_x:N     = \c_document_structure_class_str,
3430   topsect    .initial:n      = {section},
3431   topsect    .str_set_x:N     = \c_document_structure_topsect_str,
```

```

3432 showignores .bool_set:N = \c_document_structure_showignores_bool,
3433 minimal .bool_set:N = \c_document_structure_minimal_bool,
3434 report .code:n = {
3435   \ClassWarning{omdoc}{the option 'report' is deprecated, use 'class=report', instead}
3436   \str_set:Nn \c_document_structure_class_str {report}
3437 },
3438 book .code:n = {
3439   \ClassWarning{omdoc}{the option 'book' is deprecated, use 'class=book', instead}
3440   \str_set:Nn \c_document_structure_class_str {book}
3441 },
3442 bookpart .code:n = {
3443   \ClassWarning{omdoc}{the option 'bookpart' is deprecated, use 'class=book,topsect=chapter', instead}
3444   \str_set:Nn \c_document_structure_class_str {book}
3445   \str_set:Nn \c_document_structure_topsect_str {chapter}
3446 },
3447 docopt .str_set_x:N = \c_document_structure_docopt_str
3448 }
3449 </common>

```

27.2 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.

```

3450 <*cls>
3451 \ProvidesExplClass{omdoc}{2020/10/19}{1.4}{OMDoc Documents}
3452 \RequirePackage{document-structure-common}

```

27.3 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`

```

3453 \ProcessKeysOptions{ document-structure / clspkg }
3454 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
3455   {\c_document_structure_class_str}
3456

```

27.4 Beefing up the document environment

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

```

3457 \RequirePackage{omdoc}
3458 \bool_if:NF \c_document_structure_minimal_bool {
3459   \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

```

3460 \keys_define:nn { document-structure / document }{
3461   id .str_set_x:N = \c_document_structure_document_id_str
3462 }
3463 \let\__document_structure_orig_document=\document
3464 \renewcommand{\document}[1][]{
3465   \keys_set:nn{ document-structure / document }{ #1 }
3466   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
3467   \__document_structure_orig_document
3468 }

    Finally, we end the test for the minimal option.
3469 }
3470 </cls>

```

27.5 Implementation: OMDoc Package

```

3471 <*package>
3472 \ProvidesExplPackage{omdoc}{2020/10/19}{1.4}{OMDoc document Structure}
3473 \RequirePackage{document-structure-common}

```

27.6 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).

```

3474 \cs_if_exist:NF \c_document_structure_class_str {
3475   \ProcessKeysOptions{ document-structure / clspkg }
3476 }

```

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

```

3477 \RequirePackage{xspace}
3478 \RequirePackage{comment}
3479 \@ifpackageloaded{babel}{\RequirePackage[base]{babel}}

```

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

```

3480 \@ifpackageloaded{babel}{
3481   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
3482   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
3483     \input{omdoc-ngerman.ldf}
3484   }
3485 }{}
3486 %\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.

```

3487 \int_new:N \l_document_structure_section_level_int

```

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


```

3488 \str_case:VnF \c_document_structure_topsect_str {
3489   {part}{
3490     \int_set:Nn \l_document_structure_section_level_int {0}
3491   }
3492   {chapter}{
3493     \int_set:Nn \l_document_structure_section_level_int {1}
3494   }
3495 }{
3496   \str_case:VnF \c_document_structure_class_str {
3497     {book}{
3498       \int_set:Nn \l_document_structure_section_level_int {0}
3499     }
3500     {report}{
3501       \int_set:Nn \l_document_structure_section_level_int {0}
3502     }
3503   }{
3504     \int_set:Nn \l_document_structure_section_level_int {2}
3505   }
3506 }

```

27.7 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

```

3507 \def\current@section@level{document}%
3508 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
3509 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%

```

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

`\skipomgroup`

```

3510 \cs_new_protected:Npn \skipomgroup {
3511   \ifcase\l_document_structure_section_level_int
3512   \or\stepcounter{chapter}
3513   \or\stepcounter{section}
3514   \or\stepcounter{subsection}
3515   \or\stepcounter{subsubsection}
3516   \or\stepcounter{paragraph}
3517   \or\stepcounter{subparagraph}
3518   \fi
3519 }

```

(End definition for \skipomgroup. 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.

blindomgroup

```

3520 \newcommand\at@begin@blindomgroup[1]{
3521 \newenvironment{blindomgroup}
3522 {
3523 \int_incr:N\l_document_structure_section_level_int
3524 \at@begin@blindomgroup\l_document_structure_section_level_int
3525 }{
3526 \int_decr:N\l_document_structure_section_level_int
3527 }

```

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

```

3528 \newcommand\omgroup@nonum[2]{
3529 \ifx\hyper@anchor\undefined\else\phantomsection\fi
3530 \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
3531 }

```

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

`\omgroup@num` convenience macro: `\omgroup@num{<level>}{<title>}` makes numbered sectioning with title `<title>` at level `<level>`. We have to check the `short` key was given in the `omgroup` environment and – if it is use it. But how to do that depends on whether the `rdfmata` package has been loaded. In the end we call `\sref@label@id` to enable crossreferencing.

```

3532 \newcommand\omgroup@num[2]{
3533 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
3534 \@nameuse{#1}{#2}
3535 }{
3536 \cs_if_exist:NTF\rdfmata@sectioning{
3537 \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
3538 }{
3539 \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
3540 }
3541 }
3542 %\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\omgroup@id
3543 }

```

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

omgroup

```

3544 \keys_define:nn { document-structure / omgrou }{
3545 id .str_set_x:N = \l__document_structure_omgroup_id_str,
3546 date .str_set_x:N = \l__document_structure_omgroup_date_str,
3547 creators .clist_set:N = \l__document_structure_omgroup_creators_clist,
3548 contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
3549 srccite .tl_set:N = \l__document_structure_omgroup_srccite_tl,
3550 type .tl_set:N = \l__document_structure_omgroup_type_tl,
3551 short .tl_set:N = \l__document_structure_omgroup_short_tl,
3552 display .tl_set:N = \l__document_structure_omgroup_display_tl,
3553 intro .tl_set:N = \l__document_structure_omgroup_intro_tl,
3554 loadmodules .bool_set:N = \l__document_structure_omgroup_loadmodules_bool
3555 }
3556 \cs_new_protected:Nn \__document_structure_omgroup_args:n {
3557 \str_clear:N \l__document_structure_omgroup_id_str
3558 \str_clear:N \l__document_structure_omgroup_date_str

```

```

3559 \clist_clear:N \l__document_structure_omgroup_creators_clist
3560 \clist_clear:N \l__document_structure_omgroup_contributors_clist
3561 \tl_clear:N \l__document_structure_omgroup_srccite_tl
3562 \tl_clear:N \l__document_structure_omgroup_type_tl
3563 \tl_clear:N \l__document_structure_omgroup_short_tl
3564 \tl_clear:N \l__document_structure_omgroup_display_tl
3565 \tl_clear:N \l__document_structure_omgroup_intro_tl
3566 \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
3567 \keys_set:nn { document-structure / omgrou } { #1 }
3568 }

```

\at@begin@omgroup 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 omgrou, i.e. after the section heading.

```

3569 \newif\if@mainmatter\@mainmattertrue
3570 \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.

```

3571 \keys_define:nn { document-structure / sectioning }{
3572   name      .str_set_x:N = \l__document_structure_sect_name_str  ,
3573   ref       .str_set_x:N = \l__document_structure_sect_ref_str   ,
3574   clear     .bool_set:N  = \l__document_structure_sect_clear_bool ,
3575   num       .bool_set:N  = \l__document_structure_sect_num_bool   ,
3576 }
3577 \cs_new_protected:Nn \__document_structure_sect_args:n {
3578   \str_clear:N \l__document_structure_sect_name_str
3579   \str_clear:N \l__document_structure_sect_ref_str
3580   \bool_set_false:N \l__document_structure_sect_clear_bool
3581   \bool_set_false:N \l__document_structure_sect_num_bool
3582   \keys_set:nn { document-structure / sectioning } { #1 }
3583 }
3584 \newcommand\omdoc@sectioning[3] []{
3585   \__document_structure_sect_args:n {#1 }
3586   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
3587   \if@mainmatter% numbering not overridden by frontmatter, etc.
3588     \bool_if:NTF \l__document_structure_sect_num_bool {
3589       \omgroup@num{#2}{#3}
3590     }{
3591       \omgroup@nonum{#2}{#3}
3592     }
3593   \def\current@section@level{\omdoc@sect@name}
3594   \else
3595     \omgroup@nonum{#2}{#3}
3596   \fi
3597 }% 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.

```

3598 \newcommand\omgroup@redefine@addtocontents[1]{%
3599   %\edef\__document_structureimport{#1}%
3600   %\@for\@I:=\__document_structureimport\do{%
3601     %\edef\@path{\csname module@\@I @path\endcsname}%
3602     %\@ifundefined{tf@toc}\relax%
3603     %   {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%

```

```

3604 %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
3605 %\def\addcontentsline##1##2##3{%
3606 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}
3607 %\else% hyperref.sty not loaded
3608 %\def\addcontentsline##1##2##3{%
3609 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}
3610 %\fi
3611 }% 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.

```

3612 \int_new:N \l_document_structure_omgroup_level_int
3613 \newenvironment{omgroup}[2] []% keys, title
3614 {
3615   \__document_structure_omgroup_args:n { #1 }%\sref@target%
3616   \int_incr:N \l_document_structure_omgroup_level_int

```

If the `loadmodules` key is set on `\begin{omgroup}`, we redefine the `\addcontentsline` macro that determines how the sectioning commands below construct the entries for the table of contents.

```

3617   \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
3618     \omgroup@redefine@addtocontents{
3619       %\@ifundefined{module@id}\used@modules%
3620       %{\@ifundefined{module@}\module@id @path}{\used@modules}\module@id}
3621     }
3622   }

```

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

```

3623   \int_incr:N \l_document_structure_section_level_int
3624   \ifcase\l_document_structure_section_level_int
3625     \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
3626     \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
3627     \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
3628     \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
3629     \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
3630     \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
3631     \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
3632   \fi
3633   \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
3634   \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str
3635 }% for customization
3636 {
3637   \int_decr:N \l_document_structure_section_level_int
3638   \int_decr:N \l_document_structure_omgroup_level_int
3639 }

```

and finally, we localize the sections

```

3640 \newcommand\omdoc@part@kw{Part}
3641 \newcommand\omdoc@chapter@kw{Chapter}
3642 \newcommand\omdoc@section@kw{Section}
3643 \newcommand\omdoc@subsection@kw{Subsection}
3644 \newcommand\omdoc@subsubsection@kw{Subsubsection}
3645 \newcommand\omdoc@paragraph@kw{paragraph}
3646 \newcommand\omdoc@subparagraph@kw{subparagraph}

```

27.8 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`

```
3647 \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).

```
3648 \cs_if_exist:NTF\frontmatter{
3649   \let\__document_structure_orig_frontmatter\frontmatter
3650   \let\frontmatter\relax
3651 }{
3652   \tl_set:Nn\__document_structure_orig_frontmatter{
3653     \clearpage
3654     \@mainmatterfalse
3655     \pagenumbering{roman}
3656   }
3657 }
3658 \cs_if_exist:NTF\backmatter{
3659   \let\__document_structure_orig_backmatter\backmatter
3660   \let\backmatter\relax
3661 }{
3662   \tl_set:Nn\__document_structure_orig_backmatter{
3663     \clearpage
3664     \@mainmatterfalse
3665     \pagenumbering{roman}
3666   }
3667 }
```

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.

```
3668 \newenvironment{frontmatter}{
3669   \__document_structure_orig_frontmatter
3670 }{
3671   \cs_if_exist:NTF\mainmatter{
3672     \mainmatter
3673   }{
3674     \clearpage
3675     \@mainmattertrue
3676     \pagenumbering{arabic}
3677   }
3678 }
```

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

```
3679 \newenvironment{backmatter}{
3680   \__document_structure_orig_backmatter
3681 }{}
```

```

3682 \cs_if_exist:NTF\mainmatter{
3683   \mainmatter
3684 }{
3685   \clearpage
3686   \@mainmattertrue
3687   \pagenumbering{arabic}
3688 }
3689 }

```

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

```

3690 \@mainmattertrue\pagenumbering{arabic}

```

27.9 Ignoring Inputs

`ignore`

```

3691 \bool_if:NTF \c_document_structure_showignores_bool {
3692   \keys_define:nn { document-structure / ignore }{
3693     type .tl_set:N = \l__document_structure_ignore_type_tl ,
3694     comment .tl_set:N = \l__document_structure_ignore_comment_tl
3695   }
3696   \cs_new_protected:Nn \__document_structure_ignore_args:n {
3697     \tl_clear:N \l__document_structure_ignore_type_tl
3698     \tl_clear:N \l__document_structure_ignore_comment_tl
3699     \keys_set:nn { document-structure / ignore }{ #1 }
3700   }
3701   \newenvironment{ignore}[1][{}{
3702     \__document_structure_ignore_args:n { #1 }
3703     \textless \l__document_structure_ignore_type_tl \textgreater
3704     \bgroup \itshape
3705   }{
3706     \egroup
3707     \textless / \l__document_structure_ignore_type_tl \textgreater
3708   }
3709
3710   \renewenvironment{ignore}{}{} % - why?
3711 }{
3712   \excludecomment{ignore}
3713 }

```

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

```

3714 \newcommand\afterprematurestop{}
3715 \def\prematurestop@endomgroup{
3716   \int_compare:nNf \l_document_structure_omgroup_level_int = 0 {
3717     \end{\omgroup}
3718     \int_decr:N \l_document_structure_omgroup_level_int
3719     \prematurestop@endomgroup
3720   }
3721 }
3722 \providecommand\prematurestop{
3723   \message{Stopping sTeX processing prematurely}
3724   \prematurestop@endomgroup

```

```
3725 \afterprematurestop
3726 \end{document}
3727 }
```

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

Chapter 28

MiKoSlides – Implementation

28.1 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.

```
3728 <*cls>
3729 \RequirePackage{kvoptions}
3730 \RequirePackage{stex-metakeys}
3731 \RequirePackage{etoolbox}
3732 \SetupKeyvalOptions{family=mks@cls,prefix=mks@cls@}
3733 \DeclareStringOption[article]{class}
3734 \AddToKeyvalOption*{class}{\PassOptionsToClass{class=\mks@cls@class}{omdoc}
3735 \ifdefstring{\mks@cls@class}{book}{\PassOptionsToPackage{defaulttopsect=part}{mikoslides}}
3736 \ifdefstring{\mks@cls@class}{report}{\PassOptionsToPackage{defaulttopsect=part}{mikoslides}}
3737 \DeclareBoolOption{notes}
3738 \DeclareComplementaryOption{slides}{notes}
3739 \DeclareDefaultOption{%
3740 \PassOptionsToClass{\CurrentOption}{omdoc}
3741 \PassOptionsToClass{\CurrentOption}{beamer}
3742 \PassOptionsToPackage{\CurrentOption}{mikoslides}}
3743 \ProcessKeyvalOptions{mks@cls}
3744 </cls>
```

now we do the same for the `mikoslides` package.

```
3745 <*package>
3746 %\RequirePackage{stex-base}
3747 \RequirePackage{kvoptions}
3748 \RequirePackage{stex-metakeys}
3749 \SetupKeyvalOptions{family=mks@sty,prefix=mks@sty@}
3750 \DeclareStringOption{topsect}
3751 \DeclareStringOption{defaulttopsect}
3752 \newif\ifnotes\notesttrue
3753 \DeclareBoolOption{notes}
3754 \AddToKeyvalOption*{notes}{\notesttrue}%\PassOptionsToPackage{notes}{statements}}
3755 \DeclareComplementaryOption{slides}{notes}
3756 \AddToKeyvalOption*{slides}{\notestfalse}%\PassOptionsToPackage{nontheorem}{statements}}
```



```

3757 \DeclareBoolOption{sectocframes}
3758 \DeclareBoolOption{frameimages}
3759 \DeclareBoolOption{fiboxed}
3760 \DeclareBoolOption{nopproblems}
3761 % \DeclareDefaultOption{
3762 % \PassOptionsToPackage{\CurrentOption}{stex}
3763 % \PassOptionsToPackage{\CurrentOption}{smglom}
3764 % \PassOptionsToPackage{\CurrentOption}{tikzinput}}
3765 \ProcessKeyvalOptions{mks@sty}

```

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

```

3766 \ifx\mks@sty@topsect\empty\edef\@@topsect{\mks@sty@defaulttopsect}
3767 \else\edef\@@topsect{\mks@sty@topsect}\fi
3768 \</package>

```

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

```

3769 \*cls)
3770 \ifmks@cls@notes
3771 \LoadClass{omdoc}
3772 \else
3773 \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
3774 \newcounter{Item}
3775 \newcounter{paragraph}
3776 \newcounter{subparagraph}
3777 \newcounter{Hfootnote}
3778 \RequirePackage{omdoc}
3779 \fi

```

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

```

3780 \RequirePackage{mikoslides}
3781 \</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.

```

3782 \*package)
3783 \RequirePackage{stex-compatibility}
3784 \RequirePackage{stex-tikzinput}
3785 \ifmks@sty@notes
3786 \RequirePackage{a4wide}
3787 \RequirePackage{marginnote}
3788 \PassOptionsToPackage{dvipsnames,svgnames}{xcolor}
3789 \RequirePackage{mdframed}
3790 \RequirePackage[noxcolor,noamsthm]{beamerarticle}
3791 \fi
3792 \RequirePackage{etoolbox}
3793 \RequirePackage{amssymb}
3794 \RequirePackage{amsmath}
3795 \RequirePackage{comment}
3796 \RequirePackage{textcomp}

```

```

3797 \RequirePackage{url}
3798 \RequirePackage{graphicx}
3799 \RequirePackage{pgf}
3800 \ifmks@sty@notes
3801 \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
3802 \fi

    finally, we require the metakeys package from  $\TeX$ , so that we can use the
    \addmetakey mechanism.

3803 %\RequirePackage{metakeys}

```

28.2 Notes and Slides

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

```

3804 \ifmks@sty@notes
3805 \renewcommand\usetheme[2][\usepackage[#1]{beamernotestheme#2}]
3806 \fi

```

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

```

3807 \newcounter{slide}
3808 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
3809 \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.

```

3810 \ifmks@sty@notes%
3811 \renewenvironment{note}{\ignorespaces}{}%
3812 \else%
3813 \excludecomment{note}%
3814 \fi%

```

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.

```

3815 \ifmks@sty@notes
3816 \newlength{\slideframewidth}
3817 \setlength{\slideframewidth}{1.5pt}

```

frame We first define the keys.

```

3818 \addmetakey{frame}{label}
3819 \addmetakey[yes]{frame}{allowframebreaks}
3820 \addmetakey{frame}{allowdisplaybreaks}
3821 \addmetakey[yes]{frame}{fragile}
3822 \addmetakey[yes]{frame}{shrink}
3823 \addmetakey[yes]{frame}{squeeze}
3824 \addmetakey[yes]{frame}{t}

```

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

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

```

3825 \renewenvironment{frame}[1][]{%
3826 \metasetkeys{frame}{#1}%
3827 \sffamily%
3828 \stepcounter{slide}%
3829 \def\@currentlabel{\theslide}%
3830 \ifx\frame@label\@empty\else\label{\frame@label}\fi%

```

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

```

3831 \def\itemize@level{outer}%
3832 \def\itemize@outer{outer}%
3833 \def\itemize@inner{inner}%
3834 \renewcommand\newpage{\addtocounter{framenumber}{1}}%
3835 \renewcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}%
3836 \renewenvironment{itemize}{%
3837 \ifx\itemize@level\itemize@outer%
3838 \def\itemize@label{$\rhd$}%
3839 \fi%
3840 \ifx\itemize@level\itemize@inner%
3841 \def\itemize@label{$\scriptstyle\rhd$}%
3842 \fi%
3843 \begin{list}%
3844 {\itemize@label}%
3845 {\setlength{\labelsep}{.3em}%
3846 \setlength{\labelwidth}{.5em}%
3847 \setlength{\leftmargin}{1.5em}%
3848 }%
3849 \edef\itemize@level{\itemize@inner}%
3850 }{%
3851 \end{list}%
3852 }%

```

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

```

3853 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth]
3854 }{%
3855 \medskip\miko@slidelabel\end{mdframed}%
3856 }%

```

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

`\frametitle`

```

3857 \renewcommand{\frametitle}[1]{\Large\bf\sffcolor{blue}{#1}\medskip}%
3858 \fi %ifmks@sty@notes

```

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

EdN:11

`\pause` 11

```

3859 \ifmks@sty@notes\newcommand\pause{}\fi

```

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

`nomtext`

```

3860 \ifmks@sty@notes\newenvironment{nomtext}[1][]{\begin{omtext}{#1}}{\end{omtext}}%
3861 \else\excludecomment{nomtext}\fi%

```

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

```

nomgroup
3862 \ifmks@sty@notes\newenvironment{nomgroup}[2][\begin{omgroup}[#1]{#2}}{\end{omgroup}}%
3863 \else\excludecomment{nomgroup}\fi%

ndefinition
3864 \ifmks@sty@notes\newenvironment{ndefinition}[1][\begin{definition}[#1]]{\end{definition}}%
3865 \else\excludecomment{ndefinition}\fi%

nassertion
3866 \ifmks@sty@notes\newenvironment{nassertion}[1][\begin{assertion}[#1]]{\end{assertion}}%
3867 \else\excludecomment{nassertion}\fi%

nsproof
3868 \ifmks@sty@notes\newenvironment{nsproof}[2][\begin{sproof}[#1]{#2}}{\end{sproof}}%
3869 \else\excludecomment{nsproof}\fi%

nexample
3870 \ifmks@sty@notes\newenvironment{nexample}[1][\begin{example}[#1]]{\end{example}}%
3871 \else\excludecomment{nexample}\fi%

\inputref*skip We customize the hooks for in \inputref.
3872 \def\inputref@preskip{\smallskip}
3873 \def\inputref@postskip{\medskip}

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

\inputref*
3874 \let\orig@inputref\inputref
3875 \def\inputref{\@ifstar\ninputref\orig@inputref}
3876 \newcommand\ninputref[2][\ifmks@sty@notes\orig@inputref[#1]{#2}\fi}

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

```

28.3 Header and Footer Lines

Now, we set up the infrastructure for the footer line of the slides, we use boxes for the logos, so that they are only loaded once, that considerably speeds up processing.

```

\setslidelogo The default logo is the  $\TeX$  logo. Customization can be done by \setslidelogo{<logo
name>}.
3877 \newlength{\slidelogoheight}
3878 \ifmks@sty@notes%
3879 \setlength{\slidelogoheight}{.4cm}%
3880 \else%
3881 \setlength{\slidelogoheight}{1cm}%
3882 \fi%
3883 \newsavebox{\slidelogo}%
3884 \sbox{\slidelogo}{\TeX}%
3885 \newrobustcmd{\setslidelogo}[1]{%
3886 \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}%
3887 }%

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

```

`\setsource` `\source` stores the writer's name. By default it is *Michael Kohlhase* since he is the main user and designer of this package. `\setsource{<name>}` can change the writer's name.

```
3888 \def\source{Michael Kohlhase}% customize locally
3889 \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.

```
3890 \def\copyrightnotice{\footnotesize\copyright:\hspace{.3ex}{\source}}%
3891 \newsavebox{\cclogo}%
3892 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}%
3893 \newif\ifcchref\cchreffalse%
3894 \AtBeginDocument{%
3895   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
3896 }%
3897 \def\licensing{%
3898   \ifcchref%
3899     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}%
3900   \else%
3901     {\usebox{\cclogo}}%
3902   \fi%
3903 }%
3904 \newrobustcmd{\setlicensing}[2][{}]{%
3905   \def\@url{#1}%
3906   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}%
3907   \ifx\@url\@empty%
3908     \def\licensing{{\usebox{\cclogo}}}%
3909   \else%
3910     \def\licensing{%
3911       \ifcchref%
3912         \href{#1}{\usebox{\cclogo}}%
3913       \else%
3914         {\usebox{\cclogo}}%
3915       \fi%
3916     }%
3917   \fi%
3918 }%
```

(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

```
3919 \newrobustcmd\miko@slidelabel{%
3920   \vbox to \slidelogoheight{%
3921     \vss\hbox to \slidewidth%
3922     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}%
3923   }%
3924 }%
3925 % \subsection{Frame Images}\label{sec:impl:frameimage}
3926 %
```

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

```

3927 % \begin{macro}{\frameimage}
3928 % We have to make sure that the width is overwritten, for that we check the
3929 % |\Gin@ewidth| macro from the |graphicx| package. We also add the |label| key.
3930 % \begin{macrocode}
3931 \def\Gin@mhrepos{}
3932 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
3933 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
3934 \newrobustcmd\frameimage[2][{}]{%
3935   \stepcounter{slide}%
3936   \ifmks@sty@frameimages%
3937     \def\Gin@ewidth{}\setkeys{Gin}{#1}%
3938     \ifmks@sty@notes\else\vfill\fi%
3939     \begin{center}
3940       \ifmks@sty@fiboxed%
3941         \fbox{\ifx\Gin@ewidth\@empty%
3942           \ifx\Gin@mhrepos\@empty\mhgraphics[width=\slidewidth,#1]{#2}%
3943           \else\mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}\fi%
3944           \else% Gin@ewidth empty
3945           \ifx\Gin@mhrepos\@empty\mhgraphics[#1]{#2}%
3946           \else\mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}\fi%
3947           \fi}% Gin@ewidth empty
3948         \else% ifmks@sty@fiboxed
3949         \ifx\Gin@ewidth\@empty%
3950           \ifx\Gin@mhrepos\@empty\mhgraphics[width=\slidewidth,#1]{#2}%
3951           \else\mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}\fi%
3952           \ifx\Gin@mhrepos\@empty\else\mhgraphics[#1]{#2}%
3953           \else\mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}\fi%
3954           \fi% Gin@ewidth empty
3955         \fi% ifmks@sty@fiboxed
3956       \end{center}
3957       \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
3958     \ifmks@sty@notes\else\vfill\fi%
3959   \fi} % ifmks@sty@frameimages

```

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

28.4 Colors and Highlighting

We first specify sans serif fonts as the default.

```

3960 \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

```

3961 \AtBeginDocument{%
3962   \definecolor{green}{rgb}{0,.5,0}%
3963   \definecolor{purple}{cmyk}{.3,1,0,.17}%
3964 }%

```

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

```

3965 % \def\STpresent#1{\textcolor{blue}{#1}}

```

```

3966 \def\defemph#1{\textcolor{magenta}{#1}}
3967 \def\symrefemph#1{\textcolor{cyan}{#1}}
3968 \def\compemph#1{\textcolor{blue}{#1}}
3969 \def\titleemph#1{\textcolor{blue}{#1}}
3970 \def\@lec#1{\textcolor{green}{#1}}

```

I like to use the dangerous bend symbol for warnings, so we provide it here.

`\textwarning` as the macro can be used quite often we put it into a box register, so that it is only loaded once.

```

3971 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
3972 \def\smalltextwarning{%
3973   \pgfuseimage{miko@small@dbend}%
3974   \xspace%
3975 }%
3976 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}
3977 \newrobustcmd\textwarning{%
3978   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}%
3979   \xspace%
3980 }%
3981 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}%
3982 \newrobustcmd\bigtextwarning{%
3983   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}%
3984   \xspace%
3985 }%

```

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

```

3986 \newrobustcmd\putgraphicsat[3]{%
3987   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}}%
3988 }%
3989 \newrobustcmd\putat[2]{%
3990   \begin{picture}(0,0)\put(#1){#2}\end{picture}}%
3991 }%

```

28.5 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`.

```

3992 \ifmks@sty@sectocframes%
3993 \ifdefstring\@topsect{part}{%
3994   \newcounter{chapter}\counterwithin*{section}{chapter}}
3995 {\ifdefstring\@topsect{chapter}{\newcounter{chapter}\counterwithin*{section}{chapter}}{}}
3996 \fi% ifsectocframes

```

`\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
3997 \section@level=2
3998 \def\part@prefix{}
3999 \ifdefstring\@topsect{part}{
4000   \section@level=0%
4001   \def\thesection{\arabic{chapter}.\arabic{section}}%

```

```

4002 \def\part@prefix{\arabic{chapter}.}}{}
4003 \ifdefstring{\@@topsect}{chapter}
4004 {\section@level=1%
4005 \def\thesection{\arabic{chapter}.\arabic{section}}}%
4006 \def\part@prefix{\arabic{chapter}.}}{}
4007 \ifmks@sty@notes\else% only in slides

```

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

The new counters are used in the omgroup environment that choses the L^AT_EX sectioning macros according to \section@level.

omgroup

```

4008 \renewenvironment{omgroup}[2][{}]{%
4009 \metasetkeys{omgroup}{#1}%
4010 \advance\section@level by 1%
4011 \advance\omgroup@level by 1%
4012 \ifmks@sty@sectocframes%
4013 \stepcounter{slide}
4014 \begin{frame}[noframenumbering]%
4015 \vfill\Large\centering%
4016 \red{%
4017 \ifcase\section@level\or
4018 \stepcounter{part}
4019 \def\@@label{\omdoc@part@kw~\Roman{part}}
4020 \def\currentsectionlevel{\omdoc@part@kw}
4021 \or%
4022 \stepcounter{chapter}
4023 \def\@@label{\omdoc@chapter@kw~\arabic{chapter}}
4024 \def\currentsectionlevel{\omdoc@chapter@kw}
4025 \or
4026 \stepcounter{section}
4027 \def\@@label{\part@prefix\arabic{section}}
4028 \def\currentsectionlevel{\omdoc@section@kw}
4029 \or
4030 \stepcounter{subsection}
4031 \def\@@label{\part@prefix\arabic{section}.\arabic{subsection}}
4032 \def\currentsectionlevel{\omdoc@subsection@kw}
4033 \or
4034 \stepcounter{subsubsection}
4035 \def\@@label{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
4036 \def\currentsectionlevel{\omdoc@subsubsection@kw}
4037 \or
4038 \stepcounter{mparagraph}
4039 \def\@@label{\part@prefix\arabic{section}.\arabic{msubsection}.\arabic{subsubsection}.\arabic{mparagraph}}
4040 \def\currentsectionlevel{\omdoc@paragraph@kw}
4041 \fi% end ifcase
4042 \@@label\sref@label@id\@@label
4043 \quad #2%
4044 }%
4045 \vfill%
4046 \end{frame}%
4047 \fi %ifmks@sty@sectocframes
4048 \csname stex_ref_new_doc_target:n\endcsname\omgroup@id%
4049 }

```



```

4050 {\advance\section@level by -1}%
4051 \fi% ifmks@sty@notes

```

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

```

4052 \def\inserttheorembodyfont{\normalfont}
4053 \ifmks@sty@notes\else% only in slides
4054 \defbeamertemplate{theorem begin}{miko}
4055 {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
4056 \ifx\inserttheoremaddition\empty\else\ (\inserttheoremaddition)\fi%
4057 \inserttheorempunctuation\inserttheorembodyfont\xspace}
4058 \defbeamertemplate{theorem end}{miko}{\fi}

```

and we set it as the default one.

```

4059 \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.

```

4060 \expandafter\def\csname Parent2\endcsname{}
4061 \fi% ifmks@sty@notes
4062 \ifmks@sty@notes%
4063 \renewenvironment{columns}[1][\fi]{%
4064 \par\noindent%
4065 \begin{minipage}%
4066 \slidewidth\centering\leavevmode%
4067 }{%
4068 \end{minipage}\par\noindent%
4069 }%
4070 \newsavebox\columnbox%
4071 \renewenvironment<>{column}[2][\fi]{%
4072 \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
4073 }{%
4074 \end{minipage}\end{lrbox}\usebox\columnbox%
4075 }%
4076 \fi% ifmks@sty@notes
4077 \ifmks@sty@noproblems%
4078 \newenvironment{problems}{}{}%
4079 \else%
4080 \excludecomment{problems}%
4081 \fi%

```

28.6 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.

```

4082 \gdef\printexcursions{}
4083 \newcommand\excursionref[2]{% label, text
4084 \ifnotes\begin{omtext}[title=Excursion]{#2 \sref[fallback=the appendix]{#1}.\end{omtext}}\fi}
4085 \newcommand\activate@excursion[2][\gappto\printexcursions{\inputref{#1}{#2}}]
4086 \newcommand\excursion[4][\fi]{% repos, label, path, text
4087 \ifnotes\activate@excursion[#1]{#3}\excursionref{#2}{#4}\fi}%

```

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

\excursiongroup

```
4088 \srefaddidkey{excursiongroup}%
4089 \addmetakey{excursiongroup}{intro}%
4090 \addmetakey{excursiongroup}{mhrepos}%
4091 \newcommand\excursiongroup[1][\metasetkeys{excursiongroup}{#1}%
4092   \ifdefempty\printexcursions{ }% only if there are excursions
4093   {\begin{omgroup}[#1]{Excursions}%
4094     \ifdefempty\excursiongroup@intro{\inputref[\excursiongroup@mhrepos]{\excursiongroup@
4095       \printexcursions%
4096     \end{omgroup}}}%
4097 \end{package}
```

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

28.7 Miscellaneous