

# **stex.sty: $\text{\TeX}$ 2.0\***

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

2021-09-15

## **Abstract**

TODO

## **1 Introduction**

TODO

---

\*Version v1.9 (last revised 2021/08/01)

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Manual</b>	<b>3</b>
2.1	Modules . . . . .	3
2.2	Semantic Macros and Notations . . . . .	3
2.3	Archives and Imports . . . . .	7
<b>3</b>	<b>Documentation</b>	<b>8</b>
3.1	Utils . . . . .	8
3.2	Files, Paths, URIs . . . . .	9
3.3	MathHub Archives . . . . .	10
3.4	The Module System . . . . .	12
3.5	Symbols and Terms . . . . .	20
<b>4</b>	<b>Implementation</b>	<b>25</b>
4.1	The $\text{\LaTeX}$ document class . . . . .	25
4.2	Preliminaries . . . . .	25
4.3	Files, Paths and URIs . . . . .	30
4.4	MathHub Repositories . . . . .	33
4.5	Module System . . . . .	37
4.6	Symbol Declarations . . . . .	55
4.7	Notations . . . . .	61
4.8	Terms . . . . .	69
4.9	Notation Components . . . . .	75
4.10	Put these somewhere . . . . .	77
4.11	Metatheory . . . . .	77
4.12	Auxiliary Packages . . . . .	79

## 2 Manual

### 2.1 Modules

`{module}`, `{@module}`

### 2.2 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

$$\backslash\mathrm{symdef}[args=2]\{mult\}\{#1\ #2\}$$

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 changed<sup>1</sup>.

---

<sup>1</sup>EdNOTE: TODO

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  $x \in A$

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.

### Example 6

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

*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]`

### 2.2.1 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,  $\text{\TeX}$  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  $\text{\TeX}$ , but are treated very differently in OMDoc and by MMT. More interesting *within*  $\text{\TeX}$  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 7

```
\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 8

```
\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.

### 2.2.2 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  $\texttt{\textbackslash foo}$  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.

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

Notations for symbols of arity 0 have a default precedence of  $\texttt{\textbackslash infprec}$ , 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  $A$ s operator precedence should be larger than  $B$ s argument precedences.

For example:

### Example 9

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

$$a + b \cdot c \text{ and } a \cdot (b + c)$$

<sup>2</sup>EdNOTE: what about e.g.  $\int \int \int f \, dx \, dy \, dz$ ?

<sup>3</sup>EdNOTE: “decompose” a-type arguments into fixed-arity operators?

## 2.3 Archives and Imports

### 2.3.1 Namespaces

Ideally,  $\text{\S\TeX}$  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,  $\text{\TeX}$  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  $\text{\S\TeX}$  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  $\text{\begin{module}\{Foo\}}$  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 suffix<sup>1</sup>.

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.

### 2.3.2 Paths in Import-Statements

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

- $\text{\importmodule}\{Foo\}$  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  $\text{\importmodule}\{some/path?Foo\}$  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,  $\text{\importmodule}[Some/Archive]\{some/path?Foo\}$  is resolved like the previous cases, but relative to the archive `Some/Archive` in the mathhub-directory.

---

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

- 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.

## 3 Documentation

### 3.1 Utils

<hr/> <code>\sTeX</code> <code>\stex</code> <hr/>	both print this $\text{\TeX}$ logo.
<hr/> <code>\stex_debug:n</code> <hr/>	<code>\stex_debug:n {<i>message</i>}</code> Logs <i>message</i> , if the package option <code>debug</code> is used.

<hr/> <code>\stex_kpsewhich:n</code> <hr/>	<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.
--	---

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

#### 3.1.1 $\text{\LaTeX}$ , $\text{\LaTeXML}$ and $\text{\HTML}$ Annotations

<hr/> <code>\if@latexml</code> <code>\latexml_if_p:</code> <code>\latexml_if:T</code> <code>\latexml_if:F</code> <code>\latexml_if:TF</code> <hr/>	$\text{\LaTeX}$ 2e and $\text{\LaTeX}$ 3 conditionals for $\text{\LaTeXML}$ .
--	---

We have four macros for annotating generated  $\text{\HTML}$  (via  $\text{\LaTeXML}$  or  $\text{\ScalTeX}$ ) with attributes:

<hr/> <code>\stex_annotate:nnn</code> <code>\stex_annotate_invisible:nnn</code> <code>\stex_annotate_invisible:n</code> <hr/>	<code>\stex_annotate:nnn {<i>property</i>} {<i>resource</i>} {<i>content</i>}</code>
---	--

Annotates the  $\text{\HTML}$  generated by *content* with

`property="stex:~{property}", resource="~{resource}"`.

`\stex_annotate_invisible:n` adds the attributes

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

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



<code>stex_annotate_env</code>	$\begin{array}{l} \backslash\text{begin}\{\text{stex\_annotate\_env}\}\{\langle\text{property}\rangle\}\{\langle\text{resource}\rangle\} \\ \langle\text{content}\rangle \\ \backslash\text{end}\{\text{stex\_annotate\_env}\} \end{array}$ behaves like <code>\stex_annotate:nnn</code> $\{\langle\text{property}\rangle\} \{\langle\text{resource}\rangle\} \{\langle\text{content}\rangle\}$ .
--------------------------------	---

### 3.1.2 Languages

---

<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`.

## 3.2 Files, Paths, URIs

---

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

---

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

---

<code>\stex_path_to_string:NN</code>
<code>\stex_path_to_string:N</code>

---

The inverse; turns a path into a string and stores it in the second argument variable, or leaves it in the input stream.

---

<code>\stex_path_canonicalize:N</code>
--

---

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

---

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

---

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

---

<code>\c_stex_pwd_seq</code>
<code>\c_stex_pwd_str</code>
<code>\c_stex_mainfile_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 `\jobname`.

---

<code>\g_stex_currentfile_seq</code>
--------------------------------------

---

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 / .. / ..		

## 3.3 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).

---

---

`\stex_set_current_repository:n`

Sets the current repository to the one with the provided ID. calls `\__stex_mathhub_do_manifest:n`, so works whether this repository's MANIFEST.MF-file has already been read or not.

---

---

`\stex_require_repository:n`

Calls `\__stex_mathhub_do_manifest:n` iff the corresponding archive property list does not already exist, and adds a corresponding definition to the `.sms`-file.

---

---

`\libinput`

`\libinput{<filename>}`

Inputs `<filename>.tex` from the `lib` folders in the current archive and the `meta-inf`-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:
```

### 3.4 The Module System

---

`\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`,

---

`\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.

---

<code>\stex_modules_compute_namespace:nN</code>	<code>\stex_modules_compute_namespace:nN</code> <code>{\langle namespace \rangle} {\langle path \rangle}</code>
---	--

---

Computes the namespace for file  $\langle path \rangle$  in repository with namespace  $\langle namespace \rangle$  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`.

---

<code>\stex_modules_current_namespace:</code>
---

---

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/sty/stex-master/stextest
Faking a repository:
Namespace 2:
http://mathhub.info/tests/Foo/Bar/test/stextest

```

#### 3.4.1 The module-environment

<code>module</code>	<code>\begin{module}[\langle options \rangle]{\langle name \rangle}</code> Opens a new module with name $\langle name \rangle$ . TODO document options.
---------------------	---

---

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

---

<code>@module</code>	<code>\begin{@module}[\langle options \rangle]{\langle name \rangle}</code> 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:n{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:n{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 3.1[Bar] (FooBar)
Module path: http://mathhub.info/tests/Foo/Bar/Foo?Bar
Language:
Signature:
Metatheory:
```

\l\_stex\_all\_modules\_seq

Stores full URIs for all modules currently in scope.

\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 `!\langle macro \rangle` or `?{\langle symbolname \rangle}`. In the first case, it stores the full URI in `\langle macro \rangle`; in the second case, it invokes the symbol `\langle symbolname \rangle` in the selected module.

### Test 6

```
\begin{module}{STEXModuleTest1}
\syndeci{foo}
\end{module}
\begin{module}{STEXModuleTest2}
\importmodule{STEXModuleTest1}
\syndeci{foo}
\end{module}
\begin{module}{STEXModuleTest3}
\importmodule{STEXModuleTest2}
\syndeci{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}
```

Module 3.2[STEXModuleTest1]

Module 3.3[STEXModuleTest2]

Module 3.4[STEXModuleTest3]  
file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?STEXModuleTest1  
file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?STEXModuleTest2  
file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?STEXModuleTest3  
foo1  
foo2  
foo3

### 3.4.2 SMS Mode

“SMS Mode” is used when loading modules from external tex files. It deactivates any output and ignores all  $\TeX$  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
```

### 3.4.3 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 3.5[Foo]
Meaning: >macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?Foo?foo}<
```

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

```
Module 3.6[Importtest]
Meaning: >macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?Foo?foo}<
```

```
Module 3.7[Importtest2]
Meaning: >macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/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 3.8[UseTest1]

Module 3.9[UseTest2]
Meaning: »macro:->\stex\_invoke\_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?UseTest1?foo}<

Module 3.10[UseTest3]
Meaning: »undefined<
Meaning: »macro:->\stex\_invoke\_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?UseTest2?bar}<
All modules: file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?UseTest3, http://mathhub.info/sTeX?Metath
file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?UseTest2
All symbols: http://mathhub.info/sTeX?Metatheory?isa, http://mathhub.info/sTeX?Metatheory?bind, http://mathhub.info/sTeX?Metatheo
http://mathhub.info/sTeX?Metatheory?fromto, http://mathhub.info/sTeX?Metatheory?apply, http://mathhub.info/sTeX?Metatheory?collec
http://mathhub.info/sTeX?Metatheory?seqtype, http://mathhub.info/sTeX?Metatheory?sequence-index, http://mathhub.info/sTeX?Metath
http://mathhub.info/sTeX?Metatheory?letin, http://mathhub.info/sTeX?Metatheory?mathematical-structure,
file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/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 3.11[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}<

<hr/> <hr/>	<hr/>
<code>\stex_import_module_uri:nn</code>	<code>\stex_import_module_uri:nn {&lt;archive-ID&gt;} {&lt;module-path&gt;}</code>
	Determines the URI of a module by splitting <code>&lt;module-path&gt;</code> into <code>&lt;path&gt;?&lt;name&gt;</code> . If <code>&lt;module-path&gt;</code> does <i>not</i> contain a <code>?</code> -character, we consider it to be the <code>&lt;name&gt;</code> , and <code>&lt;path&gt;</code> to be empty. If <code>&lt;archive-ID&gt;</code> is empty, it is automatically set to the ID of the current archive (if one exists).
	1. If <code>&lt;archive-ID&gt;</code> is empty:
	(a) If <code>&lt;path&gt;</code> is empty, then <code>&lt;name&gt;</code> must have been declared earlier in the same file and retrievable from <code>\g_stex_modules_in_file_seq</code> , or a file with name <code>&lt;name&gt;.&lt;lang&gt;.tex</code> must exist in the same folder, containing a module <code>&lt;name&gt;</code> . That module should have the same namespace as the current one.
	(b) If <code>&lt;path&gt;</code> is not empty, it must point to the relative path of the containing file as well as the namespace.
	2. Otherwise:
	(a) If <code>&lt;path&gt;</code> is empty, then <code>&lt;name&gt;</code> must have been declared earlier in the same file and retrievable from <code>\g_stex_modules_in_file_seq</code> , or a file with name <code>&lt;name&gt;.&lt;lang&gt;.tex</code> must exist in the top <code>source</code> folder of the archive, containing a module <code>&lt;name&gt;</code> . That module should lie directly in the namespace of the archive.
	(b) If <code>&lt;path&gt;</code> 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 <code>\stex_require_module:nn</code> on the <code>source</code> directory of the archive to find the file.

---

---

<code>\stex_import_require_module:nnnn</code>	<code>{&lt;ns&gt;} {&lt;archive-ID&gt;} {&lt;path&gt;} {&lt;name&gt;}</code>
---	--

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.

---

---

<code>\g_stex_module_files_prop</code>	
<code>\g_stex_modules_in_file_seq</code>	

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).

---

---

<code>\stex_activate_module:n</code>	Activate the module with the provided URI; i.e. executes all macro code of the module's <code>content</code> -field (does nothing if the module is already activated in the current context)
--------------------------------------	--

### 3.5 Symbols and Terms

---

`\symdecl` `\symdecl[⟨args⟩]{⟨macroname⟩}`

---

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  $\text{\S}\text{\TeX}$ , 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  $\text{\S}\text{\TeX}$  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\text{\Nat}}{x\geq 0}`.

---

`\abbrdef` `\abbrdef[⟨args⟩]{⟨macroname⟩}{⟨term⟩}`

---

`\abbrdef` behaves like `\symdecl`, but adds the definiens `⟨term⟩` to the symbol. The latter is largely ignored and irrelevant to  $\text{\S}\text{\TeX}$ , but exported to OMDOC.

---

`\stex_symdecl_do:n`

---

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

Ultimately stores the symbol `⟨URI⟩` 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}
\abbrdef{bardef}{\bar* abc}
\ExplSyntaxOn
Meaning:-\present\bar\
\stex_get_symbol:n { bar }
Result:-\l_stex_get_symbol_uri_str\
Meaning:-\present\bardef\
\ExplSyntaxOff
\end{module}
```

```
Module 3.12[SymdeclTest]
Meaning: »macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?SymdeclTest?foo}<
Result: file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?SymdeclTest?foo
Meaning: »macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/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...

---

---

`\STEXsymbol`

Uses `\stex_get_symbol:n` to find the symbol denoted by the first argument and passes the result on to `\stex_invoke_symbol:n`

---

---

`\symref`

`\symref{⟨symbol⟩}{⟨text⟩}`  
 shortcut for `\STEXsymbol{⟨symbol⟩}! [⟨text⟩]`

---

---

`\stex_invoke_symbol:n`

Executes a semantic macro. Outside of math mode or if followed by `*`, it continues to `\stex_term_custom:nn`. In math mode, it uses the default or optionally provided notation of the associated symbol.

If followed by `!`, it will invoke the symbol *itself* rather than its application (and continue to `\stex_term_custom:nn`), i.e. it allows to refer to `\plus!`[addition] as an operation, rather than `\plus`[addition of]{some}{terms}.

---

---

`\notation`

`\notation[⟨args⟩]{⟨symbol⟩}{⟨notations+⟩}`  
 Introduces a new notation for `⟨symbol⟩`, 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),
- argprec (sequence of integer strings)

### Test 12

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

Module 3.13[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 3.14[SymdefTest]  
 $a+b+c$

---

`\_stex_term_math_oms:nnnn`  
`\_stex_term_math_oma:nnnn`  
`\_stex_term_math_omb:nnnn`

---

`<URI><fragment><precedence><body>`

Annotates `<body>` as an OMDOC-term (OMID, OMA or OMBIND, respectively) with head symbol `<URI>`, generated by the specific notation `<fragment>` with (upwards) operator precedence `<precedence>`. Inserts parentheses according to the current downwards precedence and operator precedence.



```

Module 3.16[MathTest2]
  <a|[b,c,d,e,f]g> and <a|[b,c]g> and <a|[b]c>
  a+b·c and a·(a/b+a/c)
  a+b·c and a·(a/b+a/c)
  a+b·c and a·(a/b+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 3.17[TextTest]
  some a and some b and also some c here.
  some a and some b and also some c here.
  bar
  or just some c
  bar
  or first b, then c, and finally a

```

---

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

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

---

`\comp` `\comp{<args>}`

`\@comp` 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.

---

`\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

---

## 4 Implementation

### 4.1 The `sTeX` document class

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

### 4.2 Preliminaries

```
13 <*package>
14 \RequirePackage{expl3,l3keys2e}
15 \ProvidesExplPackage{stex}{2021/08/01}{1.9}{bla}
16
17 Package options:
18 \keys_define:nn { stex } {
19   debug      .bool_set:N = \c_stex_debug_bool ,
20   showmods   .bool_set:N = \c_stex_showmods_bool ,
21   lang        .clist_set:N = \c_stex_languages_clist ,
22   mathhub     .tl_set_x:N = \mathhub ,
23   sms         .bool_set:N = \c_stex_persist_mode_bool ,
24   image       .bool_set:N = \c_tikzinput_image_bool
25 }
26 \ProcessKeysOptions { stex }
```

`\sTeX`    The `sTeX` logo:

```
25 \protected\def\sTeX{%
26   \@ifundefined{texorpdfstring}%
27   {\let\texorpdfstring\@firstoftwo}%
28   }%
29   \texorpdfstring{\raisebox{-.5ex}{S\kern-.5exTeX}{sTeX}\xspace}%
30 }
31 \def\sTeX{\sTeX}
```

(End definition for `\sTeX`. This function is documented on page 8.)

Messages

```
32 \msg_new:nnn{stex}{debug}{}
33 \msg_new:nnn{stex}{warning/nomathhub}{
34   MATHHUB~system~variable~not~found~and~no~
35   \detokenize{\mathhub}~value~set!
36 }
37 \msg_new:nnn{stex}{error/norepository}{}
38
```

**\stex\_debug:n** Debug mode

```

38 \cs_new_protected:Nn \stex_debug:n {
39   \bool_if:nT{\c_stex_debug_bool}{
40     \exp_args:Nnnx\msg_set:nnn{stex}{debug}{\\Debug:~#1\\}
41     \msg_term:nn{stex}{debug} % should be \msg_note:nn
42   }
43 }
44
45 \stex_debug:n{Debug~mode~on}

```

(End definition for \stex\_debug:n. This function is documented on page 8.)

**\c\_\_stex\_sms\_iow** File variable used for the sms-File

```

46 \iow_new:N \c__stex_sms_iow
47 \AddToHook{begindocument}{
48   \bool_if:NTF \c_stex_persist_mode_bool {
49     \ExplSyntaxOn \input{\jobname.sms} \ExplSyntaxOff
50   } {
51     \iow_open:Nn \c__stex_sms_iow {\jobname.sms}
52   }
53 }
54 \AddToHook{enddocument}{
55   \bool_if:NF \c_stex_persist_mode_bool {
56     \iow_close:N \c__stex_sms_iow
57   }
58 }

```

(End definition for \c\_\_stex\_sms\_iow.)

**\stex\_addtosms:n**

```

59 \cs_new_protected:Nn \stex_addtosms:n {
60   \bool_if:NF \c_stex_persist_mode_bool {
61     \iow_now:Nn \c__stex_sms_iow { #1 }
62   }
63 }

```

(End definition for \stex\_addtosms:n. This function is documented on page 8.)

#### 4.2.1 L<sup>A</sup>T<sub>E</sub>X<sub>ML</sub> and S<sup>C</sup>A<sub>L</sub>T<sub>E</sub>X

```

64 \RequirePackage{scalatex}

```

We add the namespace abbreviation `ns:stex="http://kwarc.info/ns/sTeX"` to S<sup>C</sup>A<sub>L</sub>T<sub>E</sub>X:

```

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

```

**\if@latexml** Conditionals for L<sup>A</sup>T<sub>E</sub>X<sub>ML</sub>:

```

\latexml_if_p:
\latexml_if:TF
66 \ifcsname if@latexml\endcsname\else
67   \expandafter\newif\csname if@latexml\endcsname\@latexmlfalse
68 \fi
69
70 \prg_new_conditional:Nnn \latexml_if: {p, T, F, TF} {
71   \if@latexml
72     \prg_return_true:
73   \else:

```

```

74     \prg_return_false:
75   \fi:
76 }

```

(End definition for `\if@latexml` and `\latexml_if:TF`. These functions are documented on page 8.)

## 4.2.2 HTML Annotations

```

77 <@=stex_annotate>

```

`\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
78 \tl_new:N \l__stex_annotate_arg_tl
79 \tl_const:Nx \c__stex_annotate_emptyarg_tl {
80   \scalatex_if:TF {
81     \scalatex_direct_HTML:n { \c_ampsand_str lrm; }
82   }{-}
83 }

```

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

```

\__stex_annotate_checkempty:n

```

```

84 \cs_new_protected:Nn \__stex_annotate_checkempty:n {
85   \tl_set:Nn \l__stex_annotate_arg_tl { #1 }
86   \tl_if_empty:NT \l__stex_annotate_arg_tl {
87     \tl_set_eq:NN \l__stex_annotate_arg_tl \c__stex_annotate_emptyarg_tl
88   }
89 }

```

(End definition for `\__stex_annotate_checkempty:n`.)

```

\stex_annotate:nnx
\stex_annotate_invisible:n
\stex_annotate_invisible:nnn

```

We define four macros for introducing attributes in the HTML output. The definitions depend on the “backend” used (L<sup>A</sup>T<sub>E</sub>X<sub>M</sub>L, S<sup>C</sup>A<sub>L</sub>T<sub>E</sub>X, p<sub>d</sub>f<sub>l</sub>a<sub>t</sub>e<sub>x</sub>).

The p<sub>d</sub>f<sub>l</sub>a<sub>t</sub>e<sub>x</sub>-macros largely do nothing; the S<sup>C</sup>A<sub>L</sub>T<sub>E</sub>X-implementations are pretty clear in what they do, the L<sup>A</sup>T<sub>E</sub>X<sub>M</sub>L-implementations resort to perl bindings.

```

90 \scalatex_if:TF{
91   \cs_new_protected:Nn \stex_annotate:nnn {
92     \__stex_annotate_checkempty:n { #3 }
93     \scalatex_annotate_HTML:nn {
94       property="stex:#1" ~
95       resource="#2"
96     } {
97       \tl_use:N \l__stex_annotate_arg_tl
98     }
99   }
100   \cs_new_protected:Nn \stex_annotate_invisible:n {
101     \__stex_annotate_checkempty:n { #1 }
102     \scalatex_annotate_HTML:nn {
103       stex:visible="false" ~
104       style:display="none"
105     } {
106       \tl_use:N \l__stex_annotate_arg_tl
107     }
108   }
109   \cs_new_protected:Nn \stex_annotate_invisible:nnn {

```

```

110 \__stex_annotate_checkempty:n { #3 }
111 \scalatex_annotate_HTML:nn {
112     property="stex:#1" ~
113     resource="#2" ~
114     stex:visible="false" ~
115     style:display="none"
116 } {
117     \tl_use:N \l__stex_annotate_arg_tl
118 }
119 }
120 \NewDocumentEnvironment{stex_annotate_env} { m m } {
121     \par
122     \scalatex_annotate_HTML_begin:n {
123         property="stex:#1" ~
124         resource="#2"
125     }
126 }{
127     \scalatex_annotate_HTML_end:
128 }
129 }{
130     \latexml_if:TF {
131         \cs_new_protected:Nn \stex_annotate:nnn {
132             \__stex_annotate_checkempty:n { #3 }
133             \mode_if_math:TF {
134                 \cs:w latexml@annotate@math\cs_end:{#1}{#2}{
135                     \tl_use:N \l__stex_annotate_arg_tl
136                 }
137             }{
138                 \cs:w latexml@annotate@text\cs_end:{#1}{#2}{
139                     \tl_use:N \l__stex_annotate_arg_tl
140                 }
141             }
142         }
143         \cs_new_protected:Nn \stex_annotate_invisible:n {
144             \__stex_annotate_checkempty:n { #1 }
145             \mode_if_math:TF {
146                 \cs:w latexml@invisible@math\cs_end:{
147                     \tl_use:N \l__stex_annotate_arg_tl
148                 }
149             } {
150                 \cs:w latexml@invisible@text\cs_end:{
151                     \tl_use:N \l__stex_annotate_arg_tl
152                 }
153             }
154         }
155         \cs_new_protected:Nn \stex_annotate_invisible:nnn {
156             \__stex_annotate_checkempty:n { #3 }
157             \cs:w latexml@annotate@invisible\cs_end:{#1}{#2}{
158                 \tl_use:N \l__stex_annotate_arg_tl
159             }
160         }
161     }
162     \NewDocumentEnvironment{stex_annotate_env} { m m } {
163         \par\begin{latexml@annotateenv}{#1}{#2}

```

```

164     \end{latexml@annotateenv}
165   }
166   }{
167     \cs_new_protected:Nn \stex_annotate:nnn {#3}
168     \cs_new_protected:Nn \stex_annotate_invisible:n {}
169     \cs_new_protected:Nn \stex_annotate_invisible:nnn {}
170     \NewDocumentEnvironment{stex_annotate_env} { m m } {\par}{\}
171   }
172 }

```

(End definition for `\stex_annotate:nnn`, `\stex_annotate_invisible:n`, and `\stex_annotate_invisible:nnn`. These functions are documented on page 8.)

### 4.2.3 Languages

```

173 <@@=stex_language>

```

`\c_stex_languages_prop`  
`\c_stex_language_abbrevs_prop`

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

```

174 \prop_const_from_keyval:Nn \c_stex_languages_prop {
175   en = english ,
176   de = ngerman ,
177   ar = arabic ,
178   bg = bulgarian ,
179   ru = russian ,
180   fi = finnish ,
181   ro = romanian ,
182   tr = turkish ,
183   fr = french
184 }
185
186 \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop {
187   english = en ,
188   ngerman = de ,
189   arabic = ar ,
190   bulgarian = bg ,
191   russian = ru ,
192   finnish = fi ,
193   romanian = ro ,
194   turkish = tr ,
195   french = fr
196 }
197 % todo: chinese simplified (zhs)
198 %       chinese traditional (zht)

```

(End definition for `\c_stex_languages_prop` and `\c_stex_language_abbrevs_prop`. These variables are documented on page 9.)

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

```

199 \clist_if_empty:NF \c_stex_languages_clist {
200   \clist_clear:N \l_tmpa_clist
201   \clist_map_inline:Nn \c_stex_languages_clist {
202     \prop_get:NnNTF \c_stex_languages_prop { #1 } \l_tmpa_str {
203       \clist_put_right:No \l_tmpa_clist \l_tmpa_str
204     } {
205       \msg_set:nnn{stex}{error/unknownlanguage}{
206         Unknown~language~\l_tmpa_str

```

```

207     }
208     \msg_error:nn{stex}{error/unknownlanguage}
209   }
210 }
211 \stex_debug:n {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
212 \RequirePackage[\clist_use:Nn \l_tmpa_clist ,]{babel}
213 }

```

### 4.3 Files, Paths and URIs

```
214 <@@=stex_path>
```

#### 4.3.1 Generic Path Handling

We treat paths as L<sup>A</sup>T<sub>E</sub>X3-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

```

```

215 \cs_new_protected:Nn \stex_path_from_string:Nn {
216   \str_set:Nx \l_tmpa_str { #2 }
217   \str_if_empty:NTF \l_tmpa_str {
218     \seq_clear:N #1
219   }{
220     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
221     \sys_if_platform_windows:T{
222       \seq_clear:N \l_tmpa_tl
223       \seq_map_inline:Nn #1 {
224         \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
225         \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
226       }
227       \seq_set_eq:NN #1 \l_tmpa_tl
228     }
229     \stex_path_canonicalize:N #1
230   }
231 }
232 \cs_generate_variant:Nn \stex_path_from_string:Nn
233 { NV, cn, cV }

```

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

```

\stex_path_to_string:NN
\stex_path_to_string:N

```

```

234 \cs_new_protected:Nn \stex_path_to_string:NN {
235   \exp_args:NNe \str_set:Nn #2 { \seq_use:Nn #1 / }
236 }
237
238 \cs_new:Nn \stex_path_to_string:N {
239   \seq_use:Nn #1 /
240 }

```

(End definition for `\stex_path_to_string:NN` and `\stex_path_to_string:N`. These functions are documented on page 9.)

```

\c__stex_path_dot_str . and ..., respectively.
\c__stex_path_up_str

```

```

241 \str_const:Nn \c__stex_path_dot_str {.}
242 \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.

```

243 \cs_new_protected:Nn \stex_path_canonicalize:N {
244   \seq_if_empty:NF #1 {
245     \seq_clear:N \l_tmpa_seq
246     \seq_get_left:NN #1 \l_tmpa_tl
247     \str_if_empty:NT \l_tmpa_tl {
248       \seq_put_right:Nn \l_tmpa_seq {}
249     }
250     \seq_map_inline:Nn #1 {
251       \str_set:Nn \l_tmpa_tl { ##1 }
252       \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_dot_str {} {
253         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
254           \seq_if_empty:NNTF \l_tmpa_seq {
255             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
256               \c__stex_path_up_str
257             }
258           }{
259             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
260             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
261               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
262                 \c__stex_path_up_str
263               }
264             }{
265               \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
266             }
267           }
268         }{
269           \str_if_empty:NF \l_tmpa_tl {
270             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
271           }
272         }
273       }
274     }
275     \seq_gset_eq:NN #1 \l_tmpa_seq
276   }
277 }

```

(End definition for \stex\_path\_canonicalize:N. This function is documented on page 9.)

**\stex\_path\_if\_absolute\_p:N**

**\stex\_path\_if\_absolute:N $\underline{TF}$**

```

278 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
279   \seq_if_empty:NNTF #1 {
280     \prg_return_false:
281   }{
282     \seq_get_left:NN #1 \l_tmpa_tl
283     \str_if_empty:NNTF \l_tmpa_tl {
284       \prg_return_true:
285     }{
286       \prg_return_false:
287     }
288   }
289 }

```

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

### 4.3.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

290 \str_new:N\l_stex_kpsewhich_return_str
291 \cs_new_protected:Nn \stex_kpsewhich:n {
292   \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
293   \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
294   \tl_trim_spaces:N \l_stex_kpsewhich_return_str
295 }
```

(End definition for `\stex_kpsewhich:n`. This function is documented on page 8.)

We determine the PWD

`\c_stex_pwd_seq`

`\c_stex_pwd_str`

```

296 \sys_if_platform_windows:TF{
297   \stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
298 }{
299   \stex_kpsewhich:n{-var-value~PWD}
300 }
301
302 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
303 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
304 \stex_debug:n {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 9.)

### 4.3.3 File Hooks and Tracking

```

305 <@@=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

```

306 \seq_gclear_new:N\g__stex_files_stack
```

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`

```

307 \stex_path_from_string:Nn \c_stex_mainfile_seq {
308   \c_stex_pwd_str/\jobname.tex
309 }
```

(End definition for `\c_stex_mainfile_seq`. This variable is documented on page 9.)



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

```

310 \seq_gclear_new:N\g_stex_currentfile_seq
311 \AddToHook{file/before}{
312   \stex_path_from_string:Nn\g_stex_currentfile_seq{\CurrentFilePath}
313   \stex_path_if_absolute:N\g_stex_currentfile_seq{
314     \exp_args:NNe\seq_put_right:Nn\g_stex_currentfile_seq{\CurrentFile}
315   }{
316     \stex_path_from_string:Nn\g_stex_currentfile_seq{
317       \c_stex_pwd_str/\CurrentFilePath/\CurrentFile
318     }
319   }
320   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
321   \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
322 }
323 \AddToHook{file/after}{
324   \seq_if_empty:N\g__stex_files_stack{
325     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
326   }
327   \seq_if_empty:N\g__stex_files_stack{
328     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
329   }{
330     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
331     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
332   }
333 }

```

(End definition for `\g_stex_currentfile_seq`. This variable is documented on page 9.)

## 4.4 MathHub Repositories

```

334 <@@=stex_mathhub>

\mathhub
\c_stex_mathhub_seq
\c_stex_mathhub_str
335 \str_if_empty:N\mathhub{
336   \stex_kpsewhich:n{-var-value-MATHHUB}
337   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
338 }
339 \str_if_empty:N\c_stex_mathhub_str{
340   \msg_warning:nn{stex}{warning/nomathhub}
341 }{
342   \stex_debug:n {MathHub:~\str_use:N\c_stex_mathhub_str}
343   \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
344 }
345 }{
346   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
347   \stex_path_if_absolute:N\c_stex_mathhub_seq {
348     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
349       \c_stex_pwd_str/\mathhub
350     }
351   }
352   \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
353   \stex_debug:n {MathHub:~\str_use:N\c_stex_mathhub_str}
354 }

```

(End definition for `\mathhub`, `\c_stex_mathhub_seq`, and `\c_stex_mathhub_str`. These variables are documented on page 10.)

`\_stex_mathhub_do_manifest:n`

```

355 \cs_new_protected:Nn \_stex_mathhub_do_manifest:n {
356   \str_set:Nx \l_tmpa_str { #1 }
357   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
358     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
359     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
360     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
361     \_stex_mathhub_find_manifest:N \l_tmpa_seq
362     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
363       \msg_set:nnn{stex}{error/norepository}{
364         No~archive~#1~found~in~
365         \stex_path_to_string:N \c_stex_mathhub_str
366       }
367       \msg_error:nn{stex}{error/norepository}
368     } {
369       \exp_args:No \_stex_mathhub_parse_manifest:n { \l_tmpa_str }
370     }
371   }
372 }
```

(End definition for `\_stex_mathhub_do_manifest:n`.)

`\l__stex_mathhub_manifest_file_seq`

```

373 \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`:

```

374 \cs_new_protected:Nn \_stex_mathhub_find_manifest:N {
375   \seq_set_eq:NN\l_tmpa_seq #1
376   \bool_set_true:N\l_tmpa_bool
377   \bool_while_do:Nn \l_tmpa_bool {
378     \seq_if_empty:NTF \l_tmpa_seq {
379       \bool_set_false:N\l_tmpa_bool
380     }{
381       \file_if_exist:nTF{
382         \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
383       }{
384         \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
385         \bool_set_false:N\l_tmpa_bool
386       }{
387         \file_if_exist:nTF{
388           \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
389         }{
390           \seq_put_right:Nn\l_tmpa_seq{META-INF}
391           \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
392           \bool_set_false:N\l_tmpa_bool
393         }{
394           \file_if_exist:nTF{
395             \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
```

```

396         }{
397         \seq_put_right:Nn\l_tmpa_seq{meta-inf}
398         \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
399         \bool_set_false:N\l_tmpa_bool
400         }{
401         \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
402         }
403     }
404 }
405 }
406 }
407 \seq_set_eq:NN\l__stex_mathhub_manifest_file_seq\l_tmpa_seq
408 }

```

(End definition for `\_stex_mathhub_find_manifest:N`.)

`\c_stex_mathhub_manifest_ior` File variable used for MANIFEST-files

```

409 \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:

```

410 \cs_new_protected:Nn \_stex_mathhub_parse_manifest:n {
411   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
412   \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
413   \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
414     \str_set:Nn \l_tmpa_str {##1}
415     \exp_args:NNoo \seq_set_split:Nnn
416       \l_tmpb_seq \c_colon_str \l_tmpa_str
417     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
418       \exp_args:NNe \str_set:Nn \l_tmpb_tl {
419         \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
420       }
421       \exp_args:No \str_case:nnTF \l_tmpa_tl {
422         {id} {
423           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
424             { id } \l_tmpb_tl
425         }
426         {narration-base} {
427           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
428             { narr } \l_tmpb_tl
429         }
430         {source-base} {
431           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
432             { ns } \l_tmpb_tl
433         }
434         {ns} {
435           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
436             { ns } \l_tmpb_tl
437         }
438         {dependencies} {
439           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
440             { deps } \l_tmpb_tl
441         }

```

```

442     }{}{}
443   }{}
444 }
445 \ior_close:N \c__stex_mathhub_manifest_ior
446 }

```

(End definition for `\__stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

447 \cs_new_protected:Nn \stex_set_current_repository:n {
448   \stex_require_repository:n { #1 }
449   \prop_set_eq:Nc \l_stex_current_repository_prop {
450     c_stex_mathhub_#1_manifest_prop
451   }
452 }

```

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

`\stex_require_repository:n`

```

453 \cs_new_protected:Nn \stex_require_repository:n {
454   \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
455     \stex_debug:n{Opening~archive:~#1}
456     \__stex_mathhub_do_manifest:n { #1 }
457     \exp_args:Nx \stex_addtosms:n {
458       \prop_const_from_keyval:cn { c_stex_mathhub_#1_manifest_prop } {
459         id = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { id },
460         ns = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { ns },
461         narr = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { narr },
462         deps = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { deps }
463       }
464     }
465   }
466 }

```

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

`\l_stex_current_repository_prop`

Current MathHub repository

```

467 \prop_new:N \l_stex_current_repository_prop
468
469 \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
470 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
471   \stex_debug:n{Not~currently~in~a~MathHub~repository}
472 } {
473   \__stex_mathhub_parse_manifest:n { main }
474   \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
475   \l_tmpa_str
476   \prop_set_eq:cN { c_stex_mathhub_#1_manifest_prop }
477   \c_stex_mathhub_main_manifest_prop
478   \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
479   \stex_debug:n{Current~repository:~
480     \prop_item:Nn \l_stex_current_repository_prop {id}
481   }
482 }

```

(End definition for `\l_stex_current_repository_prop`. This variable is documented on page 10.)

**\libinput**

```
483 \cs_new_protected:Npn \libinput #1 {
484   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
485     \msg_set:nnn{stex}{error/norepository}{
486       \c_backslash_str libinput~needs~to~be~called~in~an~archive
487     }
488     \msg_error:nn{stex}{error/norepository}
489   }
490   \bool_set_false:N \l_tmpa_bool
491   \tl_clear:N \l_tmpa_tl
492   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
493   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
494   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str
495   \seq_pop_left:NNT \l_tmpb_seq \l_tmpb_str {
496     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
497     \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
498       / meta-inf / lib / #1.tex}{
499       \bool_set_true:N \l_tmpa_bool
500       \tl_put_right:Nx \l_tmpa_tl {
501         \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
502           / meta-inf / lib / #1.tex}
503       }
504     }{}
505   }
506   \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
507     / \l_tmpa_str / lib / #1.tex
508   }{
509     \bool_set_true:N \l_tmpa_bool
510     \tl_put_right:Nx \l_tmpa_tl {
511       \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
512         / \l_tmpa_str / lib / #1.tex}
513     }
514   }{}
515   \bool_if:NF \l_tmpa_bool {
516     \msg_set:nnn{stex}{error/nofile}{
517       \c_backslash_str libinput~no~file~#1.tex~found!
518     }
519     \msg_error:nn{stex}{error/nofile}
520   }
521   \l_tmpa_tl
522 }
```

(End definition for \libinput. This function is documented on page 11.)

## 4.5 Module System

```
523 <@@=stex_module>
```

**\l\_stex\_current\_module\_prop**

```
524 \prop_new:N \l_stex_current_module_prop
```

(End definition for \l\_stex\_current\_module\_prop. This variable is documented on page 12.)

**stex\_if\_in\_module\_p:**

**stex\_if\_in\_module:** TF

```

525 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
526   \prop_if_empty:NTF \l_stex_current_module_prop
527   \prg_return_false: \prg_return_true:
528 }

```

(End definition for `stex_if_in_module:TF`. This function is documented on page 12.)

`stex_if_module_exists_p:n`  
`stex_if_module_exists:nTF`

```

529 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
530   \prop_if_exist:cTF { c_stex_module_#1_prop }
531   \prg_return_true: \prg_return_false:
532 }

```

(End definition for `stex_if_module_exists:nTF`. This function is documented on page 12.)

`\stex_add_to_current_module:n`  
`\STEXexport`

```

533 \cs_new_protected:Nn \stex_add_to_current_module:n {
534   \prop_get:NnN \l_stex_current_module_prop { content } \l_tmpa_tl
535   \tl_put_right:Nn \l_tmpa_tl { #1 }
536   \prop_put:Nno \l_stex_current_module_prop { content } { \l_tmpa_tl }
537 }
538 \NewDocumentCommand \STEXexport { m }{
539   \stex_smsmode_set_codes:
540   \stex_add_to_current_module:n { #1 }
541   #1
542 }

```

(End definition for `\stex_add_to_current_module:n` and `\STEXexport`. These functions are documented on page 12.)

`\stex_add_constant_to_current_module:n`

```

543 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
544   \str_set:Nx \l_tmpa_str { #1 }
545   \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
546   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
547   \prop_put:Nno \l_stex_current_module_prop { constants } \l_tmpa_seq
548 }

```

(End definition for `\stex_add_constant_to_current_module:n`. This function is documented on page 12.)

`\stex_add_import_to_current_module:n`

```

549 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
550   \str_set:Nx \l_tmpa_str { #1 }
551   \prop_get:NnN \l_stex_current_module_prop { imports } \l_tmpa_seq
552   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
553   \prop_put:Nno \l_stex_current_module_prop { imports } \l_tmpa_seq
554 }

```

(End definition for `\stex_add_import_to_current_module:n`. This function is documented on page 12.)

`\stex_modules_compute_namespace:nN` stores its return values in:

```

\l_stex_modules_ns_str 555 \str_new:N \l_stex_modules_ns_str

```

```

556 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
557   \str_set:Nx \l_tmpa_str { #1 }
558   \seq_set_eq:NN \l_tmpa_seq #2
559   % split off file extension
560   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
561   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
562   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
563   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
564
565   \bool_set_true:N \l_tmpa_bool
566   \bool_while_do:Nn \l_tmpa_bool {
567     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
568     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
569       {source} { \bool_set_false:N \l_tmpa_bool }
570     }{}{
571       \seq_if_empty:NT \l_tmpa_seq {
572         \bool_set_false:N \l_tmpa_bool
573       }
574     }
575   }
576
577   \seq_if_empty:NTF \l_tmpa_seq {
578     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
579   }{
580     \str_set:Nx \l_stex_modules_ns_str {
581       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
582     }
583   }
584 }

```

(End definition for `\stex_modules_compute_namespace:nN` and `\l_stex_modules_ns_str`. These functions are documented on page 13.)

`\stex_modules_current_namespace:`

```

585 \cs_new_protected:Nn \stex_modules_current_namespace: {
586   \prop_get:NnNTF \l_stex_current_repository_prop { ns } \l_tmpa_str {
587     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
588   }{
589     % split off file extension
590     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
591     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
592     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
593     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
594     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
595     \str_set:Nx \l_stex_modules_ns_str {
596       file:/\stex_path_to_string:N \l_tmpa_seq
597     }
598   }
599 }

```

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

#### 4.5.1 The module environment

`\l_stex_all_modules_seq` Stores all available modules

```
600 \seq_new:N \l_stex_all_modules_seq
```

(End definition for `\l_stex_all_modules_seq`. This variable is documented on page 14.)

```
\STEXModule
\stex_invoke_module:n
```

```
601 \NewDocumentCommand \STEXModule { m } {
602   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
603   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
604   \tl_set:Nn \l_tmpa_tl {
605     \msg_set:nnn{stex}{error/unknownmodule}{
606       No~module~#1~found!
607     }
608     \msg_error:nn{stex}{error/unknownmodule}
609   }
610   \seq_map_inline:Nn \l_stex_all_modules_seq {
611     \str_set:Nn \l_tmpb_str { ##1 }
612     \str_if_eq:eeT { \l_tmpa_str } {
613       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
614     } {
615       \seq_map_break:n {
616         \tl_set:Nn \l_tmpa_tl {
617           \stex_invoke_module:n { ##1 }
618         }
619       }
620     }
621   }
622   \l_tmpa_tl
623 }
624
625 \cs_new_protected:Nn \stex_invoke_module:n {
626   \stex_debug:n{Invoking~module~#1}
627   \peek_charcode_remove:NTF ! {
628     \__stex_module_invoke_uri:nN { #1 }
629   } {
630     \peek_charcode_remove:NTF ? {
631       \__stex_module_invoke_symbol:nn { #1 }
632     } {
633       \msg_set:nnn{stex}{error/syntax}{
634         Syntax~error:~?~or~!~expected~after~
635         \c_backslash_str STEXModule{#1}
636       }
637       \msg_error:nn{stex}{error/syntax}
638     }
639   }
640 }
641
642 \cs_new_protected:Nn \__stex_module_invoke_uri:nN {
643   \str_set:Nn #2 { #1 }
644 }
645
646 \cs_new_protected:Nn \__stex_module_invoke_symbol:nn {
647   \stex_invoke_symbol:n{#1?#2}
648 }
```



(End definition for `\STEXModule` and `\stex_invoke_module:n`. These functions are documented on page 14.)

module module arguments:

```

649 \keys_define:nn { stex / module } {
650   title      .tl_set_x:N = \l_stex_module_title_str ,
651   ns         .tl_set_x:N = \l_stex_module_ns_str ,
652   lang       .tl_set_x:N = \l_stex_module_lang_str ,
653   sig        .tl_set_x:N = \l_stex_module_sig_str ,
654   creators   .tl_set_x:N = \l_stex_module_creators_str ,
655   contributors .tl_set_x:N = \l_stex_module_contributors_str ,
656   meta       .tl_set_x:N = \l_stex_module_meta_str
657 }
658
659 % module parameters here? In the body?
660
661 \cs_new_protected:Nn \__stex_module_args:n {
662   \str_clear:N \l_stex_module_title_str
663   \str_clear:N \l_stex_module_ns_str
664   \str_clear:N \l_stex_module_lang_str
665   \str_clear:N \l_stex_module_sig_str
666   \str_clear:N \l_stex_module_creators_str
667   \str_clear:N \l_stex_module_contributors_str
668   \str_clear:N \l_stex_module_meta_str
669   \keys_set:nn { stex / module } { #1 }
670   \exp_args:NNo \str_set:Nn \l_stex_module_title_str
671     \l_stex_module_title_str
672   \exp_args:NNo \str_set:Nn \l_stex_module_ns_str
673     \l_stex_module_ns_str
674   \exp_args:NNo \str_set:Nn \l_stex_module_lang_str
675     \l_stex_module_lang_str
676   \exp_args:NNo \str_set:Nn \l_stex_module_sig_str
677     \l_stex_module_sig_str
678   \exp_args:NNo \str_set:Nn \l_stex_module_meta_str
679     \l_stex_module_meta_str
680   \exp_args:NNo \str_set:Nn \l_stex_module_creators_str
681     \l_stex_module_creators_str
682   \exp_args:NNo \str_set:Nn \l_stex_module_contributors_str
683     \l_stex_module_contributors_str
684 }

```

`\__stex_module_begin_module:` implements `\begin{module}`

```

685 \cs_new_protected:Nn \__stex_module_begin_module: {
686   % Nested module?
687   \stex_if_in_module:TF {
688     % Nested module
689     \prop_get:NnN \l_stex_current_module_prop
690       { ns } \l_stex_module_ns_str
691     \str_set:Nx \l_stex_module_name_str {
692       \prop_item:Nn \l_stex_current_module_prop
693         { name } / \l_stex_module_name_str
694     }
695   }{
696     % not nested:

```

```

697 \str_if_empty:NT \l_stex_module_ns_str {
698   \stex_modules_current_namespace:
699   \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
700   \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
701     / {\l_stex_module_ns_str}
702   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
703   \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
704     \str_set:Nx \l_stex_module_ns_str {
705       \stex_path_to_string:N \l_tmpa_seq
706     }
707   }
708 }
709 }
710
711 % language
712 \str_if_empty:NT \l_stex_module_lang_str {
713   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
714   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
715   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
716   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
717   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
718     \stex_debug:n {Language~\l_stex_module_lang_str~
719       inferred~from~file~name}
720     \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
721   }
722 }
723
724 \str_if_empty:NF \l_stex_module_lang_str {
725   \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
726   \l_tmpa_str {
727     \ltx@ifpackageloaded{babel}{
728       \exp_args:Nx \selectlanguage { \l_tmpa_str }
729     }{}
730   } {
731     \msg_set:nnn{stex}{error/unknownlanguage}{
732       Unknown~language~\l_tmpa_str
733     }
734     \msg_error:nn{stex}{error/unknownlanguage}
735   }
736 }
737
738 % signature
739 \str_if_empty:NTF \l_stex_module_sig_str {
740   \str_clear:N \l_tmpa_str
741   \seq_clear:N \l_tmpa_seq
742   \tl_clear:N \l_tmpa_tl
743   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
744     name      = \l_stex_module_name_str ,
745     ns        = \l_stex_module_ns_str ,
746     imports   = \exp_not:o { \l_tmpa_seq } ,
747     constants = \exp_not:o { \l_tmpa_seq } ,
748     content   = \exp_not:o { \l_tmpa_tl } ,
749     file      = \exp_not:o { \g_stex_currentfile_seq } ,
750     lang      = \l_stex_module_lang_str ,

```

```

751     sig      = \l_stex_module_sig_str ,
752     meta     = \l_stex_module_meta_str
753   }
754 }{
755   \str_if_empty:NT \l_stex_module_lang_str {
756     \msg_set:nnn{stex}{error/siglanguage}{
757       Module~\l_stex_module_ns_str?\l_stex_module_name_str~
758       declares~signature~\l_stex_module_sig_str,~but~does~not~
759       declare~its~language
760     }
761     \msg_error:nn{stex}{error/siglanguage}
762   }
763
764   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
765   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
766   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
767   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
768   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
769   \str_set:Nx \l_tmpa_str {
770     \stex_path_to_string:N \l_tmpa_seq /
771     \l_tmpa_str . \l_stex_module_sig_str .tex
772   }
773   \IfFileExists \l_tmpa_str {
774     \exp_args:No \stex_in_smsmode:nn { \l_tmpa_str } {
775       \seq_clear:N \l_stex_all_modules_seq
776       \prop_clear:N \l_stex_current_module_prop
777       \stex_debug:n{Loading~signature~\l_tmpa_str}
778       \input { \l_tmpa_str }
779     }
780   }{
781     \msg_set:nnn{stex}{error/modulemissing}{
782       No~file~for~signature~module~\l_tmpa_str~found
783     }
784     \msg_error:nn{stex}{error/modulemissing}
785   }
786   \stex_activate_module:n {
787     \l_stex_module_ns_str ? \l_stex_module_name_str
788   }
789   \prop_set_eq:Nc \l_stex_current_module_prop {
790     c_stex_module_
791     \l_stex_module_ns_str ?
792     \l_stex_module_name_str
793     _prop
794   }
795 }
796
797 % metatheory
798 \str_if_empty:NT \l_stex_module_meta_str {
799   \str_set:Nx \l_stex_module_meta_str {
800     \c_stex_metatheory_ns_str ? Metatheory
801   }
802 }
803
804

```

```

805 \stex_debug:n{
806   New~module:\\
807   Namespace::~~\l_stex_module_ns_str\\
808   Name::~~\l_stex_module_name_str\\
809   Language::~~\l_stex_module_lang_str\\
810   Signature::~~\l_stex_module_sig_str\\
811   Metatheory::~~\l_stex_module_meta_str\\
812   File::~~\stex_path_to_string:N \g_stex_currentfile_seq
813 }
814
815 \seq_put_right:Nx \l_stex_all_modules_seq {
816   \l_stex_module_ns_str ? \l_stex_module_name_str
817 }
818
819 \seq_gput_right:Nx \g_stex_modules_in_file_seq
820   { \l_stex_module_ns_str ? \l_stex_module_name_str }
821
822 \stex_if_smsmode:TF {
823   \stex_smsmode_set_codes:
824 } {
825   \begin{stex_annotate_env} {theory} {
826     \l_stex_module_ns_str ? \l_stex_module_name_str
827   }
828
829   \stex_annotate_invisible:nnn{header}{ } {
830     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{ }
831     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{ }
832     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
833       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{ }
834     }
835   }
836 }
837
838 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
839   \exp_args:Nx \STEXexport{
840     \stex_activate_module:n { \l_stex_module_meta_str }
841   }
842 }
843 % TODO: Inherit metatheory for nested modules?
844 }
845 \iffalse \end{stex_annotate_env} \fi % make syntax highlighting work again

```

(End definition for \\_stex\_module\_begin\_module:.)

\\_stex\_module\_end\_module: implements \end{module}

```

846 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
847 \cs_new_protected:Nn \_stex_module_end_module: {
848   \str_set:Nx \l_tmpa_str {
849     c_stex_module_
850     \prop_item:Nn \l_stex_current_module_prop { ns } ?
851     \prop_item:Nn \l_stex_current_module_prop { name }
852     _prop
853   }
854   %^^A \prop_new:c { \l_tmpa_str }

```

```

855 \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
856 \stex_debug:n{Closing~module~\prop_item:Nn \l_stex_current_module_prop { name }}
857 \stex_if_smsmode:TF {
858   \exp_args:Nx \stex_addtosms:n {
859     \prop_gset_from_keyval:cn {
860       c_stex_module_
861       \prop_item:Nn \l_stex_current_module_prop { ns } ?
862       \prop_item:Nn \l_stex_current_module_prop { name }
863       _prop
864     } {
865       name      = \prop_item:cn { \l_tmpa_str } { name } ,
866       ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
867       imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
868       constants = \prop_item:cn { \l_tmpa_str } { constants } ,
869       content   = \prop_item:cn { \l_tmpa_str } { content } ,
870       file      = \prop_item:cn { \l_tmpa_str } { file } ,
871       lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
872       sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
873       meta      = \prop_item:cn { \l_tmpa_str } { meta }
874     }
875   }
876 }{
877   \end{stex_annotate_env}
878 }
879 }

```

(End definition for `\_stex_module_end_module:.`)

**@module** The core environment, with no header

```

880 \NewDocumentEnvironment { @module } { 0{} m } {
881   \str_set:Nx \l_stex_module_name_str { #2 }
882   \par
883   \_stex_module_args:n { #1 }
884   \_stex_module_begin_module:
885 } {
886   \_stex_module_end_module:
887 }

```

**\stex\_modules\_heading:** Code for document headers

```

888 \cs_if_exist:NTF \thesection {
889   \newcounter{module}[section]
890 }{
891   \newcounter{module}
892 }
893
894 \bool_if:NT \c_stex_showmods_bool {
895   \latexml_if:F { \RequirePackage{mdframed} }
896 }
897
898 \cs_new_protected:Nn \stex_modules_heading: {
899   \stepcounter{module}
900   \par
901   \bool_if:NT \c_stex_showmods_bool {
902     \noindent{\textbf{Module}} ~

```

```

903     \cs_if_exist:NT \thesection {\thesection.}
904     \themodule ~ [\l_stex_module_name_str]
905   }
906   % TODO references
907   % \sref@label@id{Module \thesection.\themodule [\module@name]}\%
908   \str_if_empty:NTF \l_stex_module_title_str {
909     }{
910     \quad(\l_stex_module_title_str)\hfill
911   }\par
912 }
913 }

```

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

Finally:

```

914 \NewDocumentEnvironment { module } { 0 } { m } {
915   \bool_if:NT \c_stex_showmods_bool {
916     \begin{mdframed}
917   }
918   \begin{@module} [#1] {#2}
919   \stex_modules_heading:
920 }{
921   \end{@module}
922   \bool_if:NT \c_stex_showmods_bool {
923     \end{mdframed}
924   }
925 }

```

#### 4.5.2 SMS Mode

```

926 <@@=stex_smsmode>

```

```

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq
927 \tl_new:N \g_stex_smsmode_allowedmacros_tl
928 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
929 \seq_new:N \g_stex_smsmode_allowedenvs_seq
930
931 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
932   \makeatletter
933   \makeatother
934   \ExplSyntaxOn
935   \ExplSyntaxOff
936 }
937
938 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
939   \symdef
940   \abbrdef
941   \importmodule
942   \notation
943   \symdecl
944   \STEXexport
945 }
946
947 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
948   \tl_to_str:n {

```

```

949     module,
950     @module
951   }
952 }

```

(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 15.)

`\stex_if_smsmode_p:`

```

\stex_if_smsmode:TF
953 \bool_new:N \g__stex_smsmode_bool
954 \bool_set_false:N \g__stex_smsmode_bool
955 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
956   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
957 }

```

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

`\_stex_smsmode_if_catcodes_p:`

```

\_stex_smsmode_if_catcodes:TF
958 \bool_new:N \g__stex_smsmode_catcode_bool
959 \bool_set_false:N \g__stex_smsmode_catcode_bool
960 \prg_new_conditional:Nnn \_stex_smsmode_if_catcodes: { p, T, F, TF } {
961   \bool_if:NTF \g__stex_smsmode_catcode_bool
962     \prg_return_true: \prg_return_false:
963 }

```

(End definition for `\_stex_smsmode_if_catcodes:TF`.)

`\stex_smsmode_set_codes:`

```

964 \cs_new_protected:Nn \stex_smsmode_set_codes: {
965   \stex_if_smsmode:T {
966     \_stex_smsmode_if_catcodes:F {
967       \bool_gset_true:N \g__stex_smsmode_catcode_bool
968       \exp_after:wN \char_gset_active_eq:NN
969         \c_backslash_str \_stex_smsmode_cs:
970       \tex_global:D \char_set_catcode_active:N \
971       \tex_global:D \char_set_catcode_other:N $
972       \tex_global:D \char_set_catcode_other:N ^
973       \tex_global:D \char_set_catcode_other:N _
974       \tex_global:D \char_set_catcode_other:N &
975       \tex_global:D \char_set_catcode_other:N ##
976     }
977   }
978 } \iffalse $ \fi % to make syntax highlighting work again

```

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

`\_stex_smsmode_unset_codes:` Sets category code scheme back from the one used in SMS mode.

```

979 \cs_new_protected:Nn \_stex_smsmode_unset_codes: {
980   \_stex_smsmode_if_catcodes:T {
981     \bool_gset_false:N \g__stex_smsmode_catcode_bool
982     \exp_after:wN \tex_global:D \exp_after:wN
983       \char_set_catcode_escape:N \c_backslash_str
984     \tex_global:D \char_set_catcode_math_toggle:N $
985     \tex_global:D \char_set_catcode_math_superscript:N ^
986     \tex_global:D \char_set_catcode_math_subscript:N _

```

```

987 \tex_global:D \char_set_catcode_alignment:N &
988 \tex_global:D \char_set_catcode_parameter:N ##
989 }
990 } \iffalse $ \fi % to make syntax highlighting work again

```

(End definition for `\_stex_smsmode_unset_codes:`.)

`\stex_in_smsmode:nn`

```

991 \cs_new_protected:Nn \stex_in_smsmode:nn {
992   \vbox_set:Nn \l_tmpa_box {
993     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
994     \bool_gset_true:N \g__stex_smsmode_bool
995     \stex_smsmode_set_codes:
996     #2
997     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
998     \stex_if_smsmode:F {
999       \__stex_smsmode_unset_codes:
1000     }
1001   }
1002   \box_clear:N \l_tmpa_box
1003 }

```

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

`\_stex_smsmode_cs:` is executed on encountering `\` in smsmode. It checks whether the corresponding command is allowed and executes or ignores it accordingly:

```

1004 \cs_new_protected:Nn \_stex_smsmode_cs: {
1005   \str_clear:N \l_tmpa_str
1006   \peek_analysis_map_inline:n {
1007     % #1: token (one expansion)
1008     % #2: charcode
1009     % #3 catcode
1010     \token_if_eq_charcode:NNTF ##3 B {
1011       % token is a letter
1012       \exp_args:NNo \str_put_right:Nn \l_tmpa_str { ##1 }
1013     } {
1014       \str_if_empty:NNTF \l_tmpa_str {
1015         % we don't allow (or need) single non-letter CSs
1016         % for now
1017         \peek_analysis_map_break:
1018       }{
1019         \str_if_eq:onTF \l_tmpa_str { begin } {
1020           \peek_analysis_map_break:n {
1021             \exp_after:wN \_stex_smsmode_checkbegin:n ##1
1022           }
1023         } {
1024           \str_if_eq:onTF \l_tmpa_str { end } {
1025             \peek_analysis_map_break:n {
1026               \exp_after:wN \_stex_smsmode_checkend:n ##1
1027             }
1028           } {
1029             \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
1030             \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1031             \g_stex_smsmode_allowedmacros_tl

```



```

1032         { \use:c{\l_tmpa_str} } {
1033         \stex_debug:n{Executing~1:~\l_tmpa_str}
1034         \peek_analysis_map_break:n {
1035             \exp_after:wN \l_tmpa_tl ##1
1036         }
1037     } {
1038         \exp_args:NNo \exp_args:NNo \tl_if_in:NnTF
1039         \g_stex_smsmode_allowedmacros_escape_tl
1040         { \use:c{\l_tmpa_str} } {
1041             \stex_debug:n{Executing~2:~\l_tmpa_str}
1042             % TODO \__stex_smsmode_rescan_cs:
1043             % \exp_after:wN \exp_after:wN \exp_after:wN
1044             % \token_if_eq_charcode:NNTF \exp_after:wN \c_backslash_str ##1 {
1045             %     \peek_analysis_map_break:n {
1046             %         \__stex_smsmode_unset_codes:
1047             %         \__stex_smsmode_rescan_cs:
1048             %     }
1049             % } {
1050             %     \peek_analysis_map_break:n {
1051             %         \__stex_smsmode_unset_codes:
1052             %         \exp_after:wN \l_tmpa_tl ##1
1053             %     }
1054             % }
1055             } {
1056                 \peek_analysis_map_break:n { ##1 }
1057             }
1058         }
1059     }
1060 }
1061 }
1062 }
1063 }
1064 }

```

(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:

```

1065 \cs_new_protected:Nn \__stex_smsmode_rescan_cs: {
1066     \str_clear:N \l_tmpb_str
1067     \peek_analysis_map_inline:n {
1068         \token_if_eq_charcode:NNTF ##3 B {
1069             % token is a letter
1070             \exp_args:NNo \str_put_right:Nn \l_tmpb_str { ##1 }
1071         } {
1072             \peek_analysis_map_break:n {
1073                 \exp_after:wN \use:c \exp_after:wN {
1074                     \exp_after:wN \l_tmpa_str\exp_after:wN
1075                 } \use:c { \l_tmpb_str \exp_after:wN } ##1
1076             }
1077         }
1078     }
1079 }

```

(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.

```

1080 \cs_new_protected:Nn \__stex_smsmode_checkbegin:n {
1081   \str_set:Nn \l_tmpa_str { #1 }
1082   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1083     \__stex_smsmode_unset_codes:
1084     \begin{#1}
1085   }
1086 }
```

(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.

```

1087 \cs_new_protected:Nn \__stex_smsmode_checkend:n {
1088   \str_set:Nn \l_tmpa_str { #1 }
1089   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1090     \end{#1}
1091   }
1092 }
```

(End definition for `\__stex_smsmode_checkend:n`.)

### 4.5.3 Inheritance

```

1093 <@@=stex_importmodule>
```

`\stex_import_module_uri:nn`

```

1094 \cs_new_protected:Nn \stex_import_module_uri:nn {
1095   \str_set:Nx \l__stex_importmodule_archive_str { #1 }
1096   \str_set:Nx \l__stex_importmodule_path_str { #2 }
1097   \str_if_empty:NT \l__stex_importmodule_archive_str {
1098     \prop_if_empty:NF \l_stex_current_repository_prop {
1099       \prop_get:NnN \l_stex_current_repository_prop { id } \l__stex_importmodule_archive_str
1100     }
1101   }
1102
1103   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l__stex_importmodule_path_str }
1104   \seq_pop_right:NN \l_tmpb_seq \l__stex_importmodule_name_str
1105   \str_set:Nx \l__stex_importmodule_path_str { \seq_use:Nn \l_tmpb_seq ? }
1106
1107   \str_if_empty:NTF \l__stex_importmodule_archive_str {
1108     \stex_modules_current_namespace:
1109     \str_if_empty:NF \l__stex_importmodule_path_str {
1110       \str_set:Nx \l_stex_module_ns_str {
1111         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1112       }
1113     }
1114   }{
1115     \stex_require_repository:n \l__stex_importmodule_archive_str
1116     \prop_get:cnN { c_stex_mathhub \l__stex_importmodule_archive_str _manifest_prop } { ns }
1117     \l_stex_module_ns_str
1118     \str_if_empty:NF \l__stex_importmodule_path_str {
1119       \str_set:Nx \l_stex_module_ns_str {
1120         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1121       }
1122     }
1123   }
```

```

1123 }
1124 }

```

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

```

\l__stex_importmodule_name_str Store the return values of \stex_import_module_uri:nn.
\l__stex_importmodule_archive_str 1125 \str_new:N \l__stex_importmodule_name_str
\l__stex_importmodule_path_str 1126 \str_new:N \l__stex_importmodule_archive_str
\l__stex_importmodule_file_str 1127 \str_new:N \l__stex_importmodule_path_str
1128 \str_new:N \g__stex_importmodule_file_str

```

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

```

\stex_import_require_module:nnnnn {<ns>} {<archive-ID>} {<path>} {<name>}
1129 \cs_new_protected:Nn \stex_import_require_module:nnnnn {
1130   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1131     % \stex_debug:n{Arguments: #1, #2, #3, #4}
1132
1133     % archive
1134     \str_set:Nx \l_tmpa_str { #2 }
1135     \str_if_empty:NTF \l_tmpa_str {
1136       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1137     } {
1138       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1139       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1140       \seq_put_right:Nn \l_tmpa_seq { source }
1141     }
1142
1143     % path
1144     \str_set:Nx \l_tmpb_str { #3 }
1145     \str_if_empty:NTF \l_tmpb_str {
1146       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1147
1148       \ltx@ifpackageloaded{babel} {
1149         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1150           { \language } \l_tmpb_str {
1151           \msg_set:nnn{stex}{error/unknownlanguage}{
1152             Unknown~language~\language
1153           }
1154           \msg_error:nn{stex}{error/unknownlanguage}
1155         }
1156       } {
1157         \str_clear:N \l_tmpb_str
1158       }
1159
1160       \stex_debug:n{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1161       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1162         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1163       }{
1164         \stex_debug:n{Checking~\l_tmpa_str.tex}
1165         \IfFileExists{ \l_tmpa_str.tex }{
1166           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1167         }{
1168           % try english as default

```

```

1169     \stex_debug:n{Checking~\l_tmpa_str.en.tex}
1170     \IfFileExists{ \l_tmpa_str.en.tex }{
1171         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1172     }{
1173         \msg_set:nnn{stex}{error/modulemissing}{
1174             No~file~for~module~#1?#4~found
1175         }
1176         \msg_error:nn{stex}{error/modulemissing}
1177     }
1178 }
1179 }
1180
1181 } {
1182     \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1183     \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1184
1185     \ltx@ifpackageloaded{babel} {
1186         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1187             { \language } \l_tmpb_str {
1188             \msg_set:nnn{stex}{error/unknownlanguage}{
1189                 Unknown~language~\language
1190             }
1191             \msg_error:nn{stex}{error/unknownlanguage}
1192         }
1193     } {
1194         \str_clear:N \l_tmpb_str
1195     }
1196
1197     \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1198
1199     \stex_debug:n{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1200     \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1201         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1202     }{
1203         \stex_debug:n{Checking~\l_tmpa_str/#4.tex}
1204         \IfFileExists{ \l_tmpa_str/#4.tex }{
1205             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1206         }{
1207             % try english as default
1208             \stex_debug:n{Checking~\l_tmpa_str/#4.en.tex}
1209             \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1210                 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1211             }{
1212                 \stex_debug:n{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1213                 \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1214                     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1215                 }{
1216                     \stex_debug:n{Checking~\l_tmpa_str.tex}
1217                     \IfFileExists{ \l_tmpa_str.tex }{
1218                         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1219                     }{
1220                         % try english as default
1221                         \stex_debug:n{Checking~\l_tmpa_str.en.tex}
1222                         \IfFileExists{ \l_tmpa_str.en.tex }{

```

```

1223         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1224     }{
1225         \msg_set:nnn{stex}{error/modulemissing}{
1226             No~file~for~module~#1?#4~found
1227         }
1228         \msg_error:nn{stex}{error/modulemissing}
1229     }
1230 }
1231 }
1232 }
1233 }
1234 }
1235 }
1236
1237 \seq_set_eq:NN \l_tmpa_seq \g_stex_modules_in_file_seq
1238 \seq_clear:N \g_stex_modules_in_file_seq
1239 % \exp_args:Nnx \use:nn {
1240     \exp_args:No \stex_in_smsmode:nn { \g__stex_importmodule_file_str } {
1241         \seq_clear:N \l_stex_all_modules_seq
1242         \prop_clear:N \l_stex_current_module_prop
1243         \str_set:Nx \l_tmpb_str { #2 }
1244         \str_if_empty:NF \l_tmpb_str {
1245             \stex_set_current_repository:n { #2 }
1246         }
1247         \stex_debug:n{Loading~\g__stex_importmodule_file_str}
1248         \input { \g__stex_importmodule_file_str }
1249     }
1250 % }{
1251
1252 % }
1253 \prop_gput:Noo \g_stex_module_files_prop
1254 \g__stex_importmodule_file_str \g_stex_modules_in_file_seq
1255 \seq_set_eq:NN \g_stex_modules_in_file_seq \l_tmpa_seq
1256
1257 \stex_if_module_exists:nF { #1 ? #4 } {
1258     \msg_set:nnn{stex}{error/modulemissing}{
1259         Module~#1?#4~not~found~in~file~\g__stex_importmodule_file_str
1260     }
1261     \msg_error:nn{stex}{error/modulemissing}
1262 }
1263 }
1264 \stex_activate_module:n { #1 ? #4 }
1265 }

```

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

`\stex_activate_module:n`

```

1266 \cs_new_protected:Nn \stex_activate_module:n {
1267     \stex_debug:n{Activating~module~#1}
1268     \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1269         \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1270         \prop_item:cn { c_stex_module_#1_prop } { content }
1271     }
1272 }

```

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

### `\importmodule`

```

1273 \NewDocumentCommand \importmodule { 0{ } m } {
1274   \stex_import_module_uri:nn { #1 } { #2 }
1275   \stex_debug:n{Importing~module:~
1276     \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1277   }
1278   \stex_if_smsmode:F {
1279     \stex_import_require_module:nnnn
1280     { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1281     { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1282     \stex_annotate_invisible:nnn
1283     {import} { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } {}
1284   }
1285   \exp_args:Nx \stex_add_to_current_module:n {
1286     \stex_import_require_module:nnnn
1287     { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1288     { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1289   }
1290   \exp_args:Nx \stex_add_import_to_current_module:n {
1291     \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1292   }
1293   \stex_smsmode_set_codes:
1294 }

```

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

### `\usemodule`

```

1295 \NewDocumentCommand \usemodule { 0{ } m } {
1296   \stex_if_smsmode:F {
1297     \stex_import_module_uri:nn { #1 } { #2 }
1298     \stex_import_require_module:nnnn
1299     { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1300     { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1301     \stex_annotate_invisible:nnn
1302     {usemodule} { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } {}
1303   }
1304   \stex_smsmode_set_codes:
1305 }

```

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

### `\g_stex_modules_in_file_seq` `\g_stex_module_files_prop`

```

1306 \seq_new:N \g_stex_modules_in_file_seq
1307 \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 19.)

## 4.6 Symbol Declarations

1308 `<@@=stex_symdecl>`

`\l_stex_all_symbols_seq` Stores all available symbols

1309 `\seq_new:N \l_stex_all_symbols_seq`

(End definition for `\l_stex_all_symbols_seq`. This variable is documented on page 21.)

`\STEXsymbol`

```
1310 \NewDocumentCommand \STEXsymbol { m } {
1311   \stex_get_symbol:n { #1 }
1312   \exp_args:No
1313   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1314 }
```

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

symdecl arguments:

```
1315 \keys_define:nn { stex / symdecl } {
1316   name      .tl_set:x:N = \l_stex_symdecl_name_str ,
1317   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1318   args      .tl_set:x:N = \l_stex_symdecl_args_str ,
1319   type      .tl_set:N   = \l_stex_symdecl_type_tl ,
1320   align     .tl_set:N   = \l_stex_symdecl_align_str , % TODO(?)
1321   gfc       .tl_set:N   = \l_stex_symdecl_gfc_str , % TODO(?)
1322   specializes .tl_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
1323 }
1324
1325 \bool_new:N \l_stex_symdecl_make_macro_bool
1326
1327 \cs_new_protected:Nn \__stex_symdecl_args:n {
1328   \str_clear:N \l_stex_symdecl_name_str
1329   \str_clear:N \l_stex_symdecl_args_str
1330   \bool_set_false:N \l_stex_symdecl_local_bool
1331   \tl_clear:N \l_stex_symdecl_type_tl
1332
1333   \keys_set:nn { stex / symdecl } { #1 }
1334
1335   \exp_args:NNo \str_set:Nn \l_stex_symdecl_name_str
1336     \l_stex_symdecl_name_str
1337   \exp_args:NNo \str_set:Nn \l_stex_symdecl_args_str
1338     \l_stex_symdecl_args_str
1339 }
```

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

```
1340 \cs_new_protected:Npn \symdecl {
1341   \peek_charcode_remove:NTF * {
1342     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1343     \__stex_symdecl_:
1344   } {
1345     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1346     \__stex_symdecl_:
1347   }
```

```

1348 }
1349
1350 \NewDocumentCommand \stex_symdecl_: { 0{ } m } {
1351   \stex_symdecl_args:n { #1 }
1352   \tl_clear:N \l_stex_symdecl_definiens_tl
1353   \stex_symdecl_do:n { #2 }
1354   \stex_smsmode_set_codes:
1355 }

```

(End definition for `\symdecl`. This function is documented on page 20.)

`\abbrdef`

```

1356 \NewDocumentCommand \abbrdef { 0{ } m m } {
1357   \stex_symdecl_args:n { #1 }
1358   \tl_set:Nn \l_stex_symdecl_definiens_tl { #3 }
1359   \bool_set_true:N \l_stex_symdecl_make_macro_bool
1360   \stex_symdecl_do:n { #2 }
1361 }

```

(End definition for `\abbrdef`. This function is documented on page 20.)

`\stex_symdecl_do:n`

```

1362 \cs_new_protected:Nn \stex_symdecl_do:n {
1363   \stex_if_in_module:F {
1364     % TODO throw error? some default namespace?
1365   }
1366
1367   \str_if_empty:NT \l_stex_symdecl_name_str {
1368     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1369   }
1370
1371   \prop_if_exist:cT { g_stex_symdecl_
1372     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1373     \prop_item:Nn \l_stex_current_module_prop {name} ?
1374     \l_stex_symdecl_name_str
1375     _prop
1376   }{
1377     % TODO throw error (beware of circular dependencies)
1378   }
1379
1380   \prop_clear:N \l_tmpa_prop
1381   \prop_put:Nnx \l_tmpa_prop { module } {
1382     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1383     \prop_item:Nn \l_stex_current_module_prop {name}
1384   }
1385   \seq_clear:N \l_tmpa_seq
1386   \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
1387   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1388   \prop_put:Nno \l_tmpa_prop { local } \l_stex_symdecl_local_bool
1389   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1390
1391   \exp_args:No \stex_add_constant_to_current_module:n {
1392     \l_stex_symdecl_name_str
1393   }
1394 }

```



```

1395 % arity/args
1396 \int_zero:N \l_tmpb_int
1397
1398 \bool_set_true:N \l_tmpa_bool
1399 \str_map_inline:Nn \l_stex_symdecl_args_str {
1400   \token_case_meaning:NnF ##1 {
1401     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1402     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1403     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1404     {\tl_to_str:n a} {
1405       \bool_set_false:N \l_tmpa_bool
1406       \int_incr:N \l_tmpb_int
1407     }
1408     {\tl_to_str:n B} {
1409       \bool_set_false:N \l_tmpa_bool
1410       \int_incr:N \l_tmpb_int
1411     }
1412   }{
1413     \msg_set:nnn{stex}{error/wrongargs}{
1414       args~value~in~symbol~declaration~for~
1415       \prop_item:Nn \l_stex_current_module_prop {ns} ?
1416       \prop_item:Nn \l_stex_current_module_prop {name} ?
1417       \l_stex_symdecl_name_str ~
1418       needs~to~be~
1419       i,~a,~b~or~B,~but~##1~given
1420     }
1421     \msg_error:nn{stex}{error/wrongargs}
1422   }
1423 }
1424 \bool_if:NTF \l_tmpa_bool {
1425   % possibly numeric
1426   \str_if_empty:NTF \l_stex_symdecl_args_str {
1427     \prop_put:Nnn \l_tmpa_prop { args } {}
1428     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1429   }{
1430     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1431     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1432     \str_clear:N \l_tmpa_str
1433     \int_step_inline:nn \l_tmpa_int {
1434       \str_put_right:Nn \l_tmpa_str i
1435     }
1436     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1437   }
1438 } {
1439   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1440   \prop_put:Nnx \l_tmpa_prop { arity }
1441   { \str_count:N \l_stex_symdecl_args_str }
1442 }
1443 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
1444
1445 % semantic macro
1446
1447
1448 \bool_if:NT \l_stex_symdecl_make_macro_bool {

```

```

1449 \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1450   \prop_item:Nn \l_tmpa_prop { module } ?
1451   \prop_item:Nn \l_tmpa_prop { name }
1452 } }
1453
1454 \bool_if:NF \l_stex_symdecl_local_bool {
1455   \exp_args:Nx \stex_add_to_current_module:n {
1456     \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1457       \prop_item:Nn \l_tmpa_prop { module } ?
1458       \prop_item:Nn \l_tmpa_prop { name }
1459     } }
1460   }
1461 }
1462 }
1463
1464 % add to all symbols
1465
1466 \bool_if:NF \l_stex_symdecl_local_bool {
1467   \exp_args:Nx \stex_add_to_current_module:n {
1468     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1469       \prop_item:Nn \l_tmpa_prop { module } ?
1470       \prop_item:Nn \l_tmpa_prop { name }
1471     }
1472   }
1473 }
1474
1475 \stex_debug:n{New~symbol:~
1476   \prop_item:Nn \l_tmpa_prop { module } ?
1477   \prop_item:Nn \l_tmpa_prop { name }^^J
1478   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
1479   Args:~\prop_item:Nn \l_tmpa_prop { args }
1480 }
1481
1482 % circular dependencies require this:
1483
1484 \prop_if_exist:cF {
1485   g_stex_symdecl_
1486   \prop_item:Nn \l_tmpa_prop { module } ?
1487   \prop_item:Nn \l_tmpa_prop { name }
1488   _prop
1489 } {
1490   \prop_gset_eq:cN {
1491     g_stex_symdecl_
1492     \prop_item:Nn \l_tmpa_prop { module } ?
1493     \prop_item:Nn \l_tmpa_prop { name }
1494     _prop
1495   } \l_tmpa_prop
1496 }
1497
1498 \stex_if_smsmode:TF {
1499   \bool_if:NF \l_stex_symdecl_local_bool {
1500     \exp_args:Nx \stex_addtosms:n {
1501       \prop_gset_from_keyval:cn {
1502         g_stex_symdecl_

```

```

1503     \prop_item:Nn \l_tmpa_prop { module } ?
1504     \prop_item:Nn \l_tmpa_prop { name }
1505     _prop
1506   } {
1507     name      = \prop_item:Nn \l_tmpa_prop { name }      ,
1508     module    = \prop_item:Nn \l_tmpa_prop { module }    ,
1509     notations = \prop_item:Nn \l_tmpa_prop { notations } ,
1510     local     = \prop_item:Nn \l_tmpa_prop { local }     ,
1511     type      = \prop_item:Nn \l_tmpa_prop { type }      ,
1512     args      = \prop_item:Nn \l_tmpa_prop { args }      ,
1513     arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
1514     assocs    = \prop_item:Nn \l_tmpa_prop { assocs }
1515   }
1516   \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1517     \prop_item:Nn \l_tmpa_prop { module } ?
1518     \prop_item:Nn \l_tmpa_prop { name }
1519   }
1520 }
1521 }
1522 }{
1523   \exp_args:NNx \seq_put_right:Nn \l_stex_all_symbols_seq {
1524     \prop_item:Nn \l_tmpa_prop { module } ?
1525     \prop_item:Nn \l_tmpa_prop { name }
1526   }
1527   \stex_annotate_invisible:nnn {symdecl} {
1528     \prop_item:Nn \l_tmpa_prop { module } ?
1529     \prop_item:Nn \l_tmpa_prop { name }
1530   } {
1531     \stex_annotate_invisible:nnn{type}{}{\l_stex_symdecl_type_tl$}
1532     \stex_annotate_invisible:nnn{args}{}{
1533       \prop_item:Nn \l_tmpa_prop { args }
1534     }
1535     \stex_annotate_invisible:nnn{macroname}{}{#1}
1536     \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
1537       \stex_annotate_invisible:nnn{definiens}{}{
1538         {\l_stex_symdecl_definiens_tl$}
1539       }
1540     }
1541   }
1542 }

```

(End definition for `\stex_symdecl_do:n`. This function is documented on page 20.)

`\stex_get_symbol:n`

```

1543 \str_new:N \l_stex_get_symbol_uri_str
1544
1545 \cs_new_protected:Nn \stex_get_symbol:n {
1546   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
1547     \__stex_symdecl_get_symbol_from_cs:n { #1 }
1548   }{
1549     % argument is a string
1550     % is it a command name?
1551     \cs_if_exist:cTF { #1 }{
1552       \cs_set_eq:Nc \l_tmpa_tl { #1 }

```

```

1553 \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
1554 \str_if_empty:NTF \l_tmpa_str {
1555   \exp_args:Nx \cs_if_eq:NNTF {
1556     \tl_head:N \l_tmpa_tl
1557   } \stex_invoke_symbol:n {
1558     \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
1559   }{
1560     \__stex_symdecl_get_symbol_from_string:n { #1 }
1561   }
1562 } {
1563   \__stex_symdecl_get_symbol_from_string:n { #1 }
1564 }
1565 }{
1566   % argument is not a command name
1567   \__stex_symdecl_get_symbol_from_string:n { #1 }
1568   % \l_stex_all_symbols_seq
1569 }
1570 }
1571 }
1572
1573 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
1574   \prop_get:NnN \l_stex_current_module_prop
1575   { constants } \l_tmpa_seq
1576   \seq_if_in:NnTF \l_tmpa_seq { #1 } {
1577     \str_set:Nx \l_stex_get_symbol_uri_str {
1578       \prop_item:Nn \l_stex_current_module_prop { ns } ?
1579       \prop_item:Nn \l_stex_current_module_prop { name } ? #1
1580     }
1581   } {
1582     \tl_set:Nn \l_tmpa_tl {
1583       \msg_set:nnn{stex}{error/unknownsymbol}{
1584         No~symbol~#1~found!
1585       }
1586       \msg_error:nn{stex}{error/unknownsymbol}
1587     }
1588     \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1589     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1590     \seq_map_inline:Nn \l_stex_all_symbols_seq {
1591       \str_set:Nn \l_tmpb_str { ##1 }
1592       \str_if_eq:eeT { \l_tmpa_str } {
1593         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1594       } {
1595         \seq_map_break:n {
1596           \tl_set:Nn \l_tmpa_tl {
1597             \str_set:Nn \l_stex_get_symbol_uri_str {
1598               ##1
1599             }
1600           }
1601         }
1602       }
1603     }
1604     \l_tmpa_tl
1605   }
1606 }

```

```

1607
1608 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
1609   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
1610     { \tl_tail:N \l_tmpa_tl }
1611   \tl_if_single:NTF \l_tmpa_tl {
1612     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
1613       \exp_after:wN \str_set:Nn \exp_after:wN
1614         \l_stex_get_symbol_uri_str \l_tmpa_tl
1615     }{
1616       % TODO
1617       % tail is not a single group
1618     }
1619   }{
1620     % TODO
1621     % tail is not a single group
1622   }
1623 }

```

(End definition for `\stex_get_symbol:n`. This function is documented on page 21.)

## 4.7 Notations

```

1624 <@@=stex_notation>
      notation arguments:
1625 \keys_define:nn { stex / notation } {
1626   lang      .tl_set_x:N = \l__stex_notation_lang_str ,
1627   variant   .tl_set_x:N = \l__stex_notation_variant_str ,
1628   prec      .tl_set_x:N = \l__stex_notation_prec_str ,
1629   unknown   .code:n      = \str_set:Nx
1630             \l__stex_notation_variant_str \l_keys_key_str
1631 }
1632
1633 \cs_new_protected:Nn \__stex_notation_args:n {
1634   \str_clear:N \l__stex_notation_lang_str
1635   \str_clear:N \l__stex_notation_variant_str
1636   \str_clear:N \l__stex_notation_prec_str
1637
1638   \keys_set:nn { stex / notation } { #1 }
1639
1640   \str_set:Nx \l__stex_notation_lang_str \l__stex_notation_lang_str
1641   \str_set:Nx \l__stex_notation_variant_str \l__stex_notation_variant_str
1642   \str_set:Nx \l__stex_notation_prec_str \l__stex_notation_prec_str
1643 }

```

**\notation**

```

1644 \NewDocumentCommand \notation { 0{} m } {
1645   \__stex_notation_args:n { #1 }
1646   \tl_clear:N \l_stex_symdecl_definiens_tl
1647   \stex_get_symbol:n { #2 }
1648   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
1649 }

```

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

`\stex_notation_do:nn`

```
1650 \cs_new_protected:Nn \stex_notation_do:nn {
1651   \prop_set_eq:Nc \l_tmpa_prop {
1652     g_stex_symdecl_ #1 _prop
1653   }
1654
1655   \prop_clear:N \l_tmpb_prop
1656   \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
1657   \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
1658   \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
1659
1660   % precedences
1661   \seq_clear:N \l_tmpb_seq
1662   \exp_args:NNno
1663   \str_if_empty:NTF \l__stex_notation_prec_str {
1664     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1665     \int_compare:nNnTF \l_tmpa_str = 0 {
1666       \exp_args:NNnx
1667       \prop_put:Nno \l_tmpb_prop { opprec }
1668       { \infprec }
1669     }{
1670       \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
1671     }
1672   } {
1673     \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
1674     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
1675       \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
1676       \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
1677         \exp_args:NNno \exp_args:NNno \seq_set_split:Nnn
1678         \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
1679         \seq_map_inline:Nn \l_tmpa_seq {
1680           \seq_put_right:Nn \l_tmpb_seq { ##1 }
1681         }
1682       }
1683       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1684     }{
1685       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1686       \int_compare:nNnTF \l_tmpa_str = 0 {
1687         \exp_args:NNnx
1688         \prop_put:Nno \l_tmpb_prop { opprec }
1689         { \infprec }
1690       }{
1691         \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
1692       }
1693     }
1694   }
1695
1696   \seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
1697   \int_step_inline:nn { \l_tmpa_str } {
1698     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpb_str {
1699       \exp_args:NNx
1700       \seq_put_right:Nn \l_tmpb_seq {
1701         \prop_item:Nn \l_tmpb_prop { opprec }
1702       }
1703     }
```

```

1703     }
1704 }
1705
1706 \prop_put:Nno \l_tmpb_prop { argprec } \l_tmpb_seq
1707 \tl_clear:N \l_tmpa_tl
1708
1709 \int_compare:nNnTF \l_tmpa_str = 0 {
1710   \exp_args:Nne
1711   \cs_set:Npn \l__stex_notation_macrocode_cs {
1712     \_stex_term_math_oms:nnnn { #1 }
1713     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1714     { \prop_item:Nn \l_tmpb_prop { opprec } }
1715     { \exp_not:n { #2 } }
1716   }
1717   \__stex_notation_final:
1718 }{
1719   \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
1720   \str_if_in:NnTF \l_tmpb_str b {
1721     \exp_args:Nne \use:nn
1722     {
1723       \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1724       \cs_set:Npn \l_tmpa_str } { {
1725         \_stex_term_math_omb:nnnn { #1 }
1726         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1727         { \prop_item:Nn \l_tmpb_prop { opprec } }
1728         { \exp_not:n { #2 } }
1729       } }
1730   }{
1731     \str_if_in:NnTF \l_tmpb_str B {
1732       \exp_args:Nne \use:nn
1733       {
1734         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1735         \cs_set:Npn \l_tmpa_str } { {
1736           \_stex_term_math_omb:nnnn { #1 }
1737           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1738           { \prop_item:Nn \l_tmpb_prop { opprec } }
1739           { \exp_not:n { #2 } }
1740         } }
1741     }{
1742       \exp_args:Nne \use:nn
1743       {
1744         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1745         \cs_set:Npn \l_tmpa_str } { {
1746           \_stex_term_math_oma:nnnn { #1 }
1747           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1748           { \prop_item:Nn \l_tmpb_prop { opprec } }
1749           { \exp_not:n { #2 } }
1750         } }
1751     }
1752   }
1753
1754   \int_zero:N \l_tmpa_int
1755   \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
1756   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq

```

```

1757     \__stex_notation_arguments:
1758   }
1759 }

```

(End definition for \stex\_notation\_do:nn. This function is documented on page 22.)

\\_\_stex\_notation\_arguments: Takes care of annotating the arguments in a notation macro

```

1760 \cs_new_protected:Nn \__stex_notation_arguments: {
1761   \int_incr:N \l_tmpa_int
1762   \str_if_empty:NTF \l_tmpa_str {
1763     \__stex_notation_final:
1764   }{
1765     \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
1766     \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
1767     \str_if_eq:VnTF \l_tmpb_str a {
1768       \__stex_notation_argument_assoc:n
1769     }{
1770       \str_if_eq:VnTF \l_tmpb_str B {
1771         \__stex_notation_argument_assoc:n
1772       }{
1773         \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1774         \tl_put_right:Nx \l_tmpa_tl {
1775           { \stex_term_math_arg:nnn
1776             { \int_use:N \l_tmpa_int }
1777             { \l_tmpb_str }
1778             { ####\int_use:N \l_tmpa_int }
1779           }
1780         }
1781         \__stex_notation_arguments:
1782       }
1783     }
1784   }
1785 }

```

(End definition for \\_\_stex\_notation\_arguments:.)

\\_\_stex\_notation\_argument\_assoc:n

```

1786 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
1787   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1788   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
1789   \tl_put_right:Nx \l_tmpa_tl {
1790     { \stex_term_math_assoc_arg:nnnn
1791       { \int_use:N \l_tmpa_int }
1792       { \l_tmpb_str }
1793       \exp_args:No \exp_not:n
1794       {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
1795       { ####\int_use:N \l_tmpa_int }
1796     }
1797   }
1798   \__stex_notation_arguments:
1799 }

```

(End definition for \\_\_stex\_notation\_argument\_assoc:n.)



`\_stex_notation_final:` Called after processing all notation arguments

```

1800 \cs_new_protected:Nn \_stex_notation_final: {
1801   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
1802   \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
1803   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
1804   \exp_args:Nne \use:nn
1805   {
1806     \cs_generate_from_arg_count:cNnn {
1807       stex_notation_ \l_tmpa_str \c_hash_str
1808       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1809       _cs
1810     }
1811     \cs_gset:Npn \l_tmpb_str } { {
1812       \exp_after:wN \exp_after:wN \exp_after:wN
1813       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
1814       { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
1815     } }
1816
1817   \stex_debug:n{
1818     Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1819     ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
1820     Operator~precedence:~
1821     \prop_item:Nn \l_tmpb_prop { opprec }^^J
1822     Argument~precedences:~
1823     \seq_use:Nn \l_tmpa_seq {,~}^^J
1824     Notation: \cs_meaning:c {
1825       stex_notation_ \l_tmpa_str \c_hash_str
1826       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1827       _cs
1828     }
1829   }
1830
1831   \prop_gset_eq:cN {
1832     g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
1833     \c_hash_str \l__stex_notation_lang_str _prop
1834   } \l_tmpb_prop
1835
1836   \exp_args:Nx
1837   \stex_add_to_current_module:n {
1838     \prop_get:cnN {
1839       g_stex_symdecl_
1840       \prop_item:Nn \l_tmpb_prop { symbol }
1841       _prop
1842     } { notations } \exp_not:N \l_tmpa_seq
1843     \seq_put_right:Nn \exp_not:N \l_tmpa_seq {
1844       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1845     }
1846     \prop_put:cno {
1847       g_stex_symdecl_
1848       \prop_item:Nn \l_tmpb_prop { symbol }
1849       _prop
1850     } { notations } \exp_not:N \l_tmpa_seq
1851   }
1852

```

```

1853 \stex_if_smsmode:TF {
1854   \stex_smsmode_set_codes:
1855   \exp_args:Nx \stex_addtosms:n {
1856     \prop_gset_from_keyval:cn {
1857       g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
1858       \c_hash_str \l__stex_notation_lang_str _prop
1859     } {
1860       symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
1861       language    = \prop_item:Nn \l_tmpb_prop { language }    ,
1862       variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,
1863       opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
1864       argprecs    = \prop_item:Nn \l_tmpb_prop { argprecs }    ,
1865     }
1866   }
1867 }{
1868   \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
1869   \seq_put_right:Nx \l_tmpa_seq {
1870     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1871   }
1872   \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
1873   \prop_set_eq:cN {
1874     g_stex_symdecl_ \l_tmpa_str _prop
1875   } \l_tmpa_prop
1876
1877   % HTML annotations
1878   \stex_annotate_invisible:nnn { notation }
1879   { \prop_item:Nn \l_tmpb_prop { symbol } } {
1880     \stex_annotate_invisible:nnn { notationfragment }
1881     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
1882     \prop_get:NnN \l_tmpb_prop { argprecs } \l_tmpa_seq
1883     \stex_annotate_invisible:nnn { precedence }
1884     { \prop_item:Nn \l_tmpb_prop { opprec } ;
1885       \seq_use:Nn \l_tmpa_seq { x }
1886     }{}
1887
1888     \int_zero:N \l_tmpa_int
1889     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
1890     \tl_clear:N \l_tmpa_tl
1891     \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{}{
1892       \int_incr:N \l_tmpa_int
1893       \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
1894       \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
1895       \str_if_eq:VnTF \l_tmpb_str a {
1896         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
1897           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
1898           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
1899         } }
1900       }{
1901         \str_if_eq:VnTF \l_tmpb_str B {
1902           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
1903             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
1904             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
1905           } }
1906         }{

```

```

1907         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
1908             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
1909         } }
1910     }
1911 }
1912 }
1913 \stex_annotate_invisible:nnn { notationcomp }{}{
1914     $ \exp_args:Nno \use:nn { \use:c {
1915         stex_notation_ \prop_item:Nn \l_tmpb_prop { symbol }
1916         \c_hash_str \l__stex_notation_variant_str
1917         \c_hash_str \l__stex_notation_lang_str _cs
1918     } } { \l_tmpa_tl } $
1919 }
1920 }
1921 }
1922 }

```

(End definition for `\__stex_notation_final:.`)

**\symdef**

```

1923 \keys_define:nn { stex / symdef } {
1924     name .tl_set_x:N = \l_stex_symdecl_name_str ,
1925     local .bool_set:N = \l_stex_symdecl_local_bool ,
1926     args .tl_set_x:N = \l_stex_symdecl_args_str ,
1927     type .tl_set:N = \l_stex_symdecl_type_tl ,
1928     lang .tl_set_x:N = \l__stex_notation_lang_str ,
1929     variant .tl_set_x:N = \l__stex_notation_variant_str ,
1930     prec .tl_set_x:N = \l__stex_notation_prec_str ,
1931     unknown .code:n = \str_set:Nx
1932         \l__stex_notation_variant_str \l_keys_key_str
1933 }
1934
1935 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
1936     \str_clear:N \l_stex_symdecl_name_str
1937     \str_clear:N \l_stex_symdecl_args_str
1938     \bool_set_false:N \l_stex_symdecl_local_bool
1939     \tl_clear:N \l_stex_symdecl_type_tl
1940     \str_clear:N \l__stex_notation_lang_str
1941     \str_clear:N \l__stex_notation_variant_str
1942     \str_clear:N \l__stex_notation_prec_str
1943
1944     \keys_set:nn { stex / symdef } { #1 }
1945
1946     \exp_args:Nno \str_set:Nn \l_stex_symdecl_name_str
1947         \l_stex_symdecl_name_str
1948     \exp_args:Nno \str_set:Nn \l_stex_symdecl_args_str
1949         \l_stex_symdecl_args_str
1950     \exp_args:Nno \str_set:Nn \l__stex_notation_lang_str
1951         \l__stex_notation_lang_str
1952     \exp_args:Nno \str_set:Nn \l__stex_notation_variant_str
1953         \l__stex_notation_variant_str
1954     \exp_args:Nno \str_set:Nn \l__stex_notation_prec_str
1955         \l__stex_notation_prec_str
1956 }

```

```

1957
1958 \NewDocumentCommand \symdef { 0{} m } {
1959   \__stex_notation_symdef_args:n { #1 }
1960   \tl_clear:N \l_stex_symdecl_definiens_tl
1961   \bool_set_true:N \l_stex_symdecl_make_macro_bool
1962   \stex_symdecl_do:n { #2 }
1963   \exp_args:Nx \stex_notation_do:nn {
1964     \prop_item:Nn \l_tmpa_prop { module } ?
1965     \prop_item:Nn \l_tmpa_prop { name }
1966   }
1967 }

```

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

`\stex_invoke_symbol:n` Invokes a semantic macro

```

1968 \cs_new_protected:Nn \stex_invoke_symbol:n {
1969   \peek_charcode_remove:NTF ! {
1970     \stex_term_custom:nn { #1 } { }
1971   } {
1972     \if_mode_math:
1973       \exp_after:wN \__stex_notation_invoke_math:n
1974     \else:
1975       \exp_after:wN \__stex_notation_invoke_text:n
1976     \fi: { #1 }
1977   }
1978 }

```

(End definition for `\stex_invoke_symbol:n`. This function is documented on page 21.)

`\__stex_notation_invoke_math:n`

```

1979 \cs_new_protected:Nn \__stex_notation_invoke_math:n {
1980   \peek_charcode_remove:NTF * {
1981     \__stex_notation_invoke_text:n { #1 }
1982   } {
1983     \peek_charcode:NTF [ {
1984       \__stex_notation_invoke_math:nw { #1 }
1985     } {
1986       \__stex_notation_invoke_math:nw { #1 } []
1987     }
1988   }
1989 }

```

(End definition for `\__stex_notation_invoke_math:n`.)

`\__stex_notation_invoke_math:nw`

```

1990 \cs_new_protected:Npn \__stex_notation_invoke_math:nw #1 [#2] {
1991   \__stex_notation_args:n { #2 }
1992   \prop_set_eq:Nc \l_tmpa_prop {
1993     g_stex_symdecl_ #1 _prop
1994   }
1995   \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
1996   \seq_if_empty:NTF \l_tmpa_seq {
1997     \msg_set:nnn{stex}{error/nonotations}{
1998       Symbol~#1~used,~but~has~no~notations!
1999     }

```

```

2000 \msg_error:nn{stex}{error/nonotations}
2001 } {
2002 \seq_if_in:NxTF \l_tmpa_seq
2003 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2004 \use:c{
2005 stex_notation_ #1 \c_hash_str
2006 \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2007 _cs
2008 }
2009 }{
2010 \str_if_empty:NTF \l__stex_notation_variant_str {
2011 \str_if_empty:NTF \l__stex_notation_lang_str {
2012 \seq_get_left:NN \l_tmpa_seq \l_tmpa_str
2013 \use:c{
2014 stex_notation_ #1 \c_hash_str \l_tmpa_str
2015 _cs
2016 }
2017 }{
2018 \msg_set:nnn{stex}{error/wrongnotation}{
2019 Symbol~#1~has~no~notation~
2020 \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2021 }
2022 \msg_error:nn{stex}{error/wrongnotation}
2023 }
2024 }{
2025 \msg_set:nnn{stex}{error/wrongnotation}{
2026 Symbol~#1~has~no~notation~
2027 \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2028 }
2029 \msg_error:nn{stex}{error/wrongnotation}
2030 }
2031 }
2032 }
2033 }

```

(End definition for \\_\_stex\_notation\_invoke\_math:nw.)

\\_stex\_notation\_invoke\_text:n

```

2034 \cs_new_protected:Nn \__stex_notation_invoke_text:n {
2035 \prop_set_eq:Nc \l_tmpa_prop {
2036 g_stex_symdecl_ #1 _prop
2037 }
2038 \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2039 \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2040 }

```

(End definition for \\_\_stex\_notation\_invoke\_text:n.)

## 4.8 Terms

2041 <@@=stex\_term>

Precedences:

\infprec

\neginfprec

2042 \tl\_const:Nx \infprec {\int\_use:N \c\_max\_int}

\l\_\_stex\_term\_downprec

```

2043 \tl_const:Nx \neginfpref {-\int_use:N \c_max_int}
2044 \int_new:N \l__stex_term_downprec
2045 \int_set_eq:NN \l__stex_term_downprec \neginfpref

```

(End definition for `\infpref`, `\neginfpref`, and `\l__stex_term_downprec`. These variables are documented on page 23.)

Bracketing:

```

\l__stex_term_left_bracket_str
\l__stex_term_right_bracket_str

```

```

2046 \tl_set:Nn \l__stex_term_left_bracket_str (
2047 \tl_set:Nn \l__stex_term_right_bracket_str )

```

(End definition for `\l__stex_term_left_bracket_str` and `\l__stex_term_right_bracket_str`.)

```

\__stex_term_maybe_brackets:nn

```

Compares precedences and insert brackets accordingly

```

2048 \cs_new_protected:Nn \__stex_term_maybe_brackets:nn {
2049   \int_compare:nNnTF { #1 } < \l__stex_term_downprec {
2050     \bool_if:NTF \l_stex_inarray_bool { #2 }{
2051       \dobrackets { #2 }
2052     }
2053   }{ #2 }
2054 }

```

(End definition for `\__stex_term_maybe_brackets:nn`.)

**\dobrackets**

```

2055 %\RequirePackage{scalerel}
2056 \cs_new_protected:Npn \dobrackets #1 {
2057   %\ThisStyle{\if D\m@switch
2058   %   \exp_args:Nnx \use:nn
2059   %   { \exp_after:wN \left\l__stex_term_left_bracket_str #1 }
2060   %   { \exp_not:N\right\l__stex_term_right_bracket_str }
2061   %   \else
2062     \exp_args:Nnx \use:nn
2063     { \l__stex_term_left_bracket_str #1 }
2064     { \l__stex_term_right_bracket_str }
2065   %\fi}
2066 }

```

(End definition for `\dobrackets`. This function is documented on page 23.)

**\withbrackets**

```

2067 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2068   \exp_args:Nnx \use:nn
2069   {
2070     \tl_set:Nx \l__stex_term_left_bracket_str { #1 }
2071     \tl_set:Nx \l__stex_term_right_bracket_str { #2 }
2072     #3
2073   }
2074   {
2075     \tl_set:Nn \exp_not:N \l__stex_term_left_bracket_str
2076     { \l__stex_term_left_bracket_str }
2077     \tl_set:Nn \exp_not:N \l__stex_term_right_bracket_str
2078     { \l__stex_term_right_bracket_str }
2079   }
2080 }

```

(End definition for `\withbrackets`. This function is documented on page 23.)

`\STEXinvisible`

```
2081 \cs_new_protected:Npn \STEXinvisible #1 {
2082   \stex_annotate_invisible:n { #1 }
2083 }
```

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

OMDOC terms:

`\_stex_term_math_oms:nnnn`

```
2084 \cs_new_protected:Nn \_stex_term_oms:nnn {
2085   \stex_annotate:nnn{ OMID }{ #2 }{
2086     \stex_highlight_term:nn { #1 } { #3 }
2087   }
2088 }
2089
2090 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2091   \__stex_term_maybe_brackets:nn { #3 }{
2092     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2093   }
2094 }
```

(End definition for `\_stex_term_math_oms:nnnn`. This function is documented on page 22.)

`\_stex_term_math_oma:nnnn`

```
2095 \cs_new_protected:Nn \_stex_term_oma:nnn {
2096   \stex_annotate:nnn{ OMA }{ #2 }{
2097     \stex_highlight_term:nn { #1 } { #3 }
2098   }
2099 }
2100
2101 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2102   \__stex_term_maybe_brackets:nn { #3 }{
2103     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2104   }
2105 }
```

(End definition for `\_stex_term_math_oma:nnnn`. This function is documented on page 22.)

`\_stex_term_math_omb:nnnn`

```
2106 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2107   \stex_annotate:nnn{ OMBIND }{ #2 }{
2108     \stex_highlight_term:nn { #1 } { #3 }
2109   }
2110 }
2111
2112 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2113   \__stex_term_maybe_brackets:nn { #3 }{
2114     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2115   }
2116 }
```

(End definition for `\_stex_term_math_omb:nnnn`. This function is documented on page 22.)

`\_stex_term_math_arg:nnn`

```

2117 \cs_new_protected:Nn \_stex_term_arg:nn {
2118   \stex_unhighlight_term:n {
2119     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2120   }
2121 }
2122 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
2123   \exp_args:Nnx \use:nn
2124     { \int_set:Nn \l__stex_term_downprec { #2 }
2125       \_stex_term_arg:nn { #1 } { #3 }
2126     }
2127     { \int_set:Nn \exp_not:N \l__stex_term_downprec { \int_use:N \l__stex_term_downprec } }
2128 }

```

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

`\_stex_term_math_assoc_arg:nnnn`

```

2129 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
2130   \seq_set_split:Nnn \l_tmpa_seq , { #4 }
2131   \int_compare:nNnTF { \seq_count:N \l_tmpa_seq } < 2 {
2132     \tl_set:Nn \l_tmpa_tl { #4 }
2133   }{
2134     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
2135     \seq_reverse:N \l_tmpa_seq
2136     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_tl
2137     \tl_set:No \l_tmpa_tl { \l_tmpb_tl }
2138     \seq_map_inline:Nn \l_tmpa_seq {
2139       \tl_set:Nx \l_tmpa_tl {
2140         \exp_args:Nno
2141           \l_tmpa_cs { ##1 } { \l_tmpa_tl }
2142       }
2143     }
2144   }
2145   \exp_args:Nnno
2146     \_stex_term_math_arg:nnn{#1}{#2}{ \l_tmpa_tl }
2147 }

```

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

`\stex_term_custom:nn`

```

2148 \cs_new_protected:Nn \stex_term_custom:nn {
2149   \str_set:Nn \l__stex_term_custom_uri { #1 }
2150   \str_set:Nn \l_tmpa_str { #2 }
2151   \tl_clear:N \l_tmpa_tl
2152   \int_zero:N \l_tmpa_int
2153   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2154   \__stex_term_custom_loop:
2155 }

```

(End definition for `\stex_term_custom:nn`. This function is documented on page 24.)

`\__stex_term_custom_loop:`

```

2156 \cs_new_protected:Nn \__stex_term_custom_loop: {
2157   \bool_set_false:N \l_tmpa_bool

```



```

2158 \bool_while_do:nn {
2159   \str_if_eq_p:ee X {
2160     \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2161   }
2162 }{
2163   \int_incr:N \l_tmpa_int
2164 }
2165
2166 \peek_charcode:NTF [ {
2167   % notation/text component
2168   \__stex_term_custom_component:w
2169 } {
2170   \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2171     % all arguments read => finish
2172     \__stex_term_custom_final:
2173   } {
2174     % arguments missing
2175     \peek_charcode_remove:NTF * {
2176       % invisible, specific argument position or both
2177       \peek_charcode:NTF [ {
2178         % visible specific argument position
2179         \__stex_term_custom_arg:wn
2180       } {
2181         % invisible
2182         \peek_charcode_remove:NTF * {
2183           % invisible specific argument position
2184           \__stex_term_custom_arg_inv:wn
2185         } {
2186           % invisible next argument
2187           \__stex_term_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2188         }
2189       }
2190     } {
2191       % next normal argument
2192       \__stex_term_custom_arg:wn [ \l_tmpa_int + 1 ]
2193     }
2194   }
2195 }
2196 }

```

(End definition for \\_\_stex\_term\_custom\_loop:.)

\\_\_stex\_term\_custom\_arg\_inv:wn

```

2197 \cs_new_protected:Npn \__stex_term_custom_arg_inv:wn [ #1 ] #2 {
2198   \bool_set_true:N \l_tmpa_bool
2199   \__stex_term_custom_arg:wn [ #1 ] { #2 }
2200 }

```

(End definition for \\_\_stex\_term\_custom\_arg\_inv:wn.)

\\_\_stex\_term\_custom\_arg:wn

```

2201 \cs_new_protected:Npn \__stex_term_custom_arg:wn [ #1 ] #2 {
2202   \str_set:Nx \l_tmpb_str {
2203     \str_item:Nn \l_tmpa_str { #1 }
2204   }

```

```

2205 \str_case:VnTF \l_tmpb_str {
2206   { X } { } % TODO throw error ?
2207   { i } { \__stex_term_custom_set_X:n { #1 } }
2208   { b } { \__stex_term_custom_set_X:n { #1 } }
2209   { a } { \__stex_term_custom_set_X:n { #1 } } % TODO ?
2210   { B } { \__stex_term_custom_set_X:n { #1 } } % TODO ?
2211 }{ }{
2212   % TODO throw error
2213 }
2214
2215 \bool_if:nTF \l_tmpa_bool {
2216   \tl_put_right:Nx \l_tmpa_tl {
2217     \stex_annotate_invisible:n {
2218       \stex_term_arg:nn { \int_eval:n { #1 } }
2219       \exp_not:n { { #2 } }
2220     }
2221   }
2222 } {
2223   \tl_put_right:Nx \l_tmpa_tl {
2224     \stex_term_arg:nn { \int_eval:n { #1 } }
2225     \exp_not:n { { #2 } }
2226   }
2227 }
2228
2229 \__stex_term_custom_loop:
2230 }

```

(End definition for \\_\_stex\_term\_custom\_arg:wn.)

\\_\_stex\_term\_custom\_set\_X:n

```

2231 \cs_new_protected:Nn \__stex_term_custom_set_X:n {
2232   \str_set:Nx \l_tmpa_str {
2233     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
2234     X
2235     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
2236   }
2237 }

```

(End definition for \\_\_stex\_term\_custom\_set\_X:n.)

\\_\_stex\_term\_custom\_component:

```

2238 \cs_new_protected:Npn \__stex_term_custom_component:w [ #1 ] {
2239   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
2240   \__stex_term_custom_loop:
2241 }

```

(End definition for \\_\_stex\_term\_custom\_component:.)

\\_\_stex\_term\_custom\_final:

```

2242 \cs_new_protected:Nn \__stex_term_custom_final: {
2243   \int_compare:nNnTF \l_tmpb_int = 0 {
2244     \exp_args:Nnno \stex_term_oms:nnn
2245   }{
2246     \str_if_in:NnTF \l_tmpa_str {b} {
2247       \exp_args:Nnno \stex_term_ombind:nnn

```

```

2248     } {
2249       \exp_args:Nnno \_stex_term_oma:nnn
2250     }
2251   }
2252   { \l__stex_term_custom_uri } { \l__stex_term_custom_uri } { \l_tmpa_tl }
2253 }

```

(End definition for `\_stex_term_custom_final:.`)

**`\symref`**

```

2254 \NewDocumentCommand \symref { m m }{
2255   \STEXsymbol{#1}![#2]
2256 }

```

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

## 4.9 Notation Components

```

2257 <@@=stex_notationcomps>

```

**`\stex_highlight_term:nn`**

```

2258 \latexml_if:F {
2259   \scalatex_if:F{
2260     \RequirePackage{pdfcomment}
2261   }
2262 }
2263
2264 \str_new:N \l__stex_notationcomps_highlight_uri_str
2265 \cs_new_protected:Nn \stex_highlight_term:nn {
2266   \exp_args:Nnx
2267   \use:nn {
2268     \str_set:Nx \l__stex_notationcomps_highlight_uri_str { #1 }
2269     #2
2270   } {
2271     \str_set:Nx \exp_not:N \l__stex_notationcomps_highlight_uri_str
2272       { \l__stex_notationcomps_highlight_uri_str }
2273   }
2274 }
2275
2276 \cs_new_protected:Nn \stex_unhighlight_term:n {
2277   % \latexml_if:TF {
2278   %   #1
2279   % } {
2280   %   \scalatex_if:TF {
2281   %     #1
2282   %   } {
2283     #1 %\iffalse{\fi} #1 {\iffalse}\fi
2284   %   }
2285   % }
2286 }

```

(End definition for `\stex_highlight_term:nn`. This function is documented on page 24.)

**\comp**

**\@comp**

```
2287 \cs_new_protected:Npn \comp #1 {
2288   \str_if_empty:NF \l__stex_notationcomps_highlight_uri_str {
2289     \scalatex_if:TF {
2290       \stex_annotate:nnn { comp }{ \l__stex_notationcomps_highlight_uri_str }{ #1 }
2291     }{
2292       \exp_args:Nnx \@comp { #1 } { \l__stex_notationcomps_highlight_uri_str }
2293     }
2294   }
2295 }
2296
2297 \cs_new_protected:Npn \@comp #1 #2 {
2298   \pdftooltip {
2299     \textcolor{blue}{#1}
2300   } { #2 }
2301 }
```

(End definition for \comp and \@comp. These functions are documented on page 24.)

**\ellipses**

```
2302 \NewDocumentCommand \ellipses {} { \ldots }
```

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

**\parray**  
**\prmatrix**  
**\parrayline**  
**\parraycell**

```
2303 \bool_new:N \l_stex_inparray_bool
2304 \bool_set_false:N \l_stex_inparray_bool
2305 \NewDocumentCommand \parray { m m } {
2306   \begingroup
2307   \bool_set_true:N \l_stex_inparray_bool
2308   \begin{array}{#1}
2309     #2
2310   \end{array}
2311   \endgroup
2312 }
2313
2314 \NewDocumentCommand \prmatrix { m } {
2315   \begingroup
2316   \bool_set_true:N \l_stex_inparray_bool
2317   \begin{matrix}
2318     #1
2319   \end{matrix}
2320   \endgroup
2321 }
2322
2323 \def \parrayline #1 #2 {
2324   #1 #2 \bool_if:NT \l_stex_inparray_bool {\}
2325 }
2326
2327 \def \parraycell #1 {
2328   #1 \bool_if:NT \l_stex_inparray_bool {&}
2329 }
```

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

## 4.10 Put these somewhere

`symboldoc`

```

2330 \NewDocumentEnvironment{symboldoc}{m}{
2331   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
2332   \seq_clear:N \l_tmpb_seq
2333   \seq_map_inline:Nn \l_tmpa_seq {
2334     \stex_get_symbol:n { ##1 }
2335     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
2336       \l_stex_get_symbol_uri_str
2337     }
2338   }
2339   \par
2340   \exp_args:Nnnx
2341   \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
2342 }{
2343   \end{stex_annotate_env}
2344 }
```

`\MSC`

```

2345 \NewDocumentCommand \MSC {m} {
2346   % TODO
2347 }
```

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

```

2348 \@ifpackageloaded{tikzinput}{
2349   \RequirePackage{stex-tikzinput}
2350 }{}
2351
2352 \AddToHook{begindocument}{
2353   \input{stex-metatheory}
2354 }
2355 \</package>
```

## 4.11 Metatheory

The default meta theory for an  $\text{\LaTeX}$  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  $\Pi$  in dependent type theories.

```

2356 \<*metatheory>
2357 \ExplSyntaxOn
2358 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
2359 \begin{@module}[ns=\c_stex_metatheory_ns_str,meta=NONE]{Metatheory}
2360   \ExplSyntaxOff
2361
2362   % is-a (a:A, a \in A, a is an A, etc.)
2363   \symdecl[args=ai]{isa}
```

```

2364 \notation[typed]{isa}{#1 \comp: #2}{#1 \comp, #2}
2365 \notation[in]{isa}{#1 \comp\in #2}{#1 \comp, #2}
2366 \notation[pred]{isa}{#2\comp(#1 \comp)}{#1 \comp, #2}
2367
2368 % bind (\forall, \Pi, \lambda etc.)
2369 \symdecl[args=Bi]{bind}
2370 \notation[forall]{bind}{\comp\forall #1.\;#2}{#1 \comp, #2}
2371 \notation[\Pi]{bind}{\comp\Prod_{#1}#2}{#1 \comp, #2}
2372 \notation[depfun]{bind}{\comp( #1 \comp)\;\to\; } #2}{#1 \comp, #2}
2373
2374 % dummy variable
2375 \symdecl{dummyvar}
2376 \notation[underscore]{dummyvar}{\comp\_}
2377 \notation[dot]{dummyvar}{\comp\cdot}
2378 \notation[dot]{dummyvar}{\comp\cdot}
2379 \notation[dash]{dummyvar}{\comp{\rm --}}
2380
2381 %fromto (function space, Hom-set, implication etc.)
2382 \symdecl[args=ai]{fromto}
2383 \notation[xarrow]{fromto}{#1 \comp\to #2}{#1 \comp\times #2}
2384 \notation[arrow]{fromto}{#1 \comp\to #2}{#1 \comp\to #2}
2385
2386 % mapto (lambda etc.)
2387 \symdecl[args=Bi]{mapto}
2388 \notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
2389 \notation[lambda]{mapto}{\comp\lambda #1 \comp.\; #2}{#1 \comp, #2}
2390 \notation[lambdau]{mapto}{\comp\lambda_{#1} \comp.\; #2}{#1 \comp, #2}
2391
2392 % function/operator application
2393 \symdecl[args=ia]{apply}
2394 \notation[prec=0;0x\neginfprec,parens]{apply}{#1 \comp( #2 \comp)}{#1 \comp, #2}
2395 \notation[prec=0;0x\neginfprec,lambda]{apply}{#1 \; #2 }{#1 \; #2}
2396
2397 % ‘‘type’’ of all collections (sets, classes, types, kinds)
2398 \symdecl{collection}
2399 \notation[U]{collection}{\comp{\mathcal{U}}}
2400 \notation[set]{collection}{\comp{\textsf{Set}}}
2401
2402 % sequences
2403 \symdecl[args=1]{seqtype}
2404 \notation[kleene]{seqtype}{#1^{\comp\ast}}
2405
2406 \symdef[args=2,li]{sequence-index}{#1_{#2}}
2407 \symdef[args=3]{naseqli}{#1_{#2}\comp{\ellipses},#1_{#3}}
2408
2409 % letin (‘‘let’’, local definitions, variable substitution)
2410 \symdecl[args=bii]{letin}
2411 \notation[let]{letin}{\comp{\rm let}\;#1\comp{=}\;#2\;.\;\comp{\rm in}\;#3}
2412 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
2413 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
2414
2415 % structures
2416 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
2417

```

```

2418 \STEXexport{
2419   \let\nappa\apply
2420   \def\livar{\csname sequence-index\endcsname[li]}
2421 }
2422
2423 \end{@module}
2424 \ExplSyntaxOff
2425 \</metatheory>

```

## 4.12 Auxiliary Packages

### 4.12.1 tikzinput

```

2426 \< *tikzinput>
2427 \< @@=tikzinput>
2428 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
2429 \RequirePackage{13keys2e}
2430
2431 \keys_define:nn { tikzinput } {
2432   image .bool_set:N = \c_tikzinput_image_bool
2433 }
2434
2435 \ProcessKeysOptions { tikzinput }
2436
2437 \bool_if:NTF \c_tikzinput_image_bool {
2438   \RequirePackage{graphicx}
2439
2440   \providecommand\usetikzlibrary[]{}
2441   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
2442 }{
2443   \RequirePackage{tikz}
2444   \RequirePackage{standalone}
2445
2446   \newcommand \tikzinput [2] [] {
2447     \setkeys{Gin}{#1}
2448     \ifx \Gin@width \Gin@exclamation
2449       \ifx \Gin@height \Gin@exclamation
2450         \input { #2 }
2451       \else
2452         \resizebox{!}{ \Gin@height }{
2453           \input { #2 }
2454         }
2455       \fi
2456     \else
2457       \ifx \Gin@height \Gin@exclamation
2458         \resizebox{ \Gin@width }{!}{
2459           \input { #2 }
2460         }
2461       \else
2462         \resizebox{ \Gin@width }{ \Gin@height }{
2463           \input { #2 }
2464         }
2465       \fi
2466     \fi

```

```

2467 }
2468 }
2469
2470 \newcommand \ctikzinput [2] [] {
2471   \begin{center}
2472     \tikzinput [#1] {#2}
2473   \end{center}
2474 }
2475
2476 \@ifpackageloaded{stex}{
2477   \RequirePackage{stex-tikzinput}
2478 }{}
2479 \tikzinput
2480 \*stex-tikzinput
2481 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
2482 \RequirePackage{stex}
2483 \RequirePackage{tikzinput}
2484
2485 % TODO
2486
2487 \stex-tikzinput

```

#### 4.12.2 sTeX1 Compatibility

```

2488 \*smglom
2489 \RequirePackage{expl3, l3keys2e}
2490 \ProvidesExplClass{smglom}{2021/08/01}{1.9}{sTeX1 compatibility}
2491 \LoadClass[border=1px, varwidth]{standalone}
2492 \setlength\textwidth{15cm}
2493 %\g@addto@macro{\@parboxrestore}{\setlength\parskip{\baselineskip}}
2494 \DeclareOption{mh}{\}
2495 \DeclareOption*\PassOptionsToPackage{\CurrentOption}{stex}
2496 \ProcessOptions
2497
2498 \RequirePackage{stex-compatibility}
2499 \smglom
2500
2501 \*compat
2502 \@@=stex_deprec
2503 \ProvidesExplPackage{stex-compatibility}{2021/08/01}{1.9}{bla}
2504 \RequirePackage[debug, lang={de, en}]{stex}
2505
2506 \NewDocumentEnvironment { mhmodnl } { 0{} m m } {
2507   \msg_set:nnn{stex}{warning/deprecated}{
2508     \
2509     Environment~mhmodnl~is~deprected! \
2510     Please~update~module~#2~in~file~
2511     \stex_path_to_string:N \g_stex_currentfile_seq!
2512     \
2513   }
2514   \msg_warning:nn{stex}{warning/deprecated}
2515
2516   \begin{module}[#1, lang=#3]{#2}
2517     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
2518     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str

```



```

2519 \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
2520 \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
2521 \input { \stex_path_to_string:N \l_tmpa_seq / \l_tmpa_str.tex }
2522 } {
2523 \end{module}
2524 }
2525
2526 \NewDocumentEnvironment { modsig } { 0{} m } {
2527 \stex_if_in_module:TF {
2528 \prop_get:NnN \l_stex_current_module_prop {name} \l_tmpa_str
2529 \str_set:Nn \l_tmpb_str { #2 }
2530 \str_if_eq:NNTF \l_tmpa_str \l_tmpb_str {
2531 \prop_set_eq:NN \l_modsig_old_module_prop \l_stex_current_module_prop
2532 \begin{@module}{modsig-#2}
2533 % \prop_set_eq:NN \l_stex_current_module_prop \l_modsig_old_module_prop
2534 } {
2535 \begin{@module}{#2}
2536 }
2537 } {
2538 \begin{@module}{#2}
2539 }
2540 }{
2541 \end{@module}
2542 \AddToHookNext { env / modsig / after }{
2543 \stex_if_in_module:T {
2544 \prop_get:NnN \l_stex_current_module_prop {name} \l_tmpa_str
2545 \str_set:Nn \l_tmpb_str { #2 }
2546 \str_if_eq:NNT \l_tmpa_str \l_tmpb_str {
2547 % \xdef \g_stex_module_after_group_tl {
2548 \stex_if_smsmode:TF {
2549 \exp_args:Nx
2550 \stex_add_to_current_module:n {
2551 \stex_debug:n{Activating~signature~of~#2}
2552 \exp_not:N \prop_item:cn { c_stex_module_
2553 \prop_item:Nn \l_stex_current_module_prop {ns} ?
2554 \prop_item:Nn \l_stex_current_module_prop {name}
2555 / modsig-#2_prop } { content }
2556 }
2557 }
2558 {
2559 \gdef \g_stex_modsig_after_group_tl {
2560 \stex_activate_module:n {
2561 \prop_item:Nn \l_stex_current_module_prop {ns} ?
2562 \prop_item:Nn \l_stex_current_module_prop {name}
2563 / modsig-#2
2564 }
2565
2566 \exp_args:Nx
2567 \stex_add_to_current_module:n {
2568 \stex_activate_module:n {
2569 \prop_item:Nn \l_stex_current_module_prop {ns} ?
2570 \prop_item:Nn \l_stex_current_module_prop {name}
2571 / modsig-#2
2572 }

```

```

2573         }
2574     }
2575     \aftergroup \g_stex_modsig_after_group_tl
2576 }
2577 }
2578 }
2579 }
2580 }
2581
2582 \cs_new_protected:Npn \gimport {
2583     \peek_charcode_remove:NTF * {
2584         \gimport_do:
2585     } {
2586         \gimport_do:
2587     }
2588 }
2589
2590 \NewDocumentCommand \gimport_do: { 0{ } m } {
2591     \msg_set:nnn{stex}{warning/deprecated}{
2592         \\\
2593         \c_backslash_str gimport~is~deprecated! \\\
2594         Please~use~\c_backslash_str importmodule[#1]{#2}~instead!~(in~file~
2595         \stex_path_to_string:N \g_stex_currentfile_seq)
2596         \\\ \\\
2597     }
2598     \msg_warning:nn{stex}{warning/deprecated}
2599     \importmodule[#1]{#2}
2600 }
2601
2602 \cs_new_protected:Npn \guse {
2603     \peek_charcode_remove:NTF * {
2604         \guse_do:
2605     } {
2606         \guse_do:
2607     }
2608 }
2609
2610 \NewDocumentCommand \guse_do: { 0{ } m } {
2611     \msg_set:nnn{stex}{warning/deprecated}{
2612         \\\
2613         \c_backslash_str guse~is~deprecated! \\\
2614         Please~use~\c_backslash_str usemodule[#1]{#2}~instead!~(in~file~
2615         \stex_path_to_string:N \g_stex_currentfile_seq)
2616         \\\ \\\
2617     }
2618     \msg_warning:nn{stex}{warning/deprecated}
2619     \usemodule[#1]{#2}
2620 }
2621
2622 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
2623
2624 \cs_new_protected:Npn \syms {
2625     \peek_charcode_remove:NTF * {
2626         \syms_do:

```

```

2627 } {
2628   \syml_do:
2629 }
2630 }
2631
2632 \NewDocumentCommand \syml_do: { 0{} m } {
2633   \msg_set:nnn{stex}{warning/deprecated}{
2634     \
2635     \c_backslash_str syml~is~deprecated! \
2636     Please~use~\c_backslash_str symdecl[#1]{#2}~instead!~(in~file~
2637     \stex_path_to_string:N \g_stex_currentfile_seq)
2638     \
2639   }
2640   \msg_warning:nn{stex}{warning/deprecated}
2641   \symdecl*{#1}{#2}
2642 }
2643
2644 \cs_new_protected:Npn \syml {
2645   \peek_charcode_remove:NTF * {
2646     \syml_do:
2647   } {
2648     \syml_do:
2649   }
2650 }
2651
2652 \NewDocumentCommand \syml_do: { 0{} m m } {
2653   \msg_set:nnn{stex}{warning/deprecated}{
2654     \
2655     \c_backslash_str syml~is~deprecated! \
2656     Please~use~\c_backslash_str symdecl[#1]{#2-#3}~instead!~(in~file~
2657     \stex_path_to_string:N \g_stex_currentfile_seq)
2658     \
2659   }
2660   \msg_warning:nn{stex}{warning/deprecated}
2661   \symdecl*{#1}{#2-#3}
2662 }
2663
2664 \cs_new_protected:Npn \syml {
2665   \peek_charcode_remove:NTF * {
2666     \syml_do:
2667   } {
2668     \syml_do:
2669   }
2670 }
2671
2672 \NewDocumentCommand \syml_do: { 0{} m m m } {
2673   \msg_set:nnn{stex}{warning/deprecated}{
2674     \
2675     \c_backslash_str syml~is~deprecated! \
2676     Please~use~\c_backslash_str symdecl[#1]{#2-#3-#4}~instead!~(in~file~
2677     \stex_path_to_string:N \g_stex_currentfile_seq)
2678     \
2679   }
2680   \msg_warning:nn{stex}{warning/deprecated}

```

```

2681 \symdecl*{#1}{#2-#3-#4}
2682 }
2683
2684 \keys_define:nn { stex / deprec / defi } {
2685   name .tl_set_x:N = \l_tmpa_str
2686 }
2687
2688 \cs_new_protected:Npn \defi {
2689   \peek_charcode_remove:NTF * {
2690     \defi_do:
2691   } {
2692     \defi_do:
2693   }
2694 }
2695
2696 \NewDocumentCommand \defi_do: { 0{ } m } {
2697   \str_clear:N \l_tmpa_str
2698   \keys_set:nn { stex / deprec / defi } { #1 }
2699
2700   \str_if_empty:NTF \l_tmpa_str {
2701     \msg_set:nnn{stex}{warning/deprecated}{
2702       \\\
2703       \c_backslash_str defi-is~deprecated! \\\
2704       Please~use~\c_backslash_str STEXsymbol{#2}!{#2}~instead!~(in~file~
2705       \stex_path_to_string:N \g_stex_currentfile_seq)
2706       \\\ \\\
2707     }
2708     \msg_warning:nn{stex}{warning/deprecated}
2709     \STEXsymbol { #2 }![ \comp{#2} ]
2710   } {
2711     \msg_set:nnn{stex}{warning/deprecated}{
2712       \\\
2713       \c_backslash_str defi-is~deprecated! \\\
2714       Please~use~\c_backslash_str STEXsymbol { \l_tmpa_str }[ #2 ]~instead!~(in~file~
2715       \stex_path_to_string:N \g_stex_currentfile_seq)
2716       \\\ \\\
2717     }
2718     \msg_warning:nn{stex}{warning/deprecated}
2719     \exp_args:No \STEXsymbol { \l_tmpa_str }![ \comp{#2} ]
2720   }
2721 }
2722
2723
2724 \cs_new_protected:Npn \Defi {
2725   \peek_charcode_remove:NTF * {
2726     \Defi_do:
2727   } {
2728     \Defi_do:
2729   }
2730 }
2731
2732 \NewDocumentCommand \Defi_do: { 0{ } m } {
2733   \str_clear:N \l_tmpa_str
2734   \keys_set:nn { stex / deprec / defi } { #1 }

```

```

2735
2736 \str_if_empty:NTF \l_tmpa_str {
2737   \msg_set:nnn{stex}{warning/deprecated}{
2738     \\\
2739     \c_backslash_str Defi~is~deprecated! \\\
2740     Please~use~\c_backslash_str STEXsymbol{#2}![\exp_after:wN \stex_capitalize:n #2]~inste
2741     \stex_path_to_string:N \g_stex_currentfile_seq)
2742     \\\ \\\
2743   }
2744   \msg_warning:nn{stex}{warning/deprecated}
2745   \STEXsymbol { #2 }![ \comp{\exp_after:wN \stex_capitalize:n #2} ]
2746 } {
2747   \msg_set:nnn{stex}{warning/deprecated}{
2748     \\\
2749     \c_backslash_str Defi~is~deprecated! \\\
2750     Please~use~\c_backslash_str STEXsymbol { \l_tmpa_str }[ \exp_after:wN \stex_capitalize
2751     \stex_path_to_string:N \g_stex_currentfile_seq)
2752     \\\ \\\
2753   }
2754   \msg_warning:nn{stex}{warning/deprecated}
2755   \exp_args:No \STEXsymbol { \l_tmpa_str }![ \comp{\exp_after:wN \stex_capitalize:n #2} ]
2756 }
2757 }
2758
2759 \cs_new_protected:Npn \adefi {
2760   \peek_charcode_remove:NTF * {
2761     \adefi_do:
2762   } {
2763     \adefi_do:
2764   }
2765 }
2766
2767 \NewDocumentCommand \adefi_do: { 0{} m m } {
2768   \str_clear:N \l_tmpa_str
2769   \keys_set:nn { stex / deprec / defi } { #1 }
2770
2771   \str_if_empty:NTF \l_tmpa_str {
2772     \msg_set:nnn{stex}{warning/deprecated}{
2773       \\\
2774       \c_backslash_str adefi~is~deprecated! \\\
2775       Please~use~\c_backslash_str STEXsymbol{#3}![#2]~instead!~(in~file~
2776       \stex_path_to_string:N \g_stex_currentfile_seq)
2777       \\\ \\\
2778     }
2779     \msg_warning:nn{stex}{warning/deprecated}
2780     \STEXsymbol { #3 }![ \comp{#2} ]
2781   } {
2782     \msg_set:nnn{stex}{warning/deprecated}{
2783       \\\
2784       \c_backslash_str adefi~is~deprecated! \\\
2785       Please~use~\c_backslash_str STEXsymbol { \l_tmpa_str }[ #2 ]~instead!~(in~file~
2786       \stex_path_to_string:N \g_stex_currentfile_seq)
2787       \\\ \\\
2788     }

```

```

2789 \msg_warning:nn{stex}{warning/deprecated}
2790 \exp_args:No \STEXsymbol { \l_tmpa_str }![ \comp{#2} ]
2791 }
2792 }
2793
2794 \cs_new_protected:Npn \defis {
2795 \peek_charcode_remove:NTF * {
2796 \defis_do:
2797 } {
2798 \defis_do:
2799 }
2800 }
2801
2802 \NewDocumentCommand \defis_do: { 0{} m } {
2803 \str_clear:N \l_tmpa_str
2804 \keys_set:nn { stex / deprec / defi } { #1 }
2805
2806 \str_if_empty:NTF \l_tmpa_str {
2807 \msg_set:nnn{stex}{warning/deprecated}{
2808 \
2809 \c_backslash_str defis-is-deprecated! \
2810 Please~use~\c_backslash_str STEXsymbol{#2}![#2s]~instead!~(in~file~
2811 \stex_path_to_string:N \g_stex_currentfile_seq)
2812 \
2813 }
2814 \msg_warning:nn{stex}{warning/deprecated}
2815 \STEXsymbol { #2 }![ \comp{#2s} ]
2816 } {
2817 \msg_set:nnn{stex}{warning/deprecated}{
2818 \
2819 \c_backslash_str defis-is-deprecated! \
2820 Please~use~\c_backslash_str STEXsymbol { \l_tmpa_str }[ #2s ]~instead!~(in~file~
2821 \stex_path_to_string:N \g_stex_currentfile_seq)
2822 \
2823 }
2824 \msg_warning:nn{stex}{warning/deprecated}
2825 \exp_args:No \STEXsymbol { \l_tmpa_str }![ \comp{#2s} ]
2826 }
2827 }
2828
2829 \cs_new_protected:Npn \defii {
2830 \peek_charcode_remove:NTF * {
2831 \defii_do:
2832 } {
2833 \defii_do:
2834 }
2835 }
2836
2837 \NewDocumentCommand \defii_do: { 0{} m m } {
2838 \str_clear:N \l_tmpa_str
2839 \keys_set:nn { stex / deprec / defi } { #1 }
2840 \str_if_empty:NTF \l_tmpa_str {
2841 \msg_set:nnn{stex}{warning/deprecated}{
2842 \

```

```

2843     \c_backslash_str defii~is-deprecated! \
2844     Please~use~\c_backslash_str STEXsymbol{#2~#3}![#2~#3]~instead!~(in-file~
2845     \stex_path_to_string:N \g_stex_currentfile_seq)
2846     \
2847 }
2848 \msg_warning:nn{stex}{warning/deprecated}
2849 \STEXsymbol { #2~#3 }![ \comp{#2~#3} ]
2850 } {
2851 \msg_set:nnn{stex}{warning/deprecated}{
2852     \
2853     \c_backslash_str defii~is-deprecated! \
2854     Please~use~\c_backslash_str STEXsymbol { \l_tmpa_str }[ #2~#3 ]~instead!~(in-file~
2855     \stex_path_to_string:N \g_stex_currentfile_seq)
2856     \
2857 }
2858 \msg_warning:nn{stex}{warning/deprecated}
2859 \exp_args:No \STEXsymbol { \l_tmpa_str }![ \comp{#2~#3} ]
2860 }
2861 }
2862
2863
2864 \cs_new_protected:Npn \defiis {
2865     \peek_charcode_remove:NTF * {
2866         \defiis_do:
2867     } {
2868         \defiis_do:
2869     }
2870 }
2871
2872 \NewDocumentCommand \defiis_do: { 0{} m m } {
2873     \str_clear:N \l_tmpa_str
2874     \keys_set:nn { stex / deprec / defi } { #1 }
2875     \str_if_empty:NTF \l_tmpa_str {
2876         \msg_set:nnn{stex}{warning/deprecated}{
2877             \
2878             \c_backslash_str defiiis~is-deprecated! \
2879             Please~use~\c_backslash_str STEXsymbol{#2~#3}![#2~#3s]~instead!~(in-file~
2880             \stex_path_to_string:N \g_stex_currentfile_seq)
2881             \
2882         }
2883         \msg_warning:nn{stex}{warning/deprecated}
2884         \STEXsymbol { #2~#3 }![ \comp{#2~#3s} ]
2885     } {
2886         \msg_set:nnn{stex}{warning/deprecated}{
2887             \
2888             \c_backslash_str defiiis~is-deprecated! \
2889             Please~use~\c_backslash_str STEXsymbol { \l_tmpa_str }[ #2~#3s ]~instead!~(in-file~
2890             \stex_path_to_string:N \g_stex_currentfile_seq)
2891             \
2892         }
2893         \msg_warning:nn{stex}{warning/deprecated}
2894         \exp_args:No \STEXsymbol { \l_tmpa_str }![ \comp{#2~#3s} ]
2895     }
2896 }

```

```

2897
2898
2899 \cs_new_protected:Npn \defiii {
2900   \peek_charcode_remove:NTF * {
2901     \defiii_do:
2902   } {
2903     \defiii_do:
2904   }
2905 }
2906
2907 \NewDocumentCommand \defiii_do: { 0{} m m m } {
2908   \str_clear:N \l_tmpa_str
2909   \keys_set:nn { stex / deprec / defi } { #1 }
2910   \str_if_empty:NTF \l_tmpa_str {
2911     \msg_set:nnn{stex}{warning/deprecated}{
2912       \l_
2913       \c_backslash_str defiii~is-deprecated! \l_
2914       Please~use~\c_backslash_str STEXsymbol{#2~#3~#4}![#2~#3~#4]~instead!~(in~file~
2915       \stex_path_to_string:N \g_stex_currentfile_seq)
2916       \l_ \l_
2917     }
2918     \msg_warning:nn{stex}{warning/deprecated}
2919     \STEXsymbol { #2~#3~#4 }![ \comp{#2~#3~#4} ]
2920   } {
2921     \msg_set:nnn{stex}{warning/deprecated}{
2922       \l_
2923       \c_backslash_str defiii~is-deprecated! \l_
2924       Please~use~\c_backslash_str STEXsymbol { \l_tmpa_str }[ #2~#3~#4 ]~instead!~(in~file~
2925       \stex_path_to_string:N \g_stex_currentfile_seq)
2926       \l_ \l_
2927     }
2928     \msg_warning:nn{stex}{warning/deprecated}
2929     \exp_args:No \STEXsymbol { \l_tmpa_str }![ \comp{#2~#3~#4} ]
2930   }
2931 }
2932
2933 %\RequirePackage[hyperref]{ntheorem}
2934 %\theoremstyle{plain}
2935 %\RequirePackage{amsthm}
2936
2937 \NewDocumentEnvironment {definition} { 0{} } {
2938   \stex_smsmode_set_codes:
2939   \msg_set:nnn{stex}{warning/deprecated}{
2940     \l_
2941     definition~environment~is~deprecated!~(in~file~
2942     \stex_path_to_string:N \g_stex_currentfile_seq)
2943     \l_ \l_
2944   }
2945   \msg_warning:nn{stex}{warning/deprecated}
2946 }{}
2947
2948 \NewDocumentCommand \trefi { 0{} m } {
2949   \str_set:Nn \l_tmpa_str { #1 }
2950   \str_if_empty:NTF \l_tmpa_str {

```



```

2951 \msg_set:nnn{stex}{warning/deprecated}{
2952   \
2953   \c_backslash_str trefi-is-deprecated! \
2954   Please~use~\c_backslash_str STEXsymbol{#2}![#2]~instead!~(in~file~
2955   \stex_path_to_string:N \g_stex_currentfile_seq)
2956   \
2957 }
2958 \msg_warning:nn{stex}{warning/deprecated}
2959 \STEXsymbol { #2 }![ \comp{#2} ]
2960 } {
2961 \msg_set:nnn{stex}{warning/deprecated}{
2962   \
2963   \c_backslash_str trefi-is-deprecated! \
2964   Please~use~\c_backslash_str STEXsymbol { #1?#2 }[ #2 ]~instead!~(in~file~
2965   \stex_path_to_string:N \g_stex_currentfile_seq)
2966   \
2967 }
2968 \msg_warning:nn{stex}{warning/deprecated}
2969 \STEXsymbol { #1 }![ \comp{#2} ]
2970 }
2971 }
2972
2973
2974 \NewDocumentCommand \Trefi { 0{} m } {
2975   \str_set:Nn \l_tmpa_str { #1 }
2976   \str_if_empty:NTF \l_tmpa_str {
2977     \msg_set:nnn{stex}{warning/deprecated}{
2978       \
2979       \c_backslash_str Trefi-is-deprecated! \
2980       Please~use~\c_backslash_str STEXsymbol{#2}![\exp_after:wN \stex_capitalize:n #2]~inste
2981       \stex_path_to_string:N \g_stex_currentfile_seq)
2982       \
2983     }
2984     \msg_warning:nn{stex}{warning/deprecated}
2985     \STEXsymbol { #2 }![ \comp{\exp_after:wN \stex_capitalize:n #2} ]
2986   } {
2987     \msg_set:nnn{stex}{warning/deprecated}{
2988       \
2989       \c_backslash_str Trefi-is-deprecated! \
2990       Please~use~\c_backslash_str STEXsymbol { #1 }[ \exp_after:wN \stex_capitalize:n #2 ]~i
2991       \stex_path_to_string:N \g_stex_currentfile_seq)
2992       \
2993     }
2994     \msg_warning:nn{stex}{warning/deprecated}
2995     \STEXsymbol { #1 }![ \comp{\exp_after:wN \stex_capitalize:n #2} ]
2996   }
2997 }
2998
2999 \NewDocumentCommand \trefis { 0{} m } {
3000   \str_set:Nn \l_tmpa_str { #1 }
3001   \str_if_empty:NTF \l_tmpa_str {
3002     \msg_set:nnn{stex}{warning/deprecated}{
3003       \
3004       \c_backslash_str trefi-is-deprecated! \

```

```

3005     Please~use~\c_backslash_str STExsymbol{#2}![#2s]~instead!~(in~file~
3006     \stex_path_to_string:N \g_stex_currentfile_seq)
3007     \\\ \\\
3008   }
3009   \msg_warning:nn{stex}{warning/deprecated}
3010   \STExsymbol { #2 }![ \comp{#2s} ]
3011 } {
3012   \msg_set:nnn{stex}{warning/deprecated}{
3013     \\\
3014     \c_backslash_str trefi~is~deprecated! \\\
3015     Please~use~\c_backslash_str STExsymbol { #1 }[ #2s ]~instead!~(in~file~
3016     \stex_path_to_string:N \g_stex_currentfile_seq)
3017     \\\ \\\
3018   }
3019   \msg_warning:nn{stex}{warning/deprecated}
3020   \STExsymbol { #1 }![ \comp{#2s} ]
3021 }
3022 }
3023
3024
3025 \NewDocumentCommand \Trefis { 0{} m } {
3026   \str_set:Nn \l_tmpa_str { #1 }
3027   \str_if_empty:NTF \l_tmpa_str {
3028     \msg_set:nnn{stex}{warning/deprecated}{
3029       \\\
3030       \c_backslash_str Trefis~is~deprecated! \\\
3031       Please~use~\c_backslash_str STExsymbol{#2}![\exp_after:wN \stex_capitalize:n #2s]~inst
3032       \stex_path_to_string:N \g_stex_currentfile_seq)
3033       \\\ \\\
3034     }
3035     \msg_warning:nn{stex}{warning/deprecated}
3036     \STExsymbol { #2 }![ \comp{\exp_after:wN \stex_capitalize:n #2s} ]
3037   } {
3038     \msg_set:nnn{stex}{warning/deprecated}{
3039       \\\
3040       \c_backslash_str Trefis~is~deprecated! \\\
3041       Please~use~\c_backslash_str STExsymbol { #1 }[ \exp_after:wN \stex_capitalize:n #2s ]~
3042       \stex_path_to_string:N \g_stex_currentfile_seq)
3043       \\\ \\\
3044     }
3045     \msg_warning:nn{stex}{warning/deprecated}
3046     \STExsymbol { #1 }![ \comp{\exp_after:wN \stex_capitalize:n #2s} ]
3047   }
3048 }
3049
3050 \NewDocumentCommand \trefii { 0{} m m } {
3051   \str_set:Nn \l_tmpa_str { #1 }
3052   \str_if_empty:NTF \l_tmpa_str {
3053     \msg_set:nnn{stex}{warning/deprecated}{
3054       \\\
3055       \c_backslash_str trefii~is~deprecated! \\\
3056       Please~use~\c_backslash_str STExsymbol{#2~#3}![#2~#3]~instead!~(in~file~
3057       \stex_path_to_string:N \g_stex_currentfile_seq)
3058       \\\ \\\

```

```

3059     }
3060     \msg_warning:nn{stex}{warning/deprecated}
3061     \STEXsymbol { #2-#3 }![ \comp{#2-#3} ]
3062   } {
3063     \msg_set:nnn{stex}{warning/deprecated}{
3064       \\\
3065       \c_backslash_str trefiii~is-deprecated! \\\
3066       Please~use~\c_backslash_str STEXsymbol { #1 }[ #2-#3 ]~instead!~(in~file~
3067       \stex_path_to_string:N \g_stex_currentfile_seq)
3068       \\\
3069     }
3070     \msg_warning:nn{stex}{warning/deprecated}
3071     \STEXsymbol { #1 }![ \comp{#2-#3} ]
3072   }
3073 }
3074
3075 \NewDocumentCommand \trefiii { 0{ } m m m } {
3076   \str_set:Nn \l_tmpa_str { #1 }
3077   \str_if_empty:NTF \l_tmpa_str {
3078     \msg_set:nnn{stex}{warning/deprecated}{
3079       \\\
3080       \c_backslash_str trefiii~is-deprecated! \\\
3081       Please~use~\c_backslash_str STEXsymbol{#2-#3-#4}![#2-#3-#4]~instead!~(in~file~
3082       \stex_path_to_string:N \g_stex_currentfile_seq)
3083       \\\
3084     }
3085     \msg_warning:nn{stex}{warning/deprecated}
3086     \STEXsymbol { #2-#3-#4 }![ \comp{#2-#3-#4} ]
3087   } {
3088     \msg_set:nnn{stex}{warning/deprecated}{
3089       \\\
3090       \c_backslash_str trefiii~is-deprecated! \\\
3091       Please~use~\c_backslash_str STEXsymbol { #1 }[ #2-#3-#4 ]~instead!~(in~file~
3092       \stex_path_to_string:N \g_stex_currentfile_seq)
3093       \\\
3094     }
3095     \msg_warning:nn{stex}{warning/deprecated}
3096     \STEXsymbol { #1 }![ \comp{#2-#3-#4} ]
3097   }
3098 }
3099
3100
3101 \NewDocumentCommand \trefiis { 0{ } m m } {
3102   \str_set:Nn \l_tmpa_str { #1 }
3103   \str_if_empty:NTF \l_tmpa_str {
3104     \msg_set:nnn{stex}{warning/deprecated}{
3105       \\\
3106       \c_backslash_str trefiis~is-deprecated! \\\
3107       Please~use~\c_backslash_str STEXsymbol{#2-#3}![#2-#3s]~instead!~(in~file~
3108       \stex_path_to_string:N \g_stex_currentfile_seq)
3109       \\\
3110     }
3111     \msg_warning:nn{stex}{warning/deprecated}
3112     \STEXsymbol { #2-#3 }![ \comp{#2-#3s} ]

```

```

3113 } {
3114   \msg_set:nnn{stex}{warning/deprecated}{
3115     \\\
3116     \c_backslash_str trefiis~is-deprecated! \\\
3117     Please~use~\c_backslash_str STExsymbol { #1 }[ #2~#3s ]~instead!~(in~file~
3118     \stex_path_to_string:N \g_stex_currentfile_seq)
3119     \\\
3120   }
3121   \msg_warning:nn{stex}{warning/deprecated}
3122   \STExsymbol { #1 }![ \comp{#2~#3s} ]
3123 }
3124 }
3125
3126 \NewDocumentCommand \symvariant { O{} m O{0} m m } {
3127   \msg_set:nnn{stex}{warning/deprecated}{
3128     \\\
3129     \c_backslash_str symvariant~is~deprecated! \\\
3130     Please~use~\c_backslash_str notation[#4]{ #2 }~instead!~(in~file~
3131     \stex_path_to_string:N \g_stex_currentfile_seq)
3132     \\\
3133   }
3134   \msg_warning:nn{stex}{warning/deprecated}
3135
3136   \notation[variant=#4]{#2}{#5}
3137 }
3138
3139 \NewDocumentCommand \mixfixi { O{} m m m } {
3140   \msg_set:nnn{stex}{warning/deprecated}{
3141     \c_backslash_str mixfixi~is~fatally~deprecated!\\
3142     Symbol:~\l__stex_term_highlight_uri_str\\
3143     Current~file:~\stex_path_to_string:N \g_stex_currentfile_seq
3144   }
3145   \msg_error:nn{stex}{warning/deprecated}
3146 }
3147
3148
3149 \NewDocumentCommand \infix {} {
3150   \msg_set:nnn{stex}{warning/deprecated}{
3151     \c_backslash_str infix~is~fatally~deprecated!\\
3152     Symbol:~\l__stex_term_highlight_uri_str\\
3153     Current~file:~\stex_path_to_string:N \g_stex_currentfile_seq
3154   }
3155   \msg_error:nn{stex}{warning/deprecated}
3156 }
3157
3158 \let\iprec\infprec
3159
3160 \NewDocumentCommand \inlineex { m } {
3161   \msg_set:nnn{stex}{warning/deprecated}{
3162     \c_backslash_str inlineex~is~deprecated!\\
3163     No~replacement~exists~yet.\\
3164     Current~file:~\stex_path_to_string:N \g_stex_currentfile_seq
3165   }
3166   \msg_warning:nn{stex}{warning/deprecated}

```

```

3167   #1
3168 }
3169
3170
3171 \NewDocumentCommand \term { m } {
3172   \msg_set:nnn{stex}{warning/deprecated}{
3173     \c_backslash_str term~is~deprecated!\\
3174     No~replacement~exists~yet.\\
3175     Current~file:~\stex_path_to_string:N \g_stex_currentfile_seq
3176   }
3177   \msg_warning:nn{stex}{warning/deprecated}
3178   #1
3179 }
3180
3181
3182 \seq_gput_right:Nx \g_stex_smsmode_allowedenvs_seq { \tl_to_str:n { mhmodnl } }
3183 \seq_gput_right:Nx \g_stex_smsmode_allowedenvs_seq { \tl_to_str:n { modsig } }
3184 \tl_gput_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\gimport\symsi\symsii\symsiii\symsiv\
3185
3186 % omtex:
3187 \cs_new_protected:Npn \lec #1 {
3188   \strut\hfil\strut\null\hfill(#1)
3189 }
3190 \cs_new_protected:Npn \nlex #1 {
3191   \textcolor{green}{\s1 #1}}
3192 }
3193
3194
3195 </compat>

```