

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

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

2021-11-23

## **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
3.6	Structural Features . . . . .	25
<b>4</b>	<b>Implementation</b>	<b>25</b>
4.1	The $\text{\LaTeX}$ document class . . . . .	25
4.2	Preliminaries . . . . .	26
4.3	Files, Paths and URIs . . . . .	31
4.4	MathHub Repositories . . . . .	34
4.5	Module System . . . . .	39
4.6	Symbol Declarations . . . . .	56
4.7	Notations . . . . .	62
4.8	Terms . . . . .	72
4.9	Notation Components . . . . .	78
4.10	Structural Features . . . . .	80
4.11	Put these somewhere . . . . .	87
4.12	Metatheory . . . . .	88
4.13	Auxiliary Packages . . . . .	89

## 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

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

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

```
\symdef[ 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. For that reason `\notation` (and thus `\symdef`) take an additional optional argument `op=`, which allows to assign a notation for the operator itself. e.g.

### Example 6

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

The operator  $+$  adds two elements, as in  $(a+b)$ .

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

### Example 7

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

### Example 8

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

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

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

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

### Example 9

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

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

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

### 2.2.2 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 `\foo` could be declared like this:

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

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

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

Notations for symbols of arity 0 have a default precedence of `\infprec`, i.e. by default, parentheses are never inserted around constants. Notations for symbols with

<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?

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 smaller than  $B$ s argument precedences.

For example:

### Example 10

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

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

## 2.3 Archives and Imports

### 2.3.1 Namespaces

Ideally,  $\text{\texttt{S}\TeX}$  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,  $\text{\texttt{T}\TeX}$  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  $\text{\texttt{S}\TeX}$  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 `\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 `\importmodule`:

- `\importmodule{Foo}` outside of an archive refers to module `Foo` in the current namespace. Consequently, `Foo` must have been declared earlier in the same document or, if not, in a file `Foo[.<lang>].tex` in the same directory.

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

- The same statement *within* an archive refers to either the module `Foo` declared earlier in the same document, or otherwise to the module `Foo` in the archive’s top-level namespace. In the latter case, it has to be declared in a file `Foo[.<lang>].tex` directly in the archive’s `source`-folder.
- Similarly, in `\importmodule{some/path?Foo}` the path `some/path` refers to either the sub-directory and relative namespace path of the current directory and namespace outside of an archive, or relative to the current archive’s top-level namespace and `source`-folder, respectively.

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

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

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

## 3 Documentation

### 3.1 Utils

---

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

---



---

<code>\stex_debug:n</code>	<code>\stex_debug:n {&lt;message&gt;}</code>
----------------------------	--

---

Logs `<message>`, if the package option `debug` is used.

---

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

---



---

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

---

#### 3.1.1 ~~SC~~LaTeX, LaTeXML and HTML Annotations

---

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

---



We have four macros for annotating generated HTML (via L<sup>A</sup>T<sub>E</sub>XML or S<sub>C</sub>A<sub>L</sub>T<sub>E</sub>X) with attributes:

---

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

---

Annotates the HTML generated by `⟨content⟩` with

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

### 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 ⟨path-variable⟩ {⟨string⟩}</code>
<code>\stex_path_from_string:(NV cn cV)</code>	

---

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

---

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

---



---

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

---



---

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

---

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

---

`\c_stex_pwd_seq`  
`\c_stex_pwd_str`  
`\c_stex_mainfile_seq`

---

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`.

---

`\g_stex_currentfile_seq`

---

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

### Test 1

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

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

## 3.3 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 <code>module-environment</code> 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?aseqfromto, http://mathhub.info/sTeX?Metatheory?letin, http://mathhub.info/sTeX?Metatheory?m
type, 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  $\S\mathrm{TEX}$ , 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  $\S\mathrm{TEX}$  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\mathrm{Nat}}{x\geq 0}`.

---

`\stex_symdecl_do:n`

---

Implements the core functionality of `\symdecl`, and is called by `\symdecl` and `\symdef`. 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}
\symdecl[def=\bar* abc]{bardef}
\ExplSyntaxOn
Meaning:-\present\bar\
\stex_get_symbol:n { bar }
Result:-\l_stex_get_symbol_uri_str\
Meaning:-\present\bardef\
\ExplSyntaxOff
\end{module}
```

```
Module 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),
- argprecs (sequence of integer strings)

### Test 12

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

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.







<u><u><code>\STEXinvisible</code></u></u>	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.
---	---

<u><u><code>\ellipses</code></u></u>	TODO
--------------------------------------	------

### 3.6 Structural Features

<code>symboldoc</code>	<pre>\begin{&lt;symboldoc&gt;}{&lt;symbols&gt;} &lt;text&gt; \end{&lt;symboldoc&gt;}</pre> <p>Declares <i>&lt;text&gt;</i> to be a (natural language, encyclopaedic) description of <math>\{&lt;symbols&gt;\}</math> (a comma separated list of symbol identifiers).</p>
------------------------	--

#### 3.6.1 Structures

<code>structure</code>	TODO
------------------------	------

#### Test 17

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

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

\present\magma

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

Test: $\mM{op}ab$

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

```
Module 3.18[StructureTest1]
(aob:(M))
file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?StructureTest1/Magma-feature?universe,file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?StructureTest1/Magma-feature?op
>macro->stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?StructureTest1?Magma}
Test: (a+b)
Test2: ((U,+))
```

## 4 Implementation

### 4.1 The `sTeX` document class

```
1 \*cls
2 \RequirePackage{expl3,13keys2e}
```

```

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}

Package options:
16 \keys_define:nn { stex } {
17   debug      .bool_set:N = \c_stex_debug_bool ,
18   showmods   .bool_set:N = \c_stex_showmods_bool ,
19   lang        .clist_set:N = \c_stex_languages_clist ,
20   mathhub     .tl_set_x:N = \mathhub ,
21   sms         .bool_set:N = \c_stex_persist_mode_bool ,
22   image       .bool_set:N = \c_tikzinput_image_bool
23 }
24 \ProcessKeysOptions { stex }

```

**\sTeX** The sTeX logo:

```

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

```

**\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>M</sub>L 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 LATEXML:
\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`  
`\c__stex_annotate_emptyarg_tl`

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

```
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:nnw`

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>XML, S<sup>C</sup>A<sup>L</sup>T<sub>E</sub>X, p<sup>D</sup>f<sup>L</sup>at<sub>E</sub>x).

The p<sup>D</sup>f<sup>L</sup>at<sub>E</sub>x-macros largely do nothing; the S<sup>C</sup>A<sup>L</sup>T<sub>E</sub>X-implementations are pretty clear in what they do, the L<sup>A</sup>T<sub>E</sub>XML-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     \NewDocumentEnvironment{stex_annotate_env} { m m } {
162       \par\begin{latexml@annotateenv}{#1}{#2}
163     }{
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 9.)

### 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:NTF \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:NTF`

```

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 10.)

### 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 10.)

`\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:NTF\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:NF\g__stex_files_stack{
325     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
326   }
327   \seq_if_empty:NTF\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 10.)

## 4.4 MathHub Repositories

```

334 <@@=stex_mathhub>
\mathhub
\c_stex_mathhub_seq
\c_stex_mathhub_str
335 \str_if_empty:NTF\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:NTF\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:NF \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 11.)

`\inputref`

```

483 \newif \ifinputref \inputreffalse
484
485 \cs_new_protected:Nn \inputref:nn {
486   \str_set:Nx \l_tmpa_str { #1 }
487   \str_set:Nx \l_tmpb_str { #2 }
488   \str_if_empty:NT \l_tmpa_str {
489     \prop_if_empty:NF \l_stex_current_repository_prop {
490       \prop_get:NnN \l_stex_current_repository_prop { id } \l_tmpa_str
491     }
492   }
493   \str_if_empty:NF \l_tmpa_str {
494     \stex_require_repository:n \l_tmpa_str
495   }

```

```

496 \str_set:Nx \l_tmpa_str { \c_stex_mathhub_str / \l_tmpa_str / source / \l_tmpb_str }
497 \ifinputref
498   \input{ \l_tmpa_str }
499 \else
500   \inputreftrue
501   \input{ \l_tmpa_str }
502   \inputreffalse
503 \fi
504 }
505 \NewDocumentCommand \inputref { 0{ } m }{
506   \inputref:nn{ #1 }{ #2 }
507 }

```

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

`\mhpath`

```

508 \def \mhpath #1 #2 {
509   \str_if_eq:nnTF{#1}{#2}{
510     \c_stex_mathhub_str /
511     \prop_item:Nn \l_stex_current_repository_prop { id }
512     / source / #2
513   }{
514     \c_stex_mathhub_str / #1 / source / #2
515   }
516 }

```

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

`\libinput`

```

517 \cs_new_protected:Npn \libinput #1 {
518   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
519     \msg_set:nnn{stex}{error/norepository}{
520       \c_backslash_str libinput~needs~to~be~called~in~an~archive
521     }
522     \msg_error:nn{stex}{error/norepository}
523   }
524   \bool_set_false:N \l_tmpa_bool
525   \tl_clear:N \l_tmpa_tl
526   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
527   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
528   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str
529   \seq_pop_left:NNT \l_tmpb_seq \l_tmpb_str {
530     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
531     \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
532       / meta-inf / lib / #1.tex}{
533       \bool_set_true:N \l_tmpa_bool
534       \tl_put_right:Nx \l_tmpa_tl {
535         \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
536           / meta-inf / lib / #1.tex}
537       }
538     }{}
539   }
540   \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
541     / \l_tmpa_str / lib / #1.tex
542   }{

```

```

543 \bool_set_true:N \l_tmpa_bool
544 \tl_put_right:Nx \l_tmpa_tl {
545   \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
546     / \l_tmpa_str / lib / #1.tex}
547 }
548 }{}
549 \bool_if:NF \l_tmpa_bool {
550   \msg_set:nnn{stex}{error/nofile}{
551     \c_backslash_str libinput~no~file~#1.tex~found!
552   }
553   \msg_error:nn{stex}{error/nofile}
554 }
555 \l_tmpa_tl
556 }

```

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

## 4.5 Module System

```

557 <@@=stex_module>

```

`\l_stex_current_module_prop`

```

558 \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`

```

559 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
560   \prop_if_empty:NTF \l_stex_current_module_prop
561   \prg_return_false: \prg_return_true:
562 }

```

(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`

```

563 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
564   \prop_if_exist:cTF { c_stex_module_#1_prop }
565   \prg_return_true: \prg_return_false:
566 }

```

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

`\stex_add_to_current_module:n`

`\STEXexport`

```

567 \cs_new_protected:Nn \stex_add_to_current_module:n {
568   \prop_get:NnN \l_stex_current_module_prop { content } \l_tmpa_tl
569   \tl_put_right:Nn \l_tmpa_tl { #1 }
570   \prop_put:Nno \l_stex_current_module_prop { content } { \l_tmpa_tl }
571 }
572 \NewDocumentCommand \STEXexport { m }{
573   \stex_smsmode_set_codes:
574   \stex_add_to_current_module:n { #1 }
575   #1
576 }

```

(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`

```
577 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
578   \str_set:Nx \l_tmpa_str { #1 }
579   \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
580   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
581   \prop_put:Nno \l_stex_current_module_prop { constants } \l_tmpa_seq
582 }
```

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

`\stex_add_import_to_current_module:n`

```
583 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
584   \str_set:Nx \l_tmpa_str { #1 }
585   \prop_get:NnN \l_stex_current_module_prop { imports } \l_tmpa_seq
586   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
587   \prop_put:Nno \l_stex_current_module_prop { imports } \l_tmpa_seq
588 }
```

(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`

```
589 \str_new:N \l_stex_modules_ns_str

590 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
591   \str_set:Nx \l_tmpa_str { #1 }
592   \seq_set_eq:NN \l_tmpa_seq #2
593   % split off file extension
594   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
595   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
596   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
597   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
598
599   \bool_set_true:N \l_tmpa_bool
600   \bool_while_do:Nn \l_tmpa_bool {
601     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
602     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
603       {source} { \bool_set_false:N \l_tmpa_bool }
604     }{}{
605       \seq_if_empty:NT \l_tmpa_seq {
606         \bool_set_false:N \l_tmpa_bool
607       }
608     }
609   }
610
611   \seq_if_empty:NTF \l_tmpa_seq {
612     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
613   }{
614     \str_set:Nx \l_stex_modules_ns_str {
615       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
616     }
617   }
618 }
```



(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:`

```

619 \cs_new_protected:Nn \stex_modules_current_namespace: {
620   \prop_get:NnNTF \l_stex_current_repository_prop { ns } \l_tmpa_str {
621     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
622   }{
623     % split off file extension
624     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
625     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
626     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
627     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
628     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
629     \str_set:Nx \l_stex_modules_ns_str {
630       file:/\stex_path_to_string:N \l_tmpa_seq
631     }
632   }
633 }

```

(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

```

634 \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`

```

635 \NewDocumentCommand \STEXModule { m } {
636   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
637   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
638   \tl_set:Nn \l_tmpa_tl {
639     \msg_set:nnn{stex}{error/unknownmodule}{
640       No~module~#1~found!
641     }
642     \msg_error:nn{stex}{error/unknownmodule}
643   }
644   \seq_map_inline:Nn \l_stex_all_modules_seq {
645     \str_set:Nn \l_tmpb_str { ##1 }
646     \str_if_eq:eeT { \l_tmpa_str } {
647       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
648     } {
649       \seq_map_break:n {
650         \tl_set:Nn \l_tmpa_tl {
651           \stex_invoke_module:n { ##1 }
652         }
653       }
654     }
655   }
656   \l_tmpa_tl
657 }
658

```

```

659 \cs_new_protected:Nn \stex_invoke_module:n {
660   \stex_debug:n{Invoking~module~#1}
661   \peek_charcode_remove:NTF ! {
662     \__stex_module_invoke_uri:nN { #1 }
663   } {
664     \peek_charcode_remove:NTF ? {
665       \__stex_module_invoke_symbol:nn { #1 }
666     } {
667       \msg_set:nnn{stex}{error/syntax}{
668         Syntax~error:~?~or~!~expected~after~
669         \c_backslash_str STEXModule{#1}
670       }
671       \msg_error:nn{stex}{error/syntax}
672     }
673   }
674 }
675
676 \cs_new_protected:Nn \__stex_module_invoke_uri:nN {
677   \str_set:Nn #2 { #1 }
678 }
679
680 \cs_new_protected:Nn \__stex_module_invoke_symbol:nn {
681   \stex_invoke_symbol:n{#1?#2}
682 }

```

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

module module arguments:

```

683 \keys_define:nn { stex / module } {
684   title      .tl_set_x:N = \l_stex_module_title_str ,
685   ns         .tl_set_x:N = \l_stex_module_ns_str ,
686   lang       .tl_set_x:N = \l_stex_module_lang_str ,
687   sig        .tl_set_x:N = \l_stex_module_sig_str ,
688   creators   .tl_set_x:N = \l_stex_module_creators_str ,
689   contributors .tl_set_x:N = \l_stex_module_contributors_str ,
690   meta       .tl_set_x:N = \l_stex_module_meta_str
691 }
692
693 % module parameters here? In the body?
694
695 \cs_new_protected:Nn \__stex_module_args:n {
696   \str_clear:N \l_stex_module_title_str
697   \str_clear:N \l_stex_module_ns_str
698   \str_clear:N \l_stex_module_lang_str
699   \str_clear:N \l_stex_module_sig_str
700   \str_clear:N \l_stex_module_creators_str
701   \str_clear:N \l_stex_module_contributors_str
702   \str_clear:N \l_stex_module_meta_str
703   \keys_set:nn { stex / module } { #1 }
704   \exp_args:NNo \str_set:Nn \l_stex_module_title_str
705     \l_stex_module_title_str
706   \exp_args:NNo \str_set:Nn \l_stex_module_ns_str
707     \l_stex_module_ns_str

```

```

708 \exp_args:NNo \str_set:Nn \l_stex_module_lang_str
709 \l_stex_module_lang_str
710 \exp_args:NNo \str_set:Nn \l_stex_module_sig_str
711 \l_stex_module_sig_str
712 \exp_args:NNo \str_set:Nn \l_stex_module_meta_str
713 \l_stex_module_meta_str
714 \exp_args:NNo \str_set:Nn \l_stex_module_creators_str
715 \l_stex_module_creators_str
716 \exp_args:NNo \str_set:Nn \l_stex_module_contributors_str
717 \l_stex_module_contributors_str
718 }

```

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

```

719 \cs_new_protected:Nn \_stex_module_begin_module: {
720 % Nested module?
721 \stex_if_in_module:TF {
722 % Nested module
723 \prop_get:NnN \l_stex_current_module_prop
724 { ns } \l_stex_module_ns_str
725 \str_set:Nx \l_stex_module_name_str {
726 \prop_item:Nn \l_stex_current_module_prop
727 { name } / \l_stex_module_name_str
728 }
729 }{
730 % not nested:
731 \str_if_empty:NT \l_stex_module_ns_str {
732 \stex_modules_current_namespace:
733 \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
734 \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
735 / {\l_stex_module_ns_str}
736 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
737 \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
738 \str_set:Nx \l_stex_module_ns_str {
739 \stex_path_to_string:N \l_tmpa_seq
740 }
741 }
742 }
743 }
744
745 % language
746 \str_if_empty:NT \l_stex_module_lang_str {
747 \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
748 \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
749 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
750 \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
751 \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
752 \stex_debug:n {Language~\l_stex_module_lang_str~
753 inferred~from~file~name}
754 \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
755 }
756 }
757
758 \str_if_empty:NF \l_stex_module_lang_str {
759 \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str

```

```

760 \l_tmpa_str {
761 \ltx@ifpackageloaded{babel}{
762 \exp_args:Nx \selectlanguage { \l_tmpa_str }
763 }{ }
764 } {
765 \msg_set:nnn{stex}{error/unknownlanguage}{
766 Unknown~language~\l_tmpa_str
767 }
768 \msg_error:nn{stex}{error/unknownlanguage}
769 }
770 }
771
772 % signature
773 \str_if_empty:NTF \l_stex_module_sig_str {
774 \str_clear:N \l_tmpa_str
775 \seq_clear:N \l_tmpa_seq
776 \tl_clear:N \l_tmpa_tl
777 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
778 name = \l_stex_module_name_str ,
779 ns = \l_stex_module_ns_str ,
780 imports = \exp_not:o { \l_tmpa_seq } ,
781 constants = \exp_not:o { \l_tmpa_seq } ,
782 content = \exp_not:o { \l_tmpa_tl } ,
783 file = \exp_not:o { \g_stex_currentfile_seq } ,
784 lang = \l_stex_module_lang_str ,
785 sig = \l_stex_module_sig_str ,
786 meta = \l_stex_module_meta_str
787 }
788 }{
789 \str_if_empty:NT \l_stex_module_lang_str {
790 \msg_set:nnn{stex}{error/siglanguage}{
791 Module~\l_stex_module_ns_str?\l_stex_module_name_str~
792 declares~signature~\l_stex_module_sig_str,~but~does~not~
793 declare~its~language
794 }
795 \msg_error:nn{stex}{error/siglanguage}
796 }
797
798 \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
799 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
800 \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
801 \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
802 \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
803 \str_set:Nx \l_tmpa_str {
804 \stex_path_to_string:N \l_tmpa_seq /
805 \l_tmpa_str . \l_stex_module_sig_str .tex
806 }
807 \IfFileExists \l_tmpa_str {
808 \exp_args:No \stex_in_smsmode:nn { \l_tmpa_str } {
809 \seq_clear:N \l_stex_all_modules_seq
810 \prop_clear:N \l_stex_current_module_prop
811 \stex_debug:n{Loading~signature~\l_tmpa_str}
812 \input { \l_tmpa_str }
813 }

```

```

814   }{
815     \msg_set:nnn{stex}{error/modulemissing}{
816       No~file~for~signature~module~\l_tmpa_str~found
817     }
818     \msg_error:nn{stex}{error/modulemissing}
819   }
820   \stex_activate_module:n {
821     \l_stex_module_ns_str ? \l_stex_module_name_str
822   }
823   \prop_set_eq:Nc \l_stex_current_module_prop {
824     c_stex_module_
825     \l_stex_module_ns_str ?
826     \l_stex_module_name_str
827     _prop
828   }
829 }
830
831 % metatheory
832 \str_if_empty:NT \l_stex_module_meta_str {
833   \str_set:Nx \l_stex_module_meta_str {
834     \c_stex_metatheory_ns_str ? Metatheory
835   }
836 }
837
838
839 \stex_debug:n{
840   New~module:\\
841   Namespace:~\l_stex_module_ns_str\\
842   Name:~\l_stex_module_name_str\\
843   Language:~\l_stex_module_lang_str\\
844   Signature:~\l_stex_module_sig_str\\
845   Metatheory:~\l_stex_module_meta_str\\
846   File:~\stex_path_to_string:N \g_stex_currentfile_seq
847 }
848
849 \seq_put_right:Nx \l_stex_all_modules_seq {
850   \l_stex_module_ns_str ? \l_stex_module_name_str
851 }
852
853 \seq_gput_right:Nx \g_stex_modules_in_file_seq
854   { \l_stex_module_ns_str ? \l_stex_module_name_str }
855
856 \stex_if_smsmode:TF {
857   \stex_smsmode_set_codes:
858 } {
859   \begin{stex_annotate_env} {theory} {
860     \l_stex_module_ns_str ? \l_stex_module_name_str
861   }
862
863   \stex_annotate_invisible:nnn{header}{} {
864     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
865     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
866     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
867       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}

```

```

868     }
869   }
870 }
871
872 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
873   \exp_args:Nx \STEXexport{
874     \stex_activate_module:n {\l_stex_module_meta_str}
875   }
876 }
877 % TODO: Inherit metatheory for nested modules?
878 }
879 \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}`

```

880 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
881 \cs_new_protected:Nn \_stex_module_end_module: {
882   \str_set:Nx \l_tmpa_str {
883     c_stex_module_
884     \prop_item:Nn \l_stex_current_module_prop { ns } ?
885     \prop_item:Nn \l_stex_current_module_prop { name }
886     _prop
887   }
888   %^^A \prop_new:c { \l_tmpa_str }
889   \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
890   \stex_debug:n{Closing-module~\prop_item:Nn \l_stex_current_module_prop { name }}
891   \stex_if_smsmode:TF {
892     \exp_args:Nx \stex_addtosms:n {
893       \prop_gset_from_keyval:cn {
894         c_stex_module_
895         \prop_item:Nn \l_stex_current_module_prop { ns } ?
896         \prop_item:Nn \l_stex_current_module_prop { name }
897         _prop
898       } {
899         name      = \prop_item:cn { \l_tmpa_str } { name } ,
900         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
901         imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
902         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
903         content   = \prop_item:cn { \l_tmpa_str } { content } ,
904         file      = \prop_item:cn { \l_tmpa_str } { file } ,
905         lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
906         sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
907         meta      = \prop_item:cn { \l_tmpa_str } { meta }
908       }
909     }
910   }{
911     \end{stex_annotate_env}
912   }
913 }

```

(End definition for `\_stex_module_end_module:.`)

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

```

914 \NewDocumentEnvironment { @module } { 0{} m } {
915   \str_set:Nx \l_stex_module_name_str { #2 }
916   \par
917   \__stex_module_args:n { #1 }
918   \__stex_module_begin_module:
919 } {
920   \__stex_module_end_module:
921 }

```

`\stex_modules_heading:` Code for document headers

```

922 \cs_if_exist:NTF \thesection {
923   \newcounter{module}[section]
924 }{
925   \newcounter{module}
926 }
927
928 \bool_if:NT \c_stex_showmods_bool {
929   \latexml_if:F { \RequirePackage{mdframed} }
930 }
931
932 \cs_new_protected:Nn \stex_modules_heading: {
933   \stepcounter{module}
934   \par
935   \bool_if:NT \c_stex_showmods_bool {
936     \noindent{\textbf{Module} ~
937       \cs_if_exist:NT \thesection {\thesection.}
938       \themodule ~ [\l_stex_module_name_str]
939     }
940     % TODO references
941     % \sref@label@id{Module \thesection.\themodule [\module@name]}}%
942     \str_if_empty:NTF \l_stex_module_title_str {
943     }{
944       \quad(\l_stex_module_title_str)\hfill
945     }\par
946   }
947 }

```

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

Finally:

```

948 \NewDocumentEnvironment { module } { 0{} m } {
949   \bool_if:NT \c_stex_showmods_bool {
950     \begin{mdframed}
951   }
952   \begin{@module}[#1]{#2}
953   \stex_modules_heading:
954 }{
955   \end{@module}
956   \bool_if:NT \c_stex_showmods_bool {
957     \end{mdframed}
958   }
959 }

```

## 4.5.2 SMS Mode

960  $\langle @@=\text{stex\_smsmode} \rangle$

$\backslash\text{g\_stex\_smsmode\_allowedmacros\_tl}$   
 $\backslash\text{g\_stex\_smsmode\_allowedmacros\_escape\_tl}$   
 $\backslash\text{g\_stex\_smsmode\_allowedenvs\_seq}$

```

961 \tl_new:N \g_stex_smsmode_allowedmacros_tl
962 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
963 \seq_new:N \g_stex_smsmode_allowedenvs_seq
964
965 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
966   \makeatletter
967   \makeatother
968   \ExplSyntaxOn
969   \ExplSyntaxOff
970 }
971
972 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
973   \symdef
974   \importmodule
975   \notation
976   \symdecl
977   \STEXexport
978 }
979
980 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
981   \tl_to_str:n {
982     module,
983     @module
984   }
985 }
```

(End definition for  $\backslash\text{g\_stex\_smsmode\_allowedmacros\_tl}$ ,  $\backslash\text{g\_stex\_smsmode\_allowedmacros\_escape\_tl}$ , and  $\backslash\text{g\_stex\_smsmode\_allowedenvs\_seq}$ . These variables are documented on page 15.)

$\backslash\text{stex\_if\_smsmode\_p}$ :  
 $\backslash\text{stex\_if\_smsmode}$ :  $\underline{TF}$

```

986 \bool_new:N \g__stex_smsmode_bool
987 \bool_set_false:N \g__stex_smsmode_bool
988 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
989   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
990 }
```

(End definition for  $\backslash\text{stex\_if\_smsmode}$ : $\underline{TF}$ . This function is documented on page 16.)

$\backslash\_stex\_smsmode\_if\_catcodes\_p$ :  
 $\backslash\_stex\_smsmode\_if\_catcodes$ :  $\underline{TF}$

Checks whether the SMS mode category code scheme is active.

```

991 \bool_new:N \g__stex_smsmode_catcode_bool
992 \bool_set_false:N \g__stex_smsmode_catcode_bool
993 \prg_new_conditional:Nnn \_stex_smsmode_if_catcodes: { p, T, F, TF } {
994   \bool_if:NTF \g__stex_smsmode_catcode_bool
995     \prg_return_true: \prg_return_false:
996 }
```

(End definition for  $\backslash\_stex\_smsmode\_if\_catcodes$ : $\underline{TF}$ .)



`\stex_smsmode_set_codes:`

```

997 \cs_new_protected:Nn \stex_smsmode_set_codes: {
998   \stex_if_smsmode:T {
999     \__stex_smsmode_if_catcodes:F {
1000       \bool_gset_true:N \g__stex_smsmode_catcode_bool
1001       \exp_after:wN \char_gset_active_eq:NN
1002       \c_backslash_str \__stex_smsmode_cs:
1003       \tex_global:D \char_set_catcode_active:N \
1004       \tex_global:D \char_set_catcode_other:N $
1005       \tex_global:D \char_set_catcode_other:N ^
1006       \tex_global:D \char_set_catcode_other:N _
1007       \tex_global:D \char_set_catcode_other:N &
1008       \tex_global:D \char_set_catcode_other:N ##
1009     }
1010   }
1011 } \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.

```

1012 \cs_new_protected:Nn \__stex_smsmode_unset_codes: {
1013   \__stex_smsmode_if_catcodes:T {
1014     \bool_gset_false:N \g__stex_smsmode_catcode_bool
1015     \exp_after:wN \tex_global:D \exp_after:wN
1016     \char_set_catcode_escape:N \c_backslash_str
1017     \tex_global:D \char_set_catcode_math_toggle:N $
1018     \tex_global:D \char_set_catcode_math_superscript:N ^
1019     \tex_global:D \char_set_catcode_math_subscript:N _
1020     \tex_global:D \char_set_catcode_alignment:N &
1021     \tex_global:D \char_set_catcode_parameter:N ##
1022   }
1023 } \iffalse $ \fi % to make syntax highlighting work again

```

*(End definition for \\_\_stex\_smsmode\_unset\_codes:.)*

`\stex_in_smsmode:nn`

```

1024 \cs_new_protected:Nn \stex_in_smsmode:nn {
1025   \vbox_set:Nn \l_tmpa_box {
1026     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1027     \bool_gset_true:N \g__stex_smsmode_bool
1028     \stex_smsmode_set_codes:
1029     #2
1030     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1031     \stex_if_smsmode:F {
1032       \__stex_smsmode_unset_codes:
1033     }
1034   }
1035   \box_clear:N \l_tmpa_box
1036 }

```

*(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:

```

1037 \cs_new_protected:Nn \__stex_smsmode_cs: {
1038   \str_clear:N \l_tmpa_str
1039   \peek_analysis_map_inline:n {
1040     % #1: token (one expansion)
1041     % #2: charcode
1042     % #3 catcode
1043     \token_if_eq_charcode:NNTF ##3 B {
1044       % token is a letter
1045       \exp_args:NNo \str_put_right:Nn \l_tmpa_str { ##1 }
1046     } {
1047       \str_if_empty:NNTF \l_tmpa_str {
1048         % we don't allow (or need) single non-letter CSs
1049         % for now
1050         \peek_analysis_map_break:
1051       } {
1052         \str_if_eq:ontf \l_tmpa_str { begin } {
1053           \peek_analysis_map_break:n {
1054             \exp_after:wN \__stex_smsmode_checkbegin:n ##1
1055           }
1056         } {
1057           \str_if_eq:ontf \l_tmpa_str { end } {
1058             \peek_analysis_map_break:n {
1059               \exp_after:wN \__stex_smsmode_checkend:n ##1
1060             }
1061           } {
1062             \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
1063             \exp_args:NNNo \exp_args:NNo \tl_if_in:NnTF
1064               \g_stex_smsmode_allowedmacros_tl
1065               { \use:c{\l_tmpa_str} } {
1066               \stex_debug:n{Executing~1:~\l_tmpa_str}
1067               \peek_analysis_map_break:n {
1068                 \exp_after:wN \l_tmpa_tl ##1
1069               }
1070             } {
1071               \exp_args:NNNo \exp_args:NNo \tl_if_in:NnTF
1072               \g_stex_smsmode_allowedmacros_escape_tl
1073               { \use:c{\l_tmpa_str} } {
1074                 \stex_debug:n{Executing~2:~\l_tmpa_str}
1075                 % TODO \__stex_smsmode_rescan_cs:
1076                 \exp_after:wN \exp_after:wN \exp_after:wN
1077                 \token_if_eq_charcode:NNTF \exp_after:wN \c_backslash_str ##1 {
1078                 \peek_analysis_map_break:n {
1079                   \__stex_smsmode_unset_codes:
1080                   \__stex_smsmode_rescan_cs:
1081                 }
1082                 } {
1083                   \peek_analysis_map_break:n {
1084                     \__stex_smsmode_unset_codes:
1085                     \exp_after:wN \l_tmpa_tl ##1
1086                   }
1087                 }
1088               } {
1089                 \peek_analysis_map_break:n { ##1 }
1090               }

```

```

1091     }
1092   }
1093 }
1094 }
1095 }
1096 }
1097 }

```

(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:

```

1098 \cs_new_protected:Nn \_stex_smsmode_rescan_cs: {
1099   \str_clear:N \l_tmpb_str
1100   \peek_analysis_map_inline:n {
1101     \token_if_eq_charcode:NNTF ##3 B {
1102       % token is a letter
1103       \exp_args:NNo \str_put_right:Nn \l_tmpb_str { ##1 }
1104     } {
1105       \peek_analysis_map_break:n {
1106         \exp_after:wN \use:c \exp_after:wN {
1107           \exp_after:wN \l_tmpa_str\exp_after:wN
1108         } \use:c { \l_tmpb_str \exp_after:wN } ##1
1109       }
1110     }
1111   }
1112 }

```

(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.

```

1113 \cs_new_protected:Nn \_stex_smsmode_checkbegin:n {
1114   \str_set:Nn \l_tmpa_str { #1 }
1115   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1116     \_stex_smsmode_unset_codes:
1117     \begin{#1}
1118   }
1119 }

```

(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.

```

1120 \cs_new_protected:Nn \_stex_smsmode_checkend:n {
1121   \str_set:Nn \l_tmpa_str { #1 }
1122   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1123     \end{#1}
1124   }
1125 }

```

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

### 4.5.3 Inheritance

1126 <@@=stex\_importmodule>

\stex\_import\_module\_uri:nn

```

1127 \cs_new_protected:Nn \stex_import_module_uri:nn {
1128   \str_set:Nx \l__stex_importmodule_archive_str { #1 }
1129   \str_set:Nx \l__stex_importmodule_path_str { #2 }
1130   \str_if_empty:NT \l__stex_importmodule_archive_str {
1131     \prop_if_empty:NF \l_stex_current_repository_prop {
1132       \prop_get:NnN \l_stex_current_repository_prop { id } \l__stex_importmodule_archive_str
1133     }
1134   }
1135
1136   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l__stex_importmodule_path_str }
1137   \seq_pop_right:NN \l_tmpb_seq \l__stex_importmodule_name_str
1138   \str_set:Nx \l__stex_importmodule_path_str { \seq_use:Nn \l_tmpb_seq ? }
1139
1140   \str_if_empty:NTF \l__stex_importmodule_archive_str {
1141     \stex_modules_current_namespace:
1142     \str_if_empty:NF \l__stex_importmodule_path_str {
1143       \str_set:Nx \l_stex_module_ns_str {
1144         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1145       }
1146     }
1147   }{
1148     \stex_require_repository:n \l__stex_importmodule_archive_str
1149     \prop_get:cnN { c_stex_mathhub\l__stex_importmodule_archive_str _manifest_prop } { ns }
1150     \l_stex_module_ns_str
1151     \str_if_empty:NF \l__stex_importmodule_path_str {
1152       \str_set:Nx \l_stex_module_ns_str {
1153         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1154       }
1155     }
1156   }
1157 }
```

(End definition for \stex\_import\_module\_uri:nn. This function is documented on page 19.)

\l\_stex\_importmodule\_name\_str  
\l\_stex\_importmodule\_archive\_str  
\l\_stex\_importmodule\_path\_str  
\l\_stex\_importmodule\_file\_str

Store the return values of \stex\_import\_module\_uri:nn.

```

1158 \str_new:N \l__stex_importmodule_name_str
1159 \str_new:N \l__stex_importmodule_archive_str
1160 \str_new:N \l__stex_importmodule_path_str
1161 \str_new:N \g__stex_importmodule_file_str
```

(End definition for \l\_\_stex\_importmodule\_name\_str and others.)

\stex\_import\_require\_module:nnnn

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

```

1162 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1163   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1164     % \stex_debug:n{Arguments: #1, #2, #3, #4}
1165
1166     % archive
1167     \str_set:Nx \l_tmpa_str { #2 }
1168     \str_if_empty:NTF \l_tmpa_str {
```

```

1169     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1170 } {
1171   \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1172   \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1173   \seq_put_right:Nn \l_tmpa_seq { source }
1174 }
1175
1176 % path
1177 \str_set:Nx \l_tmpb_str { #3 }
1178 \str_if_empty:NTF \l_tmpb_str {
1179   \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1180 }
1181 \ltx@ifpackageloaded{babel} {
1182   \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1183     { \language } \l_tmpb_str {
1184     \msg_set:nnn{stex}{error/unknownlanguage}{
1185       Unknown-language-\language
1186     }
1187     \msg_error:nn{stex}{error/unknownlanguage}
1188   }
1189 } {
1190   \str_clear:N \l_tmpb_str
1191 }
1192
1193 \stex_debug:n{Checking-\l_tmpa_str.\l_tmpb_str.tex}
1194 \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1195   \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1196 }{
1197   \stex_debug:n{Checking-\l_tmpa_str.tex}
1198   \IfFileExists{ \l_tmpa_str.tex }{
1199     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1200   }{
1201     % try english as default
1202     \stex_debug:n{Checking-\l_tmpa_str.en.tex}
1203     \IfFileExists{ \l_tmpa_str.en.tex }{
1204       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1205     }{
1206       \msg_set:nnn{stex}{error/modulemissing}{
1207         No-file-for-module-#1?#4-found
1208       }
1209       \msg_error:nn{stex}{error/modulemissing}
1210     }
1211   }
1212 }
1213
1214 } {
1215   \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1216   \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1217
1218   \ltx@ifpackageloaded{babel} {
1219     \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1220       { \language } \l_tmpb_str {
1221       \msg_set:nnn{stex}{error/unknownlanguage}{
1222         Unknown-language-\language

```

```

1223     }
1224     \msg_error:nn{stex}{error/unknownlanguage}
1225   }
1226 } {
1227   \str_clear:N \l_tmpb_str
1228 }
1229
1230 \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1231
1232 \stex_debug:n{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1233 \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1234   \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1235 }{
1236   \stex_debug:n{Checking~\l_tmpa_str/#4.tex}
1237   \IfFileExists{ \l_tmpa_str/#4.tex }{
1238     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1239   }{
1240     % try english as default
1241     \stex_debug:n{Checking~\l_tmpa_str/#4.en.tex}
1242     \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1243       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1244     }{
1245       \stex_debug:n{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1246       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1247         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1248       }{
1249         \stex_debug:n{Checking~\l_tmpa_str.tex}
1250         \IfFileExists{ \l_tmpa_str.tex }{
1251           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1252         }{
1253           % try english as default
1254           \stex_debug:n{Checking~\l_tmpa_str.en.tex}
1255           \IfFileExists{ \l_tmpa_str.en.tex }{
1256             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1257           }{
1258             \msg_set:nnn{stex}{error/modulemissing}{
1259               No~file~for~module~#1?#4~found
1260             }
1261             \msg_error:nn{stex}{error/modulemissing}
1262           }
1263         }
1264       }
1265     }
1266   }
1267 }
1268 }
1269
1270 \seq_set_eq:NN \l_tmpa_seq \g_stex_modules_in_file_seq
1271 \seq_clear:N \g_stex_modules_in_file_seq
1272 % \exp_args:Nnx \use:nn {
1273   \exp_args:No \stex_in_smsmode:nn { \g__stex_importmodule_file_str } {
1274     \seq_clear:N \l_stex_all_modules_seq
1275     \prop_clear:N \l_stex_current_module_prop
1276     \str_set:Nx \l_tmpb_str { #2 }

```

```

1277     \str_if_empty:NF \l_tmpb_str {
1278         \stex_set_current_repository:n { #2 }
1279     }
1280     \stex_debug:n{Loading~\g__stex_importmodule_file_str}
1281     \input { \g__stex_importmodule_file_str }
1282 }
1283 % }{
1284
1285 % }
1286 \prop_gput:Noo \g_stex_module_files_prop
1287 \g__stex_importmodule_file_str \g_stex_modules_in_file_seq
1288 \seq_set_eq:NN \g_stex_modules_in_file_seq \l_tmpa_seq
1289
1290 \stex_if_module_exists:nF { #1 ? #4 } {
1291     \msg_set:nnn{stex}{error/modulemissing}{
1292         Module~#1?#4~not~found~in~file~\g__stex_importmodule_file_str
1293     }
1294     \msg_error:nn{stex}{error/modulemissing}
1295 }
1296 }
1297 \stex_activate_module:n { #1 ? #4 }
1298 }

```

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

`\stex_activate_module:n`

```

1299 \cs_new_protected:Nn \stex_activate_module:n {
1300     \stex_debug:n{Activating~module~#1}
1301     \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1302         \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1303         \prop_item:cn { c_stex_module_#1_prop } { content }
1304     }
1305 }

```

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

`\importmodule`

```

1306 \NewDocumentCommand \importmodule { 0{} m } {
1307     \stex_import_module_uri:nn { #1 } { #2 }
1308     \stex_debug:n{Importing~module:~
1309         \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1310     }
1311     \stex_if_smsmode:F {
1312         \stex_import_require_module:nnnn
1313         { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1314         { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1315         \stex_annotate_invisible:nnn
1316         {import} { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } {}
1317     }
1318     \exp_args:Nx \stex_add_to_current_module:n {
1319         \stex_import_require_module:nnnn
1320         { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1321         { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1322     }
1323     \exp_args:Nx \stex_add_import_to_current_module:n {

```

```

1324 \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1325 }
1326 \stex_smsmode_set_codes:
1327 }

```

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

#### \usemodule

```

1328 \NewDocumentCommand \usemodule { 0{} m } {
1329 \stex_if_smsmode:F {
1330 \stex_import_module_uri:nn { #1 } { #2 }
1331 \stex_import_require_module:nnnn
1332 { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1333 { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1334 \stex_annotate_invisible:nnn
1335 {usemodule} {\l_stex_module_ns_str ? \l__stex_importmodule_name_str} {}
1336 }
1337 \stex_smsmode_set_codes:
1338 }

```

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

#### \g\_stex\_modules\_in\_file\_seq \g\_stex\_module\_files\_prop

```

1339 \seq_new:N \g_stex_modules_in_file_seq
1340 \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

```

1341 <@@=stex_symdecl>

```

#### \l\_stex\_all\_symbols\_seq Stores all available symbols

```

1342 \seq_new:N \l_stex_all_symbols_seq

```

(End definition for \l\_stex\_all\_symbols\_seq. This variable is documented on page 21.)

#### \STEXsymbol

```

1343 \NewDocumentCommand \STEXsymbol { m } {
1344 \stex_get_symbol:n { #1 }
1345 \exp_args:No
1346 \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1347 }

```

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

symdecl arguments:

```

1348 \keys_define:nn { stex / symdecl } {
1349 name .tl_set:x:N = \l_stex_symdecl_name_str ,
1350 local .bool_set:N = \l_stex_symdecl_local_bool ,
1351 args .tl_set:x:N = \l_stex_symdecl_args_str ,
1352 type .tl_set:N = \l_stex_symdecl_type_tl ,
1353 align .tl_set:N = \l_stex_symdecl_align_str , % TODO(?)
1354 gfc .tl_set:N = \l_stex_symdecl_gfc_str , % TODO(?)
1355 specializes .tl_set:N = \l_stex_symdecl_specializes_str , % TODO(?)

```



```

1356   def          .tl_set:N      = \l_stex_symdecl_definiens_tl
1357 }
1358
1359 \bool_new:N \l_stex_symdecl_make_macro_bool
1360
1361 \cs_new_protected:Nn \__stex_symdecl_args:n {
1362   \str_clear:N \l_stex_symdecl_name_str
1363   \str_clear:N \l_stex_symdecl_args_str
1364   \bool_set_false:N \l_stex_symdecl_local_bool
1365   \tl_clear:N \l_stex_symdecl_type_tl
1366   \tl_clear:N \l_stex_symdecl_definiens_tl
1367
1368   \keys_set:nn { stex /symdecl } { #1 }
1369
1370   \exp_args:NNo \str_set:Nn \l_stex_symdecl_name_str
1371     \l_stex_symdecl_name_str
1372   \exp_args:NNo \str_set:Nn \l_stex_symdecl_args_str
1373     \l_stex_symdecl_args_str
1374 }

```

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

```

1375
1376 \NewDocumentCommand \symdecl { s O{} m } {
1377   \__stex_symdecl_args:n { #2 }
1378   \IfBooleanTF #1 {
1379     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1380   } {
1381     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1382   }
1383   \stex_symdecl_do:n { #3 }
1384   \stex_smsmode_set_codes:
1385 }

```

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

**\stex\_symdecl\_do:n**

```

1386 \cs_new_protected:Nn \stex_symdecl_do:n {
1387   \stex_if_in_module:F {
1388     % TODO throw error? some default namespace?
1389   }
1390
1391   \str_if_empty:NT \l_stex_symdecl_name_str {
1392     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1393   }
1394
1395   \prop_if_exist:cT { g_stex_symdecl_
1396     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1397     \prop_item:Nn \l_stex_current_module_prop {name} ?
1398     \l_stex_symdecl_name_str
1399     _prop
1400   }{
1401     % TODO throw error (beware of circular dependencies)
1402   }

```

```

1403 \prop_clear:N \l_tmpa_prop
1404 \prop_put:Nnx \l_tmpa_prop { module } {
1405   \prop_item:Nn \l_stex_current_module_prop {ns} ?
1406   \prop_item:Nn \l_stex_current_module_prop {name}
1407 }
1408 \seq_clear:N \l_tmpa_seq
1409 \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
1410 \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1411 \prop_put:Nno \l_tmpa_prop { local } \l_stex_symdecl_local_bool
1412 \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1413
1414 \exp_args:No \stex_add_constant_to_current_module:n {
1415   \l_stex_symdecl_name_str
1416 }
1417
1418 % arity/args
1419 \int_zero:N \l_tmpb_int
1420
1421 \bool_set_true:N \l_tmpa_bool
1422 \str_map_inline:Nn \l_stex_symdecl_args_str {
1423   \token_case_meaning:NnF ##1 {
1424     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1425     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1426     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1427     {\tl_to_str:n a} {
1428       \bool_set_false:N \l_tmpa_bool
1429       \int_incr:N \l_tmpb_int
1430     }
1431   }
1432   {\tl_to_str:n B} {
1433     \bool_set_false:N \l_tmpa_bool
1434     \int_incr:N \l_tmpb_int
1435   }
1436 }{
1437   \msg_set:nnn{stex}{error/wrongargs}{
1438     args~value~in~symbol~declaration~for~
1439     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1440     \prop_item:Nn \l_stex_current_module_prop {name} ?
1441     \l_stex_symdecl_name_str ~
1442     needs~to~be~
1443     i,~a,~b~or~B,~but~##1~given
1444   }
1445   \msg_error:nn{stex}{error/wrongargs}
1446 }
1447 }
1448 \bool_if:NTF \l_tmpa_bool {
1449   % possibly numeric
1450   \str_if_empty:NTF \l_stex_symdecl_args_str {
1451     \prop_put:Nnn \l_tmpa_prop { args } {}
1452     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1453   }{
1454     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1455     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1456     \str_clear:N \l_tmpa_str

```

```

1457     \int_step_inline:nn \l_tmpa_int {
1458       \str_put_right:Nn \l_tmpa_str i
1459     }
1460     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1461   }
1462 } {
1463   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1464   \prop_put:Nnx \l_tmpa_prop { arity }
1465     { \str_count:N \l_stex_symdecl_args_str }
1466 }
1467 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
1468
1469
1470 % semantic macro
1471
1472 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1473   \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1474     \prop_item:Nn \l_tmpa_prop { module } ?
1475     \prop_item:Nn \l_tmpa_prop { name }
1476   } }
1477
1478   \bool_if:NF \l_stex_symdecl_local_bool {
1479     \exp_args:Nx \stex_add_to_current_module:n {
1480       \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1481         \prop_item:Nn \l_tmpa_prop { module } ?
1482         \prop_item:Nn \l_tmpa_prop { name }
1483       } }
1484     }
1485   }
1486 }
1487
1488 % add to all symbols
1489
1490 \bool_if:NF \l_stex_symdecl_local_bool {
1491   \exp_args:Nx \stex_add_to_current_module:n {
1492     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1493       \prop_item:Nn \l_tmpa_prop { module } ?
1494       \prop_item:Nn \l_tmpa_prop { name }
1495     }
1496   }
1497 }
1498
1499 \stex_debug:n{New~symbol:~
1500   \prop_item:Nn \l_tmpa_prop { module } ?
1501   \prop_item:Nn \l_tmpa_prop { name }^^J
1502   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
1503   Args:~\prop_item:Nn \l_tmpa_prop { args }
1504 }
1505
1506 % circular dependencies require this:
1507
1508 \prop_if_exist:cF {
1509   g_stex_symdecl_
1510   \prop_item:Nn \l_tmpa_prop { module } ?

```

```

1511     \prop_item:Nn \l_tmpa_prop { name }
1512     _prop
1513   } {
1514     \prop_gset_eq:cN {
1515       g_stex_symdecl_
1516       \prop_item:Nn \l_tmpa_prop { module } ?
1517       \prop_item:Nn \l_tmpa_prop { name }
1518       _prop
1519     } \l_tmpa_prop
1520   }
1521
1522   \stex_if_smsmode:TF {
1523     \bool_if:NF \l_stex_symdecl_local_bool {
1524       \exp_args:Nx \stex_addtosms:n {
1525         \prop_gset_from_keyval:cn {
1526           g_stex_symdecl_
1527           \prop_item:Nn \l_tmpa_prop { module } ?
1528           \prop_item:Nn \l_tmpa_prop { name }
1529           _prop
1530         } {
1531           name      = \prop_item:Nn \l_tmpa_prop { name }      ,
1532           module    = \prop_item:Nn \l_tmpa_prop { module }    ,
1533           notations = \prop_item:Nn \l_tmpa_prop { notations } ,
1534           local     = \prop_item:Nn \l_tmpa_prop { local }      ,
1535           type      = \prop_item:Nn \l_tmpa_prop { type }       ,
1536           args      = \prop_item:Nn \l_tmpa_prop { args }       ,
1537           arity     = \prop_item:Nn \l_tmpa_prop { arity }      ,
1538           assocs    = \prop_item:Nn \l_tmpa_prop { assocs }
1539         }
1540         \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1541           \prop_item:Nn \l_tmpa_prop { module } ?
1542           \prop_item:Nn \l_tmpa_prop { name }
1543         }
1544       }
1545     }
1546   }{
1547     \exp_args:NNx \seq_put_right:Nn \l_stex_all_symbols_seq {
1548       \prop_item:Nn \l_tmpa_prop { module } ?
1549       \prop_item:Nn \l_tmpa_prop { name }
1550     }
1551     \stex_annotate_invisible:nnn {symdecl} {
1552       \prop_item:Nn \l_tmpa_prop { module } ?
1553       \prop_item:Nn \l_tmpa_prop { name }
1554     } {
1555       \stex_annotate_invisible:nnn{type}{}{\l_stex_symdecl_type_tl$}
1556       \stex_annotate_invisible:nnn{args}{}{
1557         \prop_item:Nn \l_tmpa_prop { args }
1558       }
1559       \stex_annotate_invisible:nnn{macroname}{}{#1}
1560       \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
1561         \stex_annotate_invisible:nnn{definiens}{}{
1562           {\l_stex_symdecl_definiens_tl$}
1563         }
1564       }

```

```

1565 }
1566 }

```

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

`\stex_get_symbol:n`

```

1567 \str_new:N \l_stex_get_symbol_uri_str
1568
1569 \cs_new_protected:Nn \stex_get_symbol:n {
1570   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
1571     \__stex_symdecl_get_symbol_from_cs:n { #1 }
1572   }{
1573     % argument is a string
1574     % is it a command name?
1575     \cs_if_exist:cTF { #1 }{
1576       \cs_set_eq:Nc \l_tmpa_tl { #1 }
1577       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
1578       \str_if_empty:NNTF \l_tmpa_str {
1579         \exp_args:Nx \cs_if_eq:NNTF {
1580           \tl_head:N \l_tmpa_tl
1581         } \stex_invoke_symbol:n {
1582           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
1583         }{
1584           \__stex_symdecl_get_symbol_from_string:n { #1 }
1585         }
1586       } {
1587         \__stex_symdecl_get_symbol_from_string:n { #1 }
1588       }
1589     }{
1590       % argument is not a command name
1591       \__stex_symdecl_get_symbol_from_string:n { #1 }
1592       % \l_stex_all_symbols_seq
1593     }
1594   }
1595 }
1596
1597 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
1598   \prop_get:NnN \l_stex_current_module_prop
1599   { constants } \l_tmpa_seq
1600   \seq_if_in:NnTF \l_tmpa_seq { #1 } {
1601     \str_set:Nx \l_stex_get_symbol_uri_str {
1602       \prop_item:Nn \l_stex_current_module_prop { ns } ?
1603       \prop_item:Nn \l_stex_current_module_prop { name } ? #1
1604     }
1605   } {
1606     \tl_set:Nn \l_tmpa_tl {
1607       \msg_set:nnn{stex}{error/unknownsymbol}{
1608         No~symbol~#1~found!
1609       }
1610       \msg_error:nn{stex}{error/unknownsymbol}
1611     }
1612     \str_set:Nn \l_tmpa_str { #1 }
1613     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1614     \seq_map_inline:Nn \l_stex_all_symbols_seq {

```

```

1615     \str_set:Nn \l_tmpb_str { ##1 }
1616     \str_if_eq:eeT { \l_tmpa_str } {
1617       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1618     } {
1619       \seq_map_break:n {
1620         \tl_set:Nn \l_tmpa_tl {
1621           \str_set:Nn \l_stex_get_symbol_uri_str {
1622             ##1
1623           }
1624         }
1625       }
1626     }
1627   }
1628   \l_tmpa_tl
1629 }
1630 }
1631
1632 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
1633   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
1634     { \tl_tail:N \l_tmpa_tl }
1635   \tl_if_single:NTF \l_tmpa_tl {
1636     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
1637       \exp_after:wN \str_set:Nn \exp_after:wN
1638         \l_stex_get_symbol_uri_str \l_tmpa_tl
1639     }{
1640       % TODO
1641       % tail is not a single group
1642     }
1643   }{
1644     % TODO
1645     % tail is not a single group
1646   }
1647 }

```

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

## 4.7 Notations

```

1648 <@@=stex_notation>
1649 notation arguments:
1650 \keys_define:nn { stex / notation } {
1651   lang .tl_set_x:N = \l__stex_notation_lang_str ,
1652   variant .tl_set_x:N = \l__stex_notation_variant_str ,
1653   prec .tl_set_x:N = \l__stex_notation_prec_str ,
1654   op .tl_set:N = \l__stex_notation_op_tl ,
1655   unknown .code:n = \str_set:Nx
1656     \l__stex_notation_variant_str \l_keys_key_str
1657 }
1658 \cs_new_protected:Nn \__stex_notation_args:n {
1659   \str_clear:N \l__stex_notation_lang_str
1660   \str_clear:N \l__stex_notation_variant_str
1661   \str_clear:N \l__stex_notation_prec_str
1662   \tl_clear:N \l__stex_notation_op_tl

```

```

1663
1664 \keys_set:nn { stex / notation } { #1 }
1665
1666 \str_set:Nx \l__stex_notation_lang_str \l__stex_notation_lang_str
1667 \str_set:Nx \l__stex_notation_variant_str \l__stex_notation_variant_str
1668 \str_set:Nx \l__stex_notation_prec_str \l__stex_notation_prec_str
1669 }

```

**\notation**

```

1670 \NewDocumentCommand \notation { 0{ } m } {
1671   \__stex_notation_args:n { #1 }
1672   \tl_clear:N \l_stex_symdecl_definiens_tl
1673   \stex_get_symbol:n { #2 }
1674   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
1675 }

```

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

**\stex\_notation\_do:nn**

```

1676 \cs_new_protected:Nn \stex_notation_do:nn {
1677   \prop_set_eq:Nc \l_tmpa_prop {
1678     g_stex_symdecl_ #1 _prop
1679   }
1680
1681   \prop_clear:N \l_tmpb_prop
1682   \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
1683   \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
1684   \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
1685
1686   % precedences
1687   \seq_clear:N \l_tmpb_seq
1688   \exp_args:NNno
1689   \str_if_empty:NTF \l__stex_notation_prec_str {
1690     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1691     \int_compare:nNnTF \l_tmpa_str = 0 {
1692       \exp_args:NNnx
1693       \prop_put:Nno \l_tmpb_prop { opprec }
1694       { \infprec }
1695     }{
1696       \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
1697     }
1698   } {
1699     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
1700       \exp_args:NNnx
1701       \prop_put:Nno \l_tmpb_prop { opprec }
1702       { \infprec }
1703       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1704       \int_step_inline:nn { \l_tmpa_str } {
1705         \exp_args:NNx
1706         \seq_put_right:Nn \l_tmpb_seq { \neginfprec }
1707       }
1708     }{
1709       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
1710       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
1711         \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str

```

```

1712     \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
1713       \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
1714         \l_tmpa_seq {\tl_to_str:n{x}} } { \l_tmpa_str }
1715       \seq_map_inline:Nn \l_tmpa_seq {
1716         \seq_put_right:Nn \l_tmpb_seq { ##1 }
1717       }
1718     }
1719     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1720   }{
1721     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1722     \int_compare:nNnTF \l_tmpa_str = 0 {
1723       \exp_args:NNnx
1724       \prop_put:Nno \l_tmpb_prop { opprec }
1725       { \infprec }
1726     }{
1727       \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
1728     }
1729   }
1730 }
1731 }
1732
1733 \seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
1734 \int_step_inline:nn { \l_tmpa_str } {
1735   \seq_pop_left:NNT \l_tmpa_seq \l_tmpb_str {
1736     \exp_args:NNx
1737     \seq_put_right:Nn \l_tmpb_seq {
1738       \prop_item:Nn \l_tmpb_prop { opprec }
1739     }
1740   }
1741 }
1742
1743 \prop_put:Nno \l_tmpb_prop { argprecs } \l_tmpb_seq
1744 \tl_clear:N \l_tmpa_tl
1745
1746 \int_compare:nNnTF \l_tmpa_str = 0 {
1747   \exp_args:NNe
1748   \cs_set:Npn \l__stex_notation_macrocode_cs {
1749     \_stex_term_math_oms:nnnn { #1 }
1750     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1751     { \prop_item:Nn \l_tmpb_prop { opprec } }
1752     { \exp_not:n { #2 } }
1753   }
1754   \__stex_notation_final:
1755 }{
1756   \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
1757   \str_if_in:NnTF \l_tmpb_str b {
1758     \exp_args:Nne \use:nn
1759     {
1760       \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1761       \cs_set:Npn \l_tmpa_str { {
1762         \_stex_term_math_omb:nnnn { #1 }
1763         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1764         { \prop_item:Nn \l_tmpb_prop { opprec } }
1765         { \exp_not:n { #2 } }

```



```

1766     }}
1767   }{
1768     \str_if_in:NnTF \l_tmpb_str B {
1769       \exp_args:Nne \use:nn
1770       {
1771         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1772         \cs_set:Npn \l_tmpa_str } { {
1773           \stex_term_math_omb:nnnn { #1 }
1774           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1775           { \prop_item:Nn \l_tmpb_prop { opprec } }
1776           { \exp_not:n { #2 } } }
1777       } }
1778   }{
1779     \exp_args:Nne \use:nn
1780     {
1781       \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1782       \cs_set:Npn \l_tmpa_str } { {
1783         \stex_term_math_oma:nnnn { #1 }
1784         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1785         { \prop_item:Nn \l_tmpb_prop { opprec } }
1786         { \exp_not:n { #2 } } }
1787       } }
1788   }
1789 }
1790
1791 \int_zero:N \l_tmpa_int
1792 \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
1793 \prop_get:NnN \l_tmpb_prop { argprecs } \l_tmpa_seq
1794 \__stex_notation_arguments:
1795 }
1796 }

```

(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

```

1797 \cs_new_protected:Nn \__stex_notation_arguments: {
1798   \int_incr:N \l_tmpa_int
1799   \str_if_empty:NnTF \l_tmpa_str {
1800     \__stex_notation_final:
1801   }{
1802     \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
1803     \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
1804     \str_if_eq:NnTF \l_tmpb_str a {
1805       \__stex_notation_argument_assoc:n
1806     }{
1807       \str_if_eq:NnTF \l_tmpb_str B {
1808         \__stex_notation_argument_assoc:n
1809       }{
1810         \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1811         \tl_put_right:Nx \l_tmpa_tl {
1812           { \stex_term_math_arg:nnn
1813             { \int_use:N \l_tmpa_int }
1814             { \l_tmpb_str }
1815             { ####\int_use:N \l_tmpa_int }

```

```

1816     }
1817   }
1818   \__stex_notation_arguments:
1819 }
1820 }
1821 }
1822 }

```

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

\\_stex\_notation\_argument\_assoc:n

```

1823 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
1824   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1825   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
1826   \tl_put_right:Nx \l_tmpa_tl {
1827     { \stex_term_math_assoc_arg:nnnn
1828       { \int_use:N \l_tmpa_int }
1829       { \l_tmpb_str }
1830       \exp_args:No \exp_not:n
1831       {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
1832       { ####\int_use:N \l_tmpa_int }
1833     }
1834   }
1835   \__stex_notation_arguments:
1836 }

```

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

\\_\_stex\_notation\_final: Called after processing all notation arguments

```

1837 \cs_new_protected:Nn \__stex_notation_final: {
1838   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
1839   \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
1840   \prop_get:NnN \l_tmpb_prop { argprecs } \l_tmpa_seq
1841   \exp_args:Nne \use:nn
1842   {
1843     \cs_generate_from_arg_count:cNnn {
1844       stex_notation_ \l_tmpa_str \c_hash_str
1845       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1846       _cs
1847     }
1848     \cs_gset:Npn \l_tmpb_str { { {
1849       \exp_after:wN \exp_after:wN \exp_after:wN
1850       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
1851       { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
1852     } } }
1853
1854     \tl_if_empty:NF \l__stex_notation_op_tl {
1855       \cs_gset:cpx {
1856         stex_op_notation_ \l_tmpa_str \c_hash_str
1857         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1858         _cs
1859       } {
1860         \stex_term_oms:nnn {
1861           \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str

```

```

1862     \l__stex_notation_lang_str
1863   }{
1864     \l_tmpa_str
1865   }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
1866 }
1867 }
1868
1869
1870
1871 \stex_debug:n{
1872   Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1873   ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
1874   Operator~precedence:~
1875   \prop_item:Nn \l_tmpb_prop { opprec }^^J
1876   Argument~precedences:~
1877   \seq_use:Nn \l_tmpa_seq {,~}^^J
1878   Notation: \cs_meaning:c {
1879     stex_notation_ \l_tmpa_str \c_hash_str
1880     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1881     _cs
1882   }
1883 }
1884
1885 \prop_gset_eq:cN {
1886   g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
1887   \c_hash_str \l__stex_notation_lang_str _prop
1888 } \l_tmpb_prop
1889
1890 \exp_args:Nx
1891 \stex_add_to_current_module:n {
1892   \prop_get:cnN {
1893     g_stex_symdecl_
1894     \prop_item:Nn \l_tmpb_prop { symbol }
1895     _prop
1896   } { notations } \exp_not:N \l_tmpa_seq
1897   \seq_put_right:Nn \exp_not:N \l_tmpa_seq {
1898     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1899   }
1900   \prop_put:cno {
1901     g_stex_symdecl_
1902     \prop_item:Nn \l_tmpb_prop { symbol }
1903     _prop
1904   } { notations } \exp_not:N \l_tmpa_seq
1905 }
1906
1907 \stex_if_smsmode:TF {
1908   \stex_smsmode_set_codes:
1909   \exp_args:Nx \stex_addtosms:n {
1910     \prop_gset_from_keyval:cn {
1911       g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
1912       \c_hash_str \l__stex_notation_lang_str _prop
1913     } {
1914       symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
1915       language    = \prop_item:Nn \l_tmpb_prop { language }    ,

```

```

1916         variant    = \prop_item:Nn \l_tmpb_prop { variant }      ,
1917         opprec     = \prop_item:Nn \l_tmpb_prop { opprec }       ,
1918         argprec    = \prop_item:Nn \l_tmpb_prop { argprec }      ,
1919     }
1920 }
1921 }{
1922   \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
1923   \seq_put_right:Nx \l_tmpa_seq {
1924     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1925   }
1926   \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
1927   \prop_set_eq:cN {
1928     g_stex_symdecl_ \l_tmpa_str _prop
1929   } \l_tmpa_prop
1930
1931   % HTML annotations
1932   \stex_annotate_invisible:nnn { notation }
1933   { \prop_item:Nn \l_tmpb_prop { symbol } } {
1934     \stex_annotate_invisible:nnn { notationfragment }
1935     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
1936     \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
1937     \stex_annotate_invisible:nnn { precedence }
1938     { \prop_item:Nn \l_tmpb_prop { opprec } ;
1939       \seq_use:Nn \l_tmpa_seq { x }
1940     }{}
1941
1942     \int_zero:N \l_tmpa_int
1943     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
1944     \tl_clear:N \l_tmpa_tl
1945     \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
1946       \int_incr:N \l_tmpa_int
1947       \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
1948       \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
1949       \str_if_eq:VnTF \l_tmpb_str a {
1950         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
1951           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
1952           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
1953         } }
1954       }{
1955         \str_if_eq:VnTF \l_tmpb_str B {
1956           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
1957             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
1958             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
1959           } }
1960         }{
1961           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
1962             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
1963           } }
1964         }
1965       }
1966     }
1967     \stex_annotate_invisible:nnn { notationcomp }{}{
1968       $ \exp_args:Nno \use:nn { \use:c {
1969         stex_notation_ \prop_item:Nn \l_tmpb_prop { symbol }

```

```

1970         \c_hash_str \l__stex_notation_variant_str
1971         \c_hash_str \l__stex_notation_lang_str _cs
1972     } } { \l_tmpa_tl } $
1973   }
1974 }
1975 }
1976 }

```

(End definition for \\_stex\_notation\_final:.)

**\symdef**

```

1977 \keys_define:nn { stex / symdef } {
1978   name .tl_set_x:N = \l_stex_symdecl_name_str ,
1979   local .bool_set:N = \l_stex_symdecl_local_bool ,
1980   args .tl_set_x:N = \l_stex_symdecl_args_str ,
1981   type .tl_set:N = \l_stex_symdecl_type_tl ,
1982   def .tl_set:N = \l_stex_symdecl_definiens_tl ,
1983   op .tl_set:N = \l__stex_notation_op_tl ,
1984   lang .tl_set_x:N = \l__stex_notation_lang_str ,
1985   variant .tl_set_x:N = \l__stex_notation_variant_str ,
1986   prec .tl_set_x:N = \l__stex_notation_prec_str ,
1987   unknown .code:n = \str_set:Nx
1988     \l__stex_notation_variant_str \l_keys_key_str
1989 }
1990
1991 \cs_new_protected:Nn \_stex_notation_symdef_args:n {
1992   \str_clear:N \l_stex_symdecl_name_str
1993   \str_clear:N \l_stex_symdecl_args_str
1994   \bool_set_false:N \l_stex_symdecl_local_bool
1995   \tl_clear:N \l_stex_symdecl_type_tl
1996   \tl_clear:N \l_stex_symdecl_definiens_tl
1997   \str_clear:N \l__stex_notation_lang_str
1998   \str_clear:N \l__stex_notation_variant_str
1999   \str_clear:N \l__stex_notation_prec_str
2000   \tl_clear:N \l__stex_notation_op_tl
2001
2002   \keys_set:nn { stex / symdef } { #1 }
2003
2004   \exp_args:NNo \str_set:Nn \l_stex_symdecl_name_str
2005     \l_stex_symdecl_name_str
2006   \exp_args:NNo \str_set:Nn \l_stex_symdecl_args_str
2007     \l_stex_symdecl_args_str
2008   \exp_args:NNo \str_set:Nn \l__stex_notation_lang_str
2009     \l__stex_notation_lang_str
2010   \exp_args:NNo \str_set:Nn \l__stex_notation_variant_str
2011     \l__stex_notation_variant_str
2012   \exp_args:NNo \str_set:Nn \l__stex_notation_prec_str
2013     \l__stex_notation_prec_str
2014 }
2015
2016 \NewDocumentCommand \symdef { 0{} m } {
2017   \_stex_notation_symdef_args:n { #1 }
2018   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2019   \stex_symdecl_do:n { #2 }

```

```

2020 \exp_args:Nx \stex_notation_do:nn {
2021   \prop_item:Nn \l_tmpa_prop { module } ?
2022   \prop_item:Nn \l_tmpa_prop { name }
2023 }
2024 }

```

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

**\stex\_invoke\_symbol:n** Invokes a semantic macro

```

2025 %\cs_new_protected:Nn \stex_invoke_symbol:n {
2026 % \peek_charcode_remove:NTF ! {
2027 %   \stex_term_custom:nn { #1 } { }
2028 % } {
2029 %   \if_mode_math:
2030 %     \exp_after:wN \__stex_notation_invoke_math:n
2031 %   \else:
2032 %     \exp_after:wN \__stex_notation_invoke_text:n
2033 %   \fi: { #1 }
2034 % }
2035 %}
2036
2037 \cs_new_protected:Nn \stex_invoke_symbol:n {
2038   \if_mode_math:
2039     \exp_after:wN \__stex_notation_invoke_math:n
2040   \else:
2041     \exp_after:wN \__stex_notation_invoke_text:n
2042   \fi: { #1 }
2043 }

```

(End definition for \stex\_invoke\_symbol:n. This function is documented on page 21.)

**\\_\_stex\_notation\_invoke\_math:n**

```

2044 \cs_new_protected:Nn \__stex_notation_invoke_math:n {
2045   \peek_charcode_remove:NTF ! {
2046     \peek_charcode:NTF [ {
2047       \__stex_notation_invoke_op:nw { #1 }
2048     }{
2049       \__stex_notation_invoke_op:nw { #1 } []
2050     }
2051   }{
2052     \peek_charcode_remove:NTF * {
2053       \__stex_notation_invoke_text:n { #1 }
2054     }{
2055       \peek_charcode:NTF [ {
2056         \__stex_notation_invoke_math:nw { #1 }
2057       }{
2058         \__stex_notation_invoke_math:nw { #1 } []
2059       }
2060     }
2061   }
2062 }

```

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

\\_stex\_notation\_invoke\_op:nw

```

2063 \cs_new_protected:Npn \__stex_notation_invoke_op:nw #1 [#2] {
2064   \__stex_notation_args:n { #2 }
2065   \cs_if_exist:cTF {
2066     stex_op_notation_ #1 \c_hash_str
2067     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str _cs
2068   }{
2069     \csname stex_op_notation_ #1 \c_hash_str
2070       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str _cs
2071     \endcsname
2072   }{
2073     % TODO throw error
2074   }
2075 }

```

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

\\_stex\_notation\_invoke\_math:nw

```

2076 \cs_new_protected:Npn \__stex_notation_invoke_math:nw #1 [#2] {
2077   \__stex_notation_args:n { #2 }
2078   \prop_set_eq:Nc \l_tmpa_prop {
2079     g_stex_symdecl_ #1 _prop
2080   }
2081   \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
2082   \seq_if_empty:NTF \l_tmpa_seq {
2083     \msg_set:nnn{stex}{error/nonotations}{
2084       Symbol~#1~used,~but~has~no~notations!
2085     }
2086     \msg_error:nn{stex}{error/nonotations}
2087   } {
2088     \seq_if_in:NxTF \l_tmpa_seq
2089     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2090       \use:c{
2091         stex_notation_ #1 \c_hash_str
2092         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2093         _cs
2094       }
2095     }{
2096       \str_if_empty:NTF \l__stex_notation_variant_str {
2097         \str_if_empty:NTF \l__stex_notation_lang_str {
2098           \seq_get_left:NN \l_tmpa_seq \l_tmpa_str
2099           \use:c{
2100             stex_notation_ #1 \c_hash_str \l_tmpa_str
2101             _cs
2102           }
2103         }{
2104           \msg_set:nnn{stex}{error/wrongnotation}{
2105             Symbol~#1~has~no~notation~
2106             \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2107           }
2108           \msg_error:nn{stex}{error/wrongnotation}
2109         }
2110       }{
2111         \msg_set:nnn{stex}{error/wrongnotation}{

```

```

2112         Symbol~#1~has~no~notation~
2113         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2114     }
2115     \msg_error:nn{stex}{error/wrongnotation}
2116 }
2117 }
2118 }
2119 }

```

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

\\_stex\_notation\_invoke\_text:n

```

2120 \cs_new_protected:Nn \__stex_notation_invoke_text:n {
2121     \peek_charcode_remove:NTF ! {
2122         \stex_term_custom:nn { #1 } { }
2123     }{
2124         \prop_set_eq:Nc \l_tmpa_prop {
2125             g_stex_symdecl_ #1 _prop
2126         }
2127         \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2128         \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2129     }
2130 }

```

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

## 4.8 Terms

2131 <@@=stex\_term>

Precedences:

```

\infprec
\neginfprec
\l__stex_term_downprec
2132 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2133 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2134 \int_new:N \l__stex_term_downprec
2135 \int_set_eq:NN \l__stex_term_downprec \neginfprec

```

(End definition for \infprec, \neginfprec, 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
2136 \tl_set:Nn \l__stex_term_left_bracket_str (
2137 \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

```

2138 \cs_new_protected:Nn \__stex_term_maybe_brackets:nn {
2139     \int_compare:nNnTF { #1 } > \l__stex_term_downprec {
2140         \bool_if:NTF \l_stex_inarray_bool { #2 }{
2141             \dobrackets { #2 }
2142         }
2143     }{ #2 }
2144 }

```



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

#### `\dobrackets`

```

2145 %\RequirePackage{scalerel}
2146 \cs_new_protected:Npn \dobrackets #1 {
2147   %\ThisStyle{\if D\m@switch
2148   %   \exp_args:Nnx \use:nn
2149   %   { \exp_after:wN \left\l__stex_term_left_bracket_str #1 }
2150   %   { \exp_not:N\right\l__stex_term_right_bracket_str }
2151   %   \else
2152   %   \exp_args:Nnx \use:nn
2153   %   { \l__stex_term_left_bracket_str #1 }
2154   %   { \l__stex_term_right_bracket_str }
2155   %\fi}
2156 }

```

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

#### `\withbrackets`

```

2157 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2158   \exp_args:Nnx \use:nn
2159   {
2160     \tl_set:Nx \l__stex_term_left_bracket_str { #1 }
2161     \tl_set:Nx \l__stex_term_right_bracket_str { #2 }
2162     #3
2163   }
2164   {
2165     \tl_set:Nn \exp_not:N \l__stex_term_left_bracket_str
2166     { \l__stex_term_left_bracket_str }
2167     \tl_set:Nn \exp_not:N \l__stex_term_right_bracket_str
2168     { \l__stex_term_right_bracket_str }
2169   }
2170 }

```

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

#### `\STEXinvisible`

```

2171 \cs_new_protected:Npn \STEXinvisible #1 {
2172   \stex_annotate_invisible:n { #1 }
2173 }

```

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

OMDOC terms:

#### `\_stex_term_math_oms:nnnn`

```

2174 \cs_new_protected:Nn \_stex_term_oms:nnn {
2175   \stex_annotate:nnn{ OMID }{ #2 }{
2176     \stex_highlight_term:nn { #1 } { #3 }
2177   }
2178 }
2179
2180 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2181   \_stex_term_maybe_brackets:nn { #3 }{
2182     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2183   }
2184 }

```

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

`\_stex_term_math_oma:nnnn`

```

2185 \cs_new_protected:Nn \_stex_term_oma:nnn {
2186   \stex_annotate:nnn{ OMA }{ #2 }{
2187     \stex_highlight_term:nn { #1 } { #3 }
2188   }
2189 }
2190
2191 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2192   \_stex_term_maybe_brackets:nn { #3 }{
2193     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2194   }
2195 }

```

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

`\_stex_term_math_omb:nnnn`

```

2196 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2197   \stex_annotate:nnn{ OMBIND }{ #2 }{
2198     \stex_highlight_term:nn { #1 } { #3 }
2199   }
2200 }
2201
2202 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2203   \_stex_term_maybe_brackets:nn { #3 }{
2204     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2205   }
2206 }

```

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

`\_stex_term_math_arg:nnn`

```

2207 \cs_new_protected:Nn \_stex_term_arg:nn {
2208   \stex_unhighlight_term:n {
2209     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2210   }
2211 }
2212 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
2213   \exp_args:Nnx \use:nn
2214   { \int_set:Nn \l__stex_term_downprec { #2 }
2215     \_stex_term_arg:nn { #1 }{ #3 }
2216   }
2217   { \int_set:Nn \exp_not:N \l__stex_term_downprec { \int_use:N \l__stex_term_downprec } }
2218 }

```

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

`\_stex_term_math_assoc_arg:nnnn`

```

2219 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
2220   \seq_set_split:Nnn \l_tmpa_seq , { #4 }
2221   \int_compare:nNnTF { \seq_count:N \l_tmpa_seq } < 2 {
2222     \tl_set:Nn \l_tmpa_tl { #4 }
2223   }{
2224     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }

```

```

2225 \seq_reverse:N \l_tmpa_seq
2226 \seq_pop_left:NN \l_tmpa_seq \l_tmpb_tl
2227 \tl_set:No \l_tmpa_tl { \l_tmpb_tl }
2228
2229 \seq_map_inline:Nn \l_tmpa_seq {
2230   \exp_args:NNo \tl_set:No \l_tmpa_tl {
2231     \exp_args:Nno
2232     \l_tmpa_cs { ##1 } \l_tmpa_tl
2233   }
2234 }
2235
2236 }
2237 \exp_args:Nnno
2238 \stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2239 }

```

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

`\stex_term_custom:nn`

```

2240 \cs_new_protected:Nn \stex_term_custom:nn {
2241   \str_set:Nn \l__stex_term_custom_uri { #1 }
2242   \str_set:Nn \l_tmpa_str { #2 }
2243   \tl_clear:N \l_tmpa_tl
2244   \int_zero:N \l_tmpa_int
2245   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2246   \__stex_term_custom_loop:
2247 }

```

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

`\__stex_term_custom_loop:`

```

2248 \cs_new_protected:Nn \__stex_term_custom_loop: {
2249   \bool_set_false:N \l_tmpa_bool
2250   \bool_while_do:nn {
2251     \str_if_eq_p:ee X {
2252       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2253     }
2254   }{
2255     \int_incr:N \l_tmpa_int
2256   }
2257
2258   \peek_charcode:NTF [ {
2259     % notation/text component
2260     \__stex_term_custom_component:w
2261   } {
2262     \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2263       % all arguments read => finish
2264       \__stex_term_custom_final:
2265     } {
2266       % arguments missing
2267       \peek_charcode_remove:NTF * {
2268         % invisible, specific argument position or both
2269         \peek_charcode:NTF [ {
2270           % visible specific argument position
2271           \__stex_term_custom_arg:wn

```

```

2272     } {
2273         % invisible
2274         \peek_charcode_remove:NTF * {
2275             % invisible specific argument position
2276             \__stex_term_custom_arg_inv:wn
2277         } {
2278             % invisible next argument
2279             \__stex_term_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2280         }
2281     }
2282 } {
2283     % next normal argument
2284     \__stex_term_custom_arg:wn [ \l_tmpa_int + 1 ]
2285 }
2286 }
2287 }
2288 }

```

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

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

```

2289 \cs_new_protected:Npn \__stex_term_custom_arg_inv:wn [ #1 ] #2 {
2290     \bool_set_true:N \l_tmpa_bool
2291     \__stex_term_custom_arg:wn [ #1 ] { #2 }
2292 }

```

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

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

```

2293 \cs_new_protected:Npn \__stex_term_custom_arg:wn [ #1 ] #2 {
2294     \str_set:Nx \l_tmpb_str {
2295         \str_item:Nn \l_tmpa_str { #1 }
2296     }
2297     \str_case:VnTF \l_tmpb_str {
2298         { X } { } % TODO throw error ?
2299         { i } { \__stex_term_custom_set_X:n { #1 } }
2300         { b } { \__stex_term_custom_set_X:n { #1 } }
2301         { a } { \__stex_term_custom_set_X:n { #1 } } % TODO ?
2302         { B } { \__stex_term_custom_set_X:n { #1 } } % TODO ?
2303     }{}{
2304         % TODO throw error
2305     }
2306
2307     \bool_if:nTF \l_tmpa_bool {
2308         \tl_put_right:Nx \l_tmpa_tl {
2309             \stex_annotate_invisible:n {
2310                 \stex_term_arg:nn { \int_eval:n { #1 } }
2311                 \exp_not:n { { #2 } }
2312             }
2313         }
2314     } {
2315         \tl_put_right:Nx \l_tmpa_tl {
2316             \stex_term_arg:nn { \int_eval:n { #1 } }
2317             \exp_not:n { { #2 } }
2318         }

```

```

2319 }
2320
2321 \__stex_term_custom_loop:
2322 }

```

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

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

```

2323 \cs_new_protected:Nn \__stex_term_custom_set_X:n {
2324   \str_set:Nx \l_tmpa_str {
2325     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
2326     X
2327     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
2328   }
2329 }

```

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

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

```

2330 \cs_new_protected:Npn \__stex_term_custom_component:w [ #1 ] {
2331   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
2332   \__stex_term_custom_loop:
2333 }

```

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

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

```

2334 \cs_new_protected:Nn \__stex_term_custom_final: {
2335   \int_compare:nNnTF \l_tmpb_int = 0 {
2336     \exp_args:Nnno \_stex_term_oms:nnn
2337   }{
2338     \str_if_in:NnTF \l_tmpa_str {b} {
2339       \exp_args:Nnno \_stex_term_ombind:nnn
2340     } {
2341       \exp_args:Nnno \_stex_term_oma:nnn
2342     }
2343   }
2344   { \l__stex_term_custom_uri } { \l__stex_term_custom_uri } { \l_tmpa_tl }
2345 }

```

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

**\symref**

\symname

```

2346 \NewDocumentCommand \symref { m m }{
2347   \STEXsymbol{#1}! [#2]
2348 }
2349
2350 \keys_define:nn { stex / symname } {
2351   post      .tl_set_x:N    = \l_stex_symname_post_str
2352 }
2353
2354 \cs_new_protected:Nn \stex_symname_args:n {
2355   \str_clear:N \l_stex_symname_post_str
2356   \keys_set:nn { stex / symname } { #1 }
2357   \exp_args:NNo \str_set:Nn \l_stex_symname_post_str

```

```

2358     \l_stex_symname_post_str
2359 }
2360
2361 \NewDocumentCommand \symname { 0{} m }{
2362   \stex_symname_args:n { #1 }
2363   \stex_get_symbol:n { #2 }
2364   \str_set:Nx \l_tmpa_str {
2365     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
2366   }
2367   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
2368   \exp_args:NNx \use:nn
2369   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }![
2370     \l_tmpa_str \l_stex_symname_post_str
2371   ] }
2372 }

```

(End definition for `\symref` and `\symname`. These functions are documented on page 21.)

## 4.9 Notation Components

```

2373 <@@=stex_notationcomps>

```

`\stex_highlight_term:nn`

```

2374 \latexml_if:F {
2375   \scalatex_if:F{
2376     \RequirePackage{pdfcomment}
2377   }
2378 }
2379
2380 \str_new:N \l__stex_notationcomps_highlight_uri_str
2381 \cs_new_protected:Nn \stex_highlight_term:nn {
2382   \exp_args:Nnx
2383   \use:nn {
2384     \str_set:Nx \l__stex_notationcomps_highlight_uri_str { #1 }
2385     #2
2386   } {
2387     \str_set:Nx \exp_not:N \l__stex_notationcomps_highlight_uri_str
2388       { \l__stex_notationcomps_highlight_uri_str }
2389   }
2390 }
2391
2392 \cs_new_protected:Nn \stex_unhighlight_term:n {
2393   % \latexml_if:TF {
2394   %   #1
2395   % } {
2396   %   \scalatex_if:TF {
2397   %     #1
2398   %   } {
2399     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
2400   % }
2401   % }
2402 }

```

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

```

\comp
\@comp
\@defemph
2403 \cs_new_protected:Npn \comp #1 {
2404   \str_if_empty:NF \l__stex_notationcomps_highlight_uri_str {
2405     \scalatex_if:TF {
2406       \stex_annotate:nnn { comp } { \l__stex_notationcomps_highlight_uri_str } { #1 }
2407     } {
2408       \exp_args:Nnx \@comp { #1 } { \l__stex_notationcomps_highlight_uri_str }
2409     }
2410   }
2411 }
2412
2413 \cs_new_protected:Npn \@comp #1 #2 {
2414   \pdftooltip {
2415     \textcolor{blue}{#1}
2416   } { #2 }
2417 }
2418
2419 \cs_new_protected:Npn \@defemph #1 #2 {
2420   \pdftooltip {
2421     \textbf{\textcolor{magenta}{#1}}
2422   } { #2 }
2423 }

```

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

## `\ellipses`

```

2424 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix
\parrayline
\parraycell
2425 \bool_new:N \l_stex_inparray_bool
2426 \bool_set_false:N \l_stex_inparray_bool
2427 \NewDocumentCommand \parray { m m } {
2428   \begingroup
2429   \bool_set_true:N \l_stex_inparray_bool
2430   \begin{array}{#1}
2431     #2
2432   \end{array}
2433   \endgroup
2434 }
2435
2436 \NewDocumentCommand \prmatrix { m } {
2437   \begingroup
2438   \bool_set_true:N \l_stex_inparray_bool
2439   \begin{matrix}
2440     #1
2441   \end{matrix}
2442   \endgroup
2443 }
2444
2445 \def \parrayline #1 #2 {
2446   #1 #2 \bool_if:NT \l_stex_inparray_bool {\}
2447 }

```

```

2448
2449 \def \parraycell #1 {
2450   #1 \bool_if:NT \l_stex_inparray_bool {&}
2451 }

```

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

## 4.10 Structural Features

```

2452 <@@=stex_features>

```

symboldoc

```

2453 \NewDocumentEnvironment{symboldoc}{m}{
2454   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
2455   \seq_clear:N \l_tmpb_seq
2456   \seq_map_inline:Nn \l_tmpa_seq {
2457     \stex_get_symbol:n { ##1 }
2458     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
2459       \l_stex_get_symbol_uri_str
2460     }
2461   }
2462   \par
2463   \exp_args:Nnnx
2464   \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
2465 }{
2466   \end{stex_annotate_env}
2467 }

```

STEXdefinition

```

2468
2469 \NewDocumentCommand \__stex_features_definiendum:w { 0{} m m } {
2470   \stex_get_symbol:n { ##2 }
2471   \scalatex_if:TF {
2472     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { ##3 }
2473   } {
2474     \exp_args:Nnx \@defemph { ##3 } { \l_stex_get_symbol_uri_str }
2475   }
2476 }
2477 \NewDocumentCommand \__stex_features_definame:w { 0{} m } {
2478   % TODO: root
2479   \stex_get_symbol:n { ##2 }
2480   \str_set:Nx \l_tmpa_str {
2481     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
2482   }
2483   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
2484   \scalatex_if:TF {
2485     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
2486       \l_tmpa_str
2487     }
2488   } {
2489     \@defemph {
2490       \l_tmpa_str
2491     } { \l_stex_get_symbol_uri_str }
2492   }

```



```

2493 }
2494
2495 \cs_new_protected:Nn \__stex_features_defi_begin:n {
2496   \let\definiendum\__stex_features_definiendum:w
2497   \let\definame\__stex_features_definame:w
2498   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
2499   \seq_clear:N \l_tmpb_seq
2500   \seq_map_inline:Nn \l_tmpa_seq {
2501     \stex_get_symbol:n { ##1 }
2502     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
2503       \l_stex_get_symbol_uri_str
2504     }
2505   }
2506   \exp_args:Nnnx
2507   \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
2508 }
2509
2510 \cs_new_protected:Nn \__stex_features_defi_end: {
2511   \end{stex_annotate_env}
2512 }
2513
2514 \NewDocumentEnvironment{STEXdefinition}{ m }{
2515   \__stex_features_defi_begin:n { #1 }
2516 }{
2517   \__stex_features_defi_end:
2518 }

```

\setSTEXdefinition

```

2519 \cs_new_protected:Npn \setSTEXdefinition #1 {
2520   \AddToHook{env/#1/before}[stex]{\__stex_features_defi_begin:n{}}
2521   \AddToHook{env/#1/after}[stex]{\__stex_features_defi_end:}
2522 }

```

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

structural@feature

```

2523
2524 \NewDocumentEnvironment{structural@feature}{ m m m }{
2525   \stex_if_in_module:F {
2526     \msg_set:nnn{stex}{error/nomodule}{
2527       Structural~Feature~has~to~occur~in~a~module:\\
2528       Feature~#2~of~type~#1\\
2529       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
2530     }
2531     \msg_error:nn{stex}{error/nomodule}
2532   }
2533
2534   \str_set:Nx \l_stex_module_name_str {
2535     \prop_item:Nn \l_stex_current_module_prop
2536       { name } / #2 - feature
2537   }
2538
2539
2540   \str_clear:N \l_tmpa_str

```

```

2541 \seq_clear:N \l_tmpa_seq
2542 \tl_clear:N \l_tmpa_tl
2543 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
2544     origname = #2,
2545     name      = \l_stex_module_name_str ,
2546     ns        = \l_stex_module_ns_str ,
2547     imports   = \exp_not:o { \l_tmpa_seq } ,
2548     constants = \exp_not:o { \l_tmpa_seq } ,
2549     content   = \exp_not:o { \l_tmpa_tl } ,
2550     file      = \exp_not:o { \g_stex_currentfile_seq } ,
2551     lang      = \l_stex_module_lang_str ,
2552     sig       = \l_tmpa_str ,
2553     meta      = \l_tmpa_str ,
2554     feature   = #1 ,
2555 }
2556
2557 \stex_if_smsmode:TF {
2558     \stex_smsmode_set_codes:
2559 } {
2560     \begin{stex_annotate_env}{ feature:#1 }{}
2561     \stex_annotate_invisible:nnn{header}{}{ #3 }
2562 }
2563 }{
2564     \str_set:Nx \l_tmpa_str {
2565         c_stex_feature_
2566         \prop_item:Nn \l_stex_current_module_prop { ns } ?
2567         \prop_item:Nn \l_stex_current_module_prop { name }
2568         _prop
2569     }
2570     \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
2571     \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
2572     \stex_if_smsmode:TF {
2573         \exp_args:Nx \stex_addtosms:n {
2574             \prop_gset_from_keyval:cn {
2575                 c_stex_feature_
2576                 \prop_item:Nn \l_stex_current_module_prop { ns } ?
2577                 \prop_item:Nn \l_stex_current_module_prop { name }
2578                 _prop
2579             } {
2580                 origname = #2,
2581                 name      = \prop_item:cn { \l_tmpa_str } { name } ,
2582                 ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
2583                 imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
2584                 constants = \prop_item:cn { \l_tmpa_str } { constants } ,
2585                 content   = \prop_item:cn { \l_tmpa_str } { content } ,
2586                 file      = \prop_item:cn { \l_tmpa_str } { file } ,
2587                 lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
2588                 sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
2589                 meta      = \prop_item:cn { \l_tmpa_str } { meta } ,
2590                 feature   = \prop_item:cn { \l_tmpa_str } { feature }
2591             }
2592         }
2593     } {
2594         \end{stex_annotate_env}

```

```

2595 }
2596 }
2597

```

structure

```

2598
2599 \prop_new:N \l_stex_all_structures_prop
2600
2601 \keys_define:nn { stex / features / structure } {
2602   name          .tl_set_x:N = \l__stex_features_structure_name_str ,
2603 }
2604
2605 \cs_new_protected:Nn \__stex_features_structure_args:n {
2606   \str_clear:N \l__stex_features_structure_name_str
2607   \keys_set:nn { stex / features / structure } { #1 }
2608   \exp_args:NNo \str_set:Nn \l__stex_features_structure_name_str
2609     \l__stex_features_structure_name_str
2610 }
2611
2612 %\stex_new_feature:nnnn { structure } { 0{ } m } {
2613 %   \__stex_features_structure_args:n { ##1 }
2614 %   \str_if_empty:NT \l__stex_features_structure_name_str {
2615 %     \str_set:Nx \l__stex_features_structure_name_str { ##2 }
2616 %   }
2617 %} {
2618 %
2619 %}
2620
2621 \NewDocumentEnvironment{structure}{ 0{ } m }{
2622   \__stex_features_structure_args:n { #1 }
2623   \str_if_empty:NT \l__stex_features_structure_name_str {
2624     \str_set:Nx \l__stex_features_structure_name_str { #2 }
2625   }
2626   \exp_args:Nnnx
2627   \begin{structural@feature}{ structure }
2628     { \l__stex_features_structure_name_str }{}
2629     \seq_clear:N \l_tmpa_seq
2630     \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
2631
2632   }{
2633     \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
2634     \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
2635     \str_set:Nx \l_tmpa_str {
2636       \prop_item:Nn \l_stex_current_module_prop { ns } ?
2637       \prop_item:Nn \l_stex_current_module_prop { name }
2638     }
2639     \seq_map_inline:Nn \l_tmpa_seq {
2640       \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
2641     }
2642     \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
2643     \exp_args:Nnx
2644     \AddToHookNext { env / structure / after }{
2645       \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}}{
2646         \_stex_term_math_oms:nnnn { \l_tmpa_str }{}{}{}{}

```

```

2647     }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }}{ #2 }
2648   \STEXexport {
2649     \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2650       {\prop_item:Nn \l_stex_current_module_prop { origname }}
2651       {\l_tmpa_str}
2652     \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2653       {#2}{\l_tmpa_str}
2654   %   \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
2655   %     \prop_item:Nn \l_stex_current_module_prop { origname },
2656   %     \l_tmpa_str
2657   %   }
2658   %   \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
2659   %     #2,\l_tmpa_str
2660   %   }
2661   %   \tl_set:cx { #2 } {
2662   %     \stex_invoke_structure:n { \l_tmpa_str }
2663   %   }
2664   }
2665
2666   \end{structural@feature}
2667   % \g_stex_last_feature_prop
2668 }

```

\instantiate

```

2669 \seq_new:N \l__stex_features_structure_field_seq
2670 \str_new:N \l__stex_features_structure_field_str
2671 \str_new:N \l__stex_features_structure_def_tl
2672 \prop_new:N \l__stex_features_structure_prop
2673 \NewDocumentCommand \instantiate { m O{} m }{
2674   \stex_smsmode_set_codes:
2675   \prop_get:Nn \l_stex_all_structures_prop {#1} \l_tmpa_str
2676   \prop_set_eq:Nc \l__stex_features_structure_prop {
2677     c_stex_feature_\l_tmpa_str _prop
2678   }
2679   \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
2680   \seq_map_inline:Nn \l__stex_features_structure_field_seq {
2681     \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
2682     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
2683       \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
2684       \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
2685       {!} \l_tmpa_tl
2686       \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
2687         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
2688         \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
2689         \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
2690       }{
2691         \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
2692         \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
2693         \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
2694         \l_tmpa_tl
2695         \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
2696           \seq_get_left:NN \l_tmpb_seq \l_tmpb_tl
2697           \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
2698         }{

```

```

2699         \tl_clear:N \l_tmpb_tl
2700     }
2701 }
2702 }{
2703   \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
2704   \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
2705     \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
2706     \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
2707     \tl_clear:N \l_tmpa_tl
2708   }{
2709     % TODO throw error
2710   }
2711 }
2712 % \l_tmpa_str: name
2713 % \l_tmpa_tl: definiens
2714 % \l_tmpb_tl: notation
2715 \tl_if_empty:NT \l__stex_features_structure_field_str {
2716   % TODO throw error
2717 }
2718 \str_clear:N \l_tmpb_str
2719
2720 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
2721 \seq_map_inline:Nn \l_tmpa_seq {
2722   \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
2723   \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
2724   \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
2725     \seq_map_break:n {
2726       \str_set:Nn \l_tmpb_str { ####1 }
2727     }
2728   }
2729 }
2730 \prop_get:cnN { g_stex_symdecl_ \l_tmpb_str _prop } {args}
2731 \l_tmpb_str
2732
2733 \tl_if_empty:NTF \l_tmpb_tl {
2734   \tl_if_empty:NF \l_tmpa_tl {
2735     \exp_args:Nx \use:n {
2736       \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fe
2737     }
2738   }
2739 }{
2740   \tl_if_empty:NTF \l_tmpa_tl {
2741     \exp_args:Nx \use:n {
2742       \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\
2743     }
2744   }{
2745     \exp_args:Nx \use:n {
2746       \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
2747       \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
2748     }
2749   }
2750 }
2751 }
2752 % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?

```

```

2753 % \prop_item:Nn \l_stex_current_module_prop {name} ?
2754 % #3/\l__stex_features_structure_field_str
2755 % \par
2756 % \expandafter\present\csname
2757 % g_stex_symdecl_
2758 % \prop_item:Nn \l_stex_current_module_prop {ns} ?
2759 % \prop_item:Nn \l_stex_current_module_prop {name} ?
2760 % #3/\l__stex_features_structure_field_str
2761 % _prop
2762 % \endcsname
2763 }
2764
2765 \tl_clear:N \l__stex_features_structure_def_tl
2766
2767 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
2768 \seq_map_inline:Nn \l_tmpa_seq {
2769   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
2770   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
2771   \exp_args:Nx \use:n {
2772     \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
2773
2774     }
2775   }
2776
2777   \prop_if_exist:cF {
2778     g_stex_symdecl_
2779     \prop_item:Nn \l_stex_current_module_prop {ns} ?
2780     \prop_item:Nn \l_stex_current_module_prop {name} ?
2781     #3/\l_tmpa_str
2782     _prop
2783   }{
2784     \prop_get:cnN { g_stex_symdecl_ ##1 _prop } {args}
2785     \l_tmpb_str
2786     \exp_args:Nx \use:n {
2787       \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
2788     }
2789   }
2790 }
2791
2792 \symdecl*[type={\STEXsymbol{module-type}}{
2793   \_stex_term_math_oms:nnnn {
2794     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
2795     \prop_item:Nn \l__stex_features_structure_prop {name}
2796     }{}{0}{}
2797   }{}{#3}
2798
2799 % TODO: -> sms file
2800
2801 \tl_set:cx{ #3 }{
2802   \stex_invoke_structure:nnn {
2803     \prop_item:Nn \l_stex_current_module_prop {ns} ?
2804     \prop_item:Nn \l_stex_current_module_prop {name} ? #3
2805   } {
2806     \prop_item:Nn \l__stex_features_structure_prop {ns} ?

```

```

2807     \prop_item:Nn \l__stex_features_structure_prop {name}
2808   }
2809 }
2810
2811 }

```

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

\stex\_invoke\_structure:nnn

```

2812 % #1: URI of the instance
2813 % #2: URI of the instantiated module
2814 \cs_new_protected:Nn \stex_invoke_structure:nnn {
2815   \tl_if_empty:nTF{ #3 }{
2816     \prop_set_eq:Nc \l__stex_features_structure_prop {
2817       c_stex_feature_ #2 _prop
2818     }
2819     \tl_clear:N \l_tmpa_tl
2820     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
2821     \seq_map_inline:Nn \l_tmpa_seq {
2822       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
2823       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
2824       \cs_if_exist:cT {
2825         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str_cs
2826       }{
2827         \tl_if_empty:NF \l_tmpa_tl {
2828           \tl_put_right:Nn \l_tmpa_tl {,}
2829         }
2830         \tl_put_right:Nx \l_tmpa_tl {
2831           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
2832         }
2833       }
2834     }
2835     \scalatexBREAK
2836     \exp_args:No \mathstrut \l_tmpa_tl
2837   }{
2838     \stex_invoke_symbol:n{#1/#3}
2839   }
2840 }

```

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

## 4.11 Put these somewhere

\MSC

```

2841 \NewDocumentCommand \MSC {m} {
2842   % TODO
2843 }

```

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

```

2844 \@ifpackageloaded{tikzinput}{
2845   \RequirePackage{stex-tikzinput}
2846 }{}
2847
2848 \AddToHook{begindocument}{

```

```

2849 \input{stex-metattheory}
2850 }
2851 </package>

```

## 4.12 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.

```

2852 <*metatheory>
2853 \ExplSyntaxOn
2854 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
2855 \begin{@module}[ns=\c_stex_metatheory_ns_str,meta=NONE]{Metatheory}
2856 \ExplSyntaxOff
2857
2858 % is-a (a:A, a \in A, a is an A, etc.)
2859 \symdecl[args=ai]{isa}
2860 \notation[typed]{isa}{#1 \comp: #2}{#1 \comp, #2}
2861 \notation[in]{isa}{#1 \comp\in #2}{#1 \comp, #2}
2862 \notation[pred]{isa}{#2\comp(#1 \comp)}{#1 \comp, #2}
2863
2864 % bind (\forall, \Pi, \lambda etc.)
2865 \symdecl[args=Bi]{bind}
2866 \notation[forall]{bind}{\comp\forall #1.;#2}{#1 \comp, #2}
2867 \notation[Pi]{bind}{\comp\prod_{#1}#2}{#1 \comp, #2}
2868 \notation[depfun]{bind}{\comp(#1 \comp)\;\to\;}{#1 \comp, #2}
2869
2870 % dummy variable
2871 \symdecl{dummyvar}
2872 \notation[underscore]{dummyvar}{\comp\_}
2873 \notation[dot]{dummyvar}{\comp\cdot}
2874 \notation[dot]{dummyvar}{\comp\cdot}
2875 \notation[dash]{dummyvar}{\comp{\rm --}}
2876
2877 %fromto (function space, Hom-set, implication etc.)
2878 \symdecl[args=ai]{fromto}
2879 \notation[xarrow]{fromto}{#1 \comp\to #2}{#1 \comp\times #2}
2880 \notation[arrow]{fromto}{#1 \comp\to #2}{#1 \comp\to #2}
2881
2882 % mapto (lambda etc.)
2883 \symdecl[args=Bi]{mapto}
2884 \notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
2885 \notation[lambda]{mapto}{\comp\lambda #1 \comp.; #2}{#1 \comp, #2}
2886 \notation[lambdau]{mapto}{\comp\lambda_{#1} \comp.; #2}{#1 \comp, #2}
2887
2888 % function/operator application
2889 \symdecl[args=ia]{apply}

```



```

2890 \notation[prec=0;0x\neginfprec,parens]{apply}{#1 \comp( #2 \comp)}{#1 \comp, #2}
2891 \notation[prec=0;0x\neginfprec,lambda]{apply}{#1 \; #2 }{#1 \; #2}
2892
2893 % ‘‘type’’ of all collections (sets, classes, types, kinds)
2894 \symdecl{collection}
2895 \notation[U]{collection}{\comp{\mathcal{U}}}
2896 \notation[set]{collection}{\comp{\textsf{Set}}}
2897
2898 % sequences
2899 \symdecl[args=1]{seqtype}
2900 \notation[kleene]{seqtype}{#1^{\comp\ast}}
2901
2902 \symdef[args=2,li]{sequence-index}{#1_{#2}}
2903 \notation[ui]{sequence-index}{#1^{#2}}
2904
2905 %\symdef[args=3,li]{sequence-from-to}{#1_{#2}\comp{,\ellipses},#1_{#3}}
2906 %\notation[ui]{sequence-from-to}{#1^{#2}\comp{,\ellipses},#1^{#3}}
2907 % ^ superceded by \aseqfromto and \livar/\uivar
2908
2909 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{,\ellipses}}{#1\comp,#2}
2910 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{,\ellipses\comp,#2 }{#1\comp,#2}
2911
2912 % letin (‘‘let’’, local definitions, variable substitution)
2913 \symdecl[args=bii]{letin}
2914 \notation[let]{letin}{\comp{\rm let}}{#1\comp{=}#2\; \comp{\rm in}}{#3}
2915 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
2916 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
2917
2918 % structures
2919 \symdecl*[args=1]{module-type}
2920 \notation{module-type}{\mathtt{MOD} #1}
2921 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
2922 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{#1 \comp, #2}
2923
2924 \STEXexport{
2925   \let\nappa\apply
2926   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
2927   \def\livar{\csname sequence-index\endcsname[li]}
2928   \def\uivar{\csname sequence-index\endcsname[ui]}
2929   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
2930   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
2931 }
2932
2933 \end{@module}
2934 \ExplSyntaxOff
2935 \metatheory

```

## 4.13 Auxiliary Packages

### 4.13.1 tikzinput

```

2936 \tikzinput
2937 \tikzinput
2938 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}

```

```

2939 \RequirePackage{l3keys2e}
2940
2941 \keys_define:nn { tikzinput } {
2942   image .bool_set:N = \c_tikzinput_image_bool
2943 }
2944
2945 \ProcessKeysOptions { tikzinput }
2946
2947 \bool_if:NTF \c_tikzinput_image_bool {
2948   \RequirePackage{graphicx}
2949
2950   \providecommand\usetikzlibrary[]{}
2951   \newcommand\tikzinput [2] []{\includegraphics[#1]{#2}}
2952 }{
2953   \RequirePackage{tikz}
2954   \RequirePackage{standalone}
2955
2956   \newcommand \tikzinput [2] [] {
2957     \setkeys{Gin}{#1}
2958     \ifx \Gin@width \Gin@exclamation
2959       \ifx \Gin@height \Gin@exclamation
2960         \input { #2 }
2961       \else
2962         \resizebox{!}{ \Gin@height }{
2963           \input { #2 }
2964         }
2965       \fi
2966     \else
2967       \ifx \Gin@height \Gin@exclamation
2968         \resizebox{ \Gin@width }{!}{
2969           \input { #2 }
2970         }
2971       \else
2972         \resizebox{ \Gin@width }{ \Gin@height }{
2973           \input { #2 }
2974         }
2975       \fi
2976     \fi
2977   }
2978 }
2979
2980 \newcommand \ctikzinput [2] [] {
2981   \begin{center}
2982     \tikzinput [#1] {#2}
2983   \end{center}
2984 }
2985
2986 \@ifpackageloaded{stex}{
2987   \RequirePackage{stex-tikzinput}
2988 }{}
2989 </tikzinput>
2990 <*stex-tikzinput>
2991 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
2992 \RequirePackage{stex}

```

```

2993 \RequirePackage{tikzinput}
2994
2995 % TODO
2996
2997 \end{stex-tikzinput}

```

#### 4.13.2 sTeX1 Compatibility

```

2998 \begin{smglom}
2999 \RequirePackage{expl3,l3keys2e}
3000 \ProvidesExplClass{smglom}{2021/08/01}{1.9}{sTeX1 compatibility}
3001 \LoadClass[border=1px,varwidth]{standalone}
3002 \setlength\textwidth{15cm}
3003 %\g@addto@macro{\@parboxrestore}{\setlength\parskip{\baselineskip}}
3004 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
3005 \ProcessOptions
3006
3007 \RequirePackage{stex-compatibility}
3008 \end{smglom}
3009
3010 \begin{compat}
3011 \@@=stex_deprec
3012 \ProvidesExplPackage{stex-compatibility}{2021/08/01}{1.9}{bla}
3013 \RequirePackage[lang={de,en,ro,tr,fr}]{stex}
3014
3015 \NewDocumentEnvironment { mhmodnl } { 0{} m m } {
3016   \msg_set:nnn{stex}{warning/deprecated}{
3017     \\\
3018     Environment~mhmodnl-is~deprected! \\\
3019     Please~update~module~#2~in~file~
3020     \stex_path_to_string:N \g_stex_currentfile_seq!
3021     \\\ \\\
3022   }
3023   \msg_warning:nn{stex}{warning/deprecated}
3024
3025   \begin{module} [#1,lang=#3]{#2}
3026     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
3027     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
3028     \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
3029     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
3030     \input { \stex_path_to_string:N \l_tmpa_seq / \l_tmpa_str.tex }
3031   } {
3032     \end{module}
3033   }
3034
3035 \NewDocumentEnvironment { modsig } { 0{} m } {
3036   \stex_if_in_module:TF {
3037     \prop_get:NnN \l_stex_current_module_prop {name} \l_tmpa_str
3038     \str_set:Nn \l_tmpb_str { #2 }
3039     \str_if_eq:NNTF \l_tmpa_str \l_tmpb_str {
3040       \prop_set_eq:NN \l_modsig_old_module_prop \l_stex_current_module_prop
3041       \begin{@module}{modsig-#2}
3042         % \prop_set_eq:NN \l_stex_current_module_prop \l_modsig_old_module_prop
3043       } {
3044         \begin{@module}{#2}

```

```

3045     }
3046   } {
3047     \begin{@module}{#2}
3048   }
3049 }{
3050   \end{@module}
3051   \AddToHookNext { env / modsig / after }{
3052     \stex_if_in_module:T {
3053       \prop_get:NnN \l_stex_current_module_prop {name} \l_tmpa_str
3054       \str_set:Nn \l_tmpb_str { #2 }
3055       \str_if_eq:NNT \l_tmpa_str \l_tmpb_str {
3056         % \xdef \g_stex_module_after_group_tl {
3057           \stex_if_smsmode:TF {
3058             \exp_args:Nx
3059             \stex_add_to_current_module:n {
3060               \stex_debug:n{Activating~signature~of~#2}
3061               \exp_not:N \prop_item:cn { c_stex_module_
3062                 \prop_item:Nn \l_stex_current_module_prop {ns} ?
3063                 \prop_item:Nn \l_stex_current_module_prop {name}
3064                 / modsig-#2_prop } { content }
3065             }
3066           }
3067         {
3068           \gdef \g_stex_modsig_after_group_tl {
3069             \stex_activate_module:n {
3070               \prop_item:Nn \l_stex_current_module_prop {ns} ?
3071               \prop_item:Nn \l_stex_current_module_prop {name}
3072               / modsig-#2
3073             }
3074           }
3075           \exp_args:Nx
3076           \stex_add_to_current_module:n {
3077             \stex_activate_module:n {
3078               \prop_item:Nn \l_stex_current_module_prop {ns} ?
3079               \prop_item:Nn \l_stex_current_module_prop {name}
3080               / modsig-#2
3081             }
3082           }
3083         }
3084         \aftergroup \g_stex_modsig_after_group_tl
3085       }
3086     }
3087   }
3088 }
3089 }
3090
3091 \cs_new_protected:Npn \gimport {
3092   \peek_charcode_remove:NTF * {
3093     \gimport_do:
3094   } {
3095     \gimport_do:
3096   }
3097 }
3098

```

```

3099 \NewDocumentCommand \gimport_do: { 0{ } m } {
3100   \msg_set:nnn{stex}{warning/deprecated}{
3101     \\
3102     \c_backslash_str gimport~is~deprecated! \\
3103     Please~use~\c_backslash_str importmodule[#1]{#2}~instead!~(in~file~
3104     \stex_path_to_string:N \g_stex_currentfile_seq)
3105     \\ \\
3106   }
3107   \msg_warning:nn{stex}{warning/deprecated}
3108   \importmodule[#1]{#2}
3109 }
3110
3111 \cs_new_protected:Npn \guse {
3112   \peek_charcode_remove:NTF * {
3113     \guse_do:
3114   } {
3115     \guse_do:
3116   }
3117 }
3118
3119 \NewDocumentCommand \guse_do: { 0{ } m } {
3120   \msg_set:nnn{stex}{warning/deprecated}{
3121     \\
3122     \c_backslash_str guse~is~deprecated! \\
3123     Please~use~\c_backslash_str usemodule[#1]{#2}~instead!~(in~file~
3124     \stex_path_to_string:N \g_stex_currentfile_seq)
3125     \\ \\
3126   }
3127   \msg_warning:nn{stex}{warning/deprecated}
3128   \usemodule[#1]{#2}
3129 }
3130
3131 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3132
3133 \cs_new_protected:Npn \symi {
3134   \peek_charcode_remove:NTF * {
3135     \symi_do:
3136   } {
3137     \symi_do:
3138   }
3139 }
3140
3141 \NewDocumentCommand \symi_do: { 0{ } m } {
3142   \msg_set:nnn{stex}{warning/deprecated}{
3143     \\
3144     \c_backslash_str symi~is~deprecated! \\
3145     Please~use~\c_backslash_str symdecl[#1]{#2}~instead!~(in~file~
3146     \stex_path_to_string:N \g_stex_currentfile_seq)
3147     \\ \\
3148   }
3149   \msg_warning:nn{stex}{warning/deprecated}
3150   \symdecl*[#1]{#2}
3151 }
3152

```

```

3153 \cs_new_protected:Npn \symii {
3154   \peek_charcode_remove:NTF * {
3155     \symii_do:
3156   } {
3157     \symii_do:
3158   }
3159 }
3160
3161 \NewDocumentCommand \symii_do: { 0{} m m } {
3162   \msg_set:nnn{stex}{warning/deprecated}{
3163     \\\
3164     \c_backslash_str symii~is~deprecated! \\\
3165     Please~use~\c_backslash_str symdecl[#1]{#2-#3}~instead!~(in~file~
3166     \stex_path_to_string:N \g_stex_currentfile_seq)
3167     \\\ \\\
3168   }
3169   \msg_warning:nn{stex}{warning/deprecated}
3170   \symdecl*{#1}{#2-#3}
3171 }
3172
3173 \cs_new_protected:Npn \symiii {
3174   \peek_charcode_remove:NTF * {
3175     \symiii_do:
3176   } {
3177     \symiii_do:
3178   }
3179 }
3180
3181 \NewDocumentCommand \symiii_do: { 0{} m m m } {
3182   \msg_set:nnn{stex}{warning/deprecated}{
3183     \\\
3184     \c_backslash_str symiii~is~deprecated! \\\
3185     Please~use~\c_backslash_str symdecl[#1]{#2-#3-#4}~instead!~(in~file~
3186     \stex_path_to_string:N \g_stex_currentfile_seq)
3187     \\\ \\\
3188   }
3189   \msg_warning:nn{stex}{warning/deprecated}
3190   \symdecl*{#1}{#2-#3-#4}
3191 }
3192
3193 \keys_define:nn { stex / deprec / defi } {
3194   name .tl_set_x:N = \l_tmpa_str
3195 }
3196
3197 \cs_new_protected:Npn \defi {
3198   \peek_charcode_remove:NTF * {
3199     \defi_do:
3200   } {
3201     \defi_do:
3202   }
3203 }
3204
3205 \NewDocumentCommand \defi_do: { 0{} m } {
3206   \str_clear:N \l_tmpa_str

```

```

3207 \keys_set:nn { stex / deprec / defi } { #1 }
3208
3209 \str_if_empty:NTF \l_tmpa_str {
3210   \msg_set:nnn{stex}{warning/deprecated}{
3211     \\\
3212     \c_backslash_str defi-is~deprecated! \\\
3213     Please~use~\c_backslash_str STEXsymbol{#2}![#2]~instead!~(in~file~
3214     \stex_path_to_string:N \g_stex_currentfile_seq)
3215     \\\ \\\
3216   }
3217   \msg_warning:nn{stex}{warning/deprecated}
3218   \STEXsymbol { #2 }![ \comp{#2} ]
3219 } {
3220   \msg_set:nnn{stex}{warning/deprecated}{
3221     \\\
3222     \c_backslash_str defi-is~deprecated! \\\
3223     Please~use~\c_backslash_str STEXsymbol { \l_tmpa_str }[ #2 ]~instead!~(in~file~
3224     \stex_path_to_string:N \g_stex_currentfile_seq)
3225     \\\ \\\
3226   }
3227   \msg_warning:nn{stex}{warning/deprecated}
3228   \exp_args:No \STEXsymbol { \l_tmpa_str }![ \comp{#2} ]
3229 }
3230 }
3231
3232
3233 \cs_new_protected:Npn \Defi {
3234   \peek_charcode_remove:NTF * {
3235     \Defi_do:
3236   } {
3237     \Defi_do:
3238   }
3239 }
3240
3241 \NewDocumentCommand \Defi_do: { 0{ } m } {
3242   \str_clear:N \l_tmpa_str
3243   \keys_set:nn { stex / deprec / defi } { #1 }
3244
3245   \str_if_empty:NTF \l_tmpa_str {
3246     \msg_set:nnn{stex}{warning/deprecated}{
3247       \\\
3248       \c_backslash_str Defi-is~deprecated! \\\
3249       Please~use~\c_backslash_str STEXsymbol{#2}![\exp_after:wN \stex_capitalize:n #2]~inste
3250       \stex_path_to_string:N \g_stex_currentfile_seq)
3251       \\\ \\\
3252     }
3253     \msg_warning:nn{stex}{warning/deprecated}
3254     \STEXsymbol { #2 }![ \comp{\exp_after:wN \stex_capitalize:n #2} ]
3255   } {
3256     \msg_set:nnn{stex}{warning/deprecated}{
3257       \\\
3258       \c_backslash_str Defi-is~deprecated! \\\
3259       Please~use~\c_backslash_str STEXsymbol { \l_tmpa_str }[ \exp_after:wN \stex_capitalize
3260       \stex_path_to_string:N \g_stex_currentfile_seq)

```

```

3261     \\\ \\\
3262   }
3263   \msg_warning:nn{stex}{warning/deprecated}
3264   \exp_args:No \STEXsymbol { \l_tmpa_str }![ \comp{\exp_after:wN \stex_capitalize:n #2} ]
3265 }
3266 }
3267
3268 \cs_new_protected:Npn \adefi {
3269   \peek_charcode_remove:NTF * {
3270     \adefi_do:
3271   } {
3272     \adefi_do:
3273   }
3274 }
3275
3276 \NewDocumentCommand \adefi_do: { 0{} m m } {
3277   \str_clear:N \l_tmpa_str
3278   \keys_set:nn { stex / deprec / defi } { #1 }
3279
3280   \str_if_empty:NTF \l_tmpa_str {
3281     \msg_set:nnn{stex}{warning/deprecated}{
3282       \\\
3283       \c_backslash_str adefi-is-deprecated! \\\
3284       Please~use~\c_backslash_str STEXsymbol{#3}![#2]~instead!~(in~file~
3285       \stex_path_to_string:N \g_stex_currentfile_seq)
3286       \\\ \\\
3287     }
3288     \msg_warning:nn{stex}{warning/deprecated}
3289     \STEXsymbol { #3 }![ \comp{#2} ]
3290   } {
3291     \msg_set:nnn{stex}{warning/deprecated}{
3292       \\\
3293       \c_backslash_str adefi-is-deprecated! \\\
3294       Please~use~\c_backslash_str STEXsymbol { \l_tmpa_str }[ #2 ]~instead!~(in~file~
3295       \stex_path_to_string:N \g_stex_currentfile_seq)
3296       \\\ \\\
3297     }
3298     \msg_warning:nn{stex}{warning/deprecated}
3299     \exp_args:No \STEXsymbol { \l_tmpa_str }![ \comp{#2} ]
3300   }
3301 }
3302
3303 \cs_new_protected:Npn \defis {
3304   \peek_charcode_remove:NTF * {
3305     \defis_do:
3306   } {
3307     \defis_do:
3308   }
3309 }
3310
3311 \NewDocumentCommand \defis_do: { 0{} m } {
3312   \str_clear:N \l_tmpa_str
3313   \keys_set:nn { stex / deprec / defi } { #1 }
3314

```



```

3315 \str_if_empty:NTF \l_tmpa_str {
3316   \msg_set:nnn{stex}{warning/deprecated}{
3317     \\\
3318     \c_backslash_str defis-is-deprecated! \\\
3319     Please~use~\c_backslash_str STEXsymbol{#2}![#2s]~instead!~(in~file~
3320     \stex_path_to_string:N \g_stex_currentfile_seq)
3321     \\\ \\\
3322   }
3323   \msg_warning:nn{stex}{warning/deprecated}
3324   \STEXsymbol { #2 }![ \comp{#2s} ]
3325 } {
3326   \msg_set:nnn{stex}{warning/deprecated}{
3327     \\\
3328     \c_backslash_str defis-is-deprecated! \\\
3329     Please~use~\c_backslash_str STEXsymbol { \l_tmpa_str }[ #2s ]~instead!~(in~file~
3330     \stex_path_to_string:N \g_stex_currentfile_seq)
3331     \\\ \\\
3332   }
3333   \msg_warning:nn{stex}{warning/deprecated}
3334   \exp_args:No \STEXsymbol { \l_tmpa_str }![ \comp{#2s} ]
3335 }
3336 }
3337
3338 \cs_new_protected:Npn \defii {
3339   \peek_charcode_remove:NTF * {
3340     \defii_do:
3341   } {
3342     \defii_do:
3343   }
3344 }
3345
3346 \NewDocumentCommand \defii_do: { 0{} m m } {
3347   \str_clear:N \l_tmpa_str
3348   \keys_set:nn { stex / deprec / defi } { #1 }
3349   \str_if_empty:NTF \l_tmpa_str {
3350     \msg_set:nnn{stex}{warning/deprecated}{
3351       \\\
3352       \c_backslash_str defii-is-deprecated! \\\
3353       Please~use~\c_backslash_str STEXsymbol{#2~#3}![#2~#3]~instead!~(in~file~
3354       \stex_path_to_string:N \g_stex_currentfile_seq)
3355       \\\ \\\
3356     }
3357     \msg_warning:nn{stex}{warning/deprecated}
3358     \STEXsymbol { #2~#3 }![ \comp{#2~#3} ]
3359   } {
3360     \msg_set:nnn{stex}{warning/deprecated}{
3361       \\\
3362       \c_backslash_str defii-is-deprecated! \\\
3363       Please~use~\c_backslash_str STEXsymbol { \l_tmpa_str }[ #2~#3 ]~instead!~(in~file~
3364       \stex_path_to_string:N \g_stex_currentfile_seq)
3365       \\\ \\\
3366     }
3367     \msg_warning:nn{stex}{warning/deprecated}
3368     \exp_args:No \STEXsymbol { \l_tmpa_str }![ \comp{#2~#3} ]

```

```

3369 }
3370 }
3371
3372
3373 \cs_new_protected:Npn \defiis {
3374   \peek_charcode_remove:NTF * {
3375     \defiis_do:
3376   } {
3377     \defiis_do:
3378   }
3379 }
3380
3381 \NewDocumentCommand \defiis_do: { 0{} m m } {
3382   \str_clear:N \l_tmpa_str
3383   \keys_set:nn { stex / deprec / defi } { #1 }
3384   \str_if_empty:NTF \l_tmpa_str {
3385     \msg_set:nnn{stex}{warning/deprecated}{
3386       \\\
3387       \c_backslash_str defiis-is-deprecated! \\\
3388       Please~use~\c_backslash_str STEXsymbol{#2~#3}![#2~#3s]~instead!~(in~file~
3389       \stex_path_to_string:N \g_stex_currentfile_seq)
3390       \\\ \\\
3391     }
3392     \msg_warning:nn{stex}{warning/deprecated}
3393     \STEXsymbol { #2~#3 }![ \comp{#2~#3s} ]
3394   } {
3395     \msg_set:nnn{stex}{warning/deprecated}{
3396       \\\
3397       \c_backslash_str defiis-is-deprecated! \\\
3398       Please~use~\c_backslash_str STEXsymbol { \l_tmpa_str }[ #2~#3s ]~instead!~(in~file~
3399       \stex_path_to_string:N \g_stex_currentfile_seq)
3400       \\\ \\\
3401     }
3402     \msg_warning:nn{stex}{warning/deprecated}
3403     \exp_args:No \STEXsymbol { \l_tmpa_str }![ \comp{#2~#3s} ]
3404   }
3405 }
3406
3407
3408 \cs_new_protected:Npn \defiii {
3409   \peek_charcode_remove:NTF * {
3410     \defiii_do:
3411   } {
3412     \defiii_do:
3413   }
3414 }
3415
3416 \NewDocumentCommand \defiii_do: { 0{} m m m } {
3417   \str_clear:N \l_tmpa_str
3418   \keys_set:nn { stex / deprec / defi } { #1 }
3419   \str_if_empty:NTF \l_tmpa_str {
3420     \msg_set:nnn{stex}{warning/deprecated}{
3421       \\\
3422       \c_backslash_str defiii-is-deprecated! \\\

```

```

3423     Please~use~\c_backslash_str STEXsymbol{#2~#3~#4}![#2~#3~#4]~instead!~(in~file~
3424     \stex_path_to_string:N \g_stex_currentfile_seq)
3425     \\\ \\\
3426   }
3427   \msg_warning:nn{stex}{warning/deprecated}
3428   \STEXsymbol { #2~#3~#4 }![ \comp{#2~#3~#4} ]
3429 } {
3430   \msg_set:nnn{stex}{warning/deprecated}{
3431     \\\
3432     \c_backslash_str defiii~is-deprecated! \\\
3433     Please~use~\c_backslash_str STEXsymbol { \l_tmpa_str }[ #2~#3~#4 ]~instead!~(in~file~
3434     \stex_path_to_string:N \g_stex_currentfile_seq)
3435     \\\ \\\
3436   }
3437   \msg_warning:nn{stex}{warning/deprecated}
3438   \exp_args:No \STEXsymbol { \l_tmpa_str }![ \comp{#2~#3~#4} ]
3439 }
3440 }
3441
3442 %\RequirePackage[hyperref]{ntheorem}
3443 %\theoremstyle{plain}
3444 %\RequirePackage{amsthm}
3445
3446 \NewDocumentEnvironment {definition} { 0{} } {
3447   \begin{STEXdefinition}{}
3448 }{
3449   \end{STEXdefinition}
3450 }
3451 \keys_define:nn { stex / omtex } {
3452   title .tl_set_x:n = \l_stex_omtext_title_str
3453 }
3454 \cs_new_protected:Nn \stex_omtext_args:n {
3455   \str_clear:N \l_stex_omtext_title_str
3456   \keys_set:nn { stex / omtex }{ #1 }
3457   \exp_args:NNo \str_set:Nn \l_stex_omtext_title_str
3458     \l_stex_omtext_title_str
3459 }
3460 \NewDocumentEnvironment {omtext} { 0{} } {
3461   \stex_omtext_args:n { #1 }
3462   \paragraph{\l_stex_omtext_title_str}
3463 }{
3464
3465 }
3466 \NewDocumentEnvironment {assertion} { 0{} } {
3467
3468 }{
3469
3470 }
3471
3472 \NewDocumentCommand \inlinedef { m } {
3473   \begingroup
3474   \let\definiendum\_stex_deprec_definiendum:w
3475   \let\definame\_stex_deprec_definame:w
3476   #1

```

```

3477 \endgroup
3478 }
3479
3480 \NewDocumentCommand \inlineass { m } { #1 }
3481
3482 \NewDocumentCommand \trefi { 0{} m } {
3483   \str_set:Nn \l_tmpa_str { #1 }
3484   \str_if_empty:NTF \l_tmpa_str {
3485     \msg_set:nnn{stex}{warning/deprecated}{
3486       \\\
3487       \c_backslash_str trefi-is-deprecated! \\\
3488       Please~use~\c_backslash_str STExsymbol{#2}! [#2]~instead!~(in~file~
3489       \stex_path_to_string:N \g_stex_currentfile_seq)
3490       \\\ \\\
3491     }
3492     \msg_warning:nn{stex}{warning/deprecated}
3493     \STExsymbol { #2 }! [ \comp{#2} ]
3494   } {
3495     \msg_set:nnn{stex}{warning/deprecated}{
3496       \\\
3497       \c_backslash_str trefi-is-deprecated! \\\
3498       Please~use~\c_backslash_str STExsymbol { #1?#2 } [ #2 ]~instead!~(in~file~
3499       \stex_path_to_string:N \g_stex_currentfile_seq)
3500       \\\ \\\
3501     }
3502     \msg_warning:nn{stex}{warning/deprecated}
3503     \STExsymbol { #1 }! [ \comp{#2} ]
3504   }
3505 }
3506
3507
3508 \NewDocumentCommand \Trefi { 0{} m } {
3509   \str_set:Nn \l_tmpa_str { #1 }
3510   \str_if_empty:NTF \l_tmpa_str {
3511     \msg_set:nnn{stex}{warning/deprecated}{
3512       \\\
3513       \c_backslash_str Trefi-is-deprecated! \\\
3514       Please~use~\c_backslash_str STExsymbol{#2}! [\exp_after:wN \stex_capitalize:n #2]~inste
3515       \stex_path_to_string:N \g_stex_currentfile_seq)
3516       \\\ \\\
3517     }
3518     \msg_warning:nn{stex}{warning/deprecated}
3519     \STExsymbol { #2 }! [ \comp{\exp_after:wN \stex_capitalize:n #2} ]
3520   } {
3521     \msg_set:nnn{stex}{warning/deprecated}{
3522       \\\
3523       \c_backslash_str Trefi-is-deprecated! \\\
3524       Please~use~\c_backslash_str STExsymbol { #1 } [ \exp_after:wN \stex_capitalize:n #2 ]~i
3525       \stex_path_to_string:N \g_stex_currentfile_seq)
3526       \\\ \\\
3527     }
3528     \msg_warning:nn{stex}{warning/deprecated}
3529     \STExsymbol { #1 }! [ \comp{\exp_after:wN \stex_capitalize:n #2} ]
3530   }

```

```

3531 }
3532
3533 \NewDocumentCommand \trefis { 0{} m } {
3534   \str_set:Nn \l_tmpa_str { #1 }
3535   \str_if_empty:NTF \l_tmpa_str {
3536     \msg_set:nnn{stex}{warning/deprecated}{
3537       \\
3538       \c_backslash_str trefi-is-deprecated! \\
3539       Please~use~\c_backslash_str STEXsymbol{#2}![#2s]~instead!~(in~file~
3540       \stex_path_to_string:N \g_stex_currentfile_seq)
3541       \\ \\
3542     }
3543     \msg_warning:nn{stex}{warning/deprecated}
3544     \STEXsymbol { #2 }![ \comp{#2s} ]
3545   } {
3546     \msg_set:nnn{stex}{warning/deprecated}{
3547       \\
3548       \c_backslash_str trefi-is-deprecated! \\
3549       Please~use~\c_backslash_str STEXsymbol { #1 }[ #2s ]~instead!~(in~file~
3550       \stex_path_to_string:N \g_stex_currentfile_seq)
3551       \\ \\
3552     }
3553     \msg_warning:nn{stex}{warning/deprecated}
3554     \STEXsymbol { #1 }![ \comp{#2s} ]
3555   }
3556 }
3557
3558
3559 \NewDocumentCommand \Trefis { 0{} m } {
3560   \str_set:Nn \l_tmpa_str { #1 }
3561   \str_if_empty:NTF \l_tmpa_str {
3562     \msg_set:nnn{stex}{warning/deprecated}{
3563       \\
3564       \c_backslash_str Trefis-is-deprecated! \\
3565       Please~use~\c_backslash_str STEXsymbol{#2}![\exp_after:wN \stex_capitalize:n #2s]~inst
3566       \stex_path_to_string:N \g_stex_currentfile_seq)
3567       \\ \\
3568     }
3569     \msg_warning:nn{stex}{warning/deprecated}
3570     \STEXsymbol { #2 }![ \comp{\exp_after:wN \stex_capitalize:n #2s} ]
3571   } {
3572     \msg_set:nnn{stex}{warning/deprecated}{
3573       \\
3574       \c_backslash_str Trefis-is-deprecated! \\
3575       Please~use~\c_backslash_str STEXsymbol { #1 }[ \exp_after:wN \stex_capitalize:n #2s ]~
3576       \stex_path_to_string:N \g_stex_currentfile_seq)
3577       \\ \\
3578     }
3579     \msg_warning:nn{stex}{warning/deprecated}
3580     \STEXsymbol { #1 }![ \comp{\exp_after:wN \stex_capitalize:n #2s} ]
3581   }
3582 }
3583
3584 \NewDocumentCommand \trefii { 0{} m m } {

```

```

3585 \str_set:Nn \l_tmpa_str { #1 }
3586 \str_if_empty:NTF \l_tmpa_str {
3587   \msg_set:nnn{stex}{warning/deprecated}{
3588     \\\
3589     \c_backslash_str trefii~is-deprecated! \\\
3590     Please~use~\c_backslash_str STEXsymbol{#2~#3}![#2~#3]~instead!~(in~file~
3591     \stex_path_to_string:N \g_stex_currentfile_seq)
3592     \\\ \\\
3593   }
3594   \msg_warning:nn{stex}{warning/deprecated}
3595   \STEXsymbol { #2~#3 }![ \comp{#2~#3} ]
3596 } {
3597   \msg_set:nnn{stex}{warning/deprecated}{
3598     \\\
3599     \c_backslash_str trefii~is-deprecated! \\\
3600     Please~use~\c_backslash_str STEXsymbol { #1 }[ #2~#3 ]~instead!~(in~file~
3601     \stex_path_to_string:N \g_stex_currentfile_seq)
3602     \\\ \\\
3603   }
3604   \msg_warning:nn{stex}{warning/deprecated}
3605   \STEXsymbol { #1 }![ \comp{#2~#3} ]
3606 }
3607 }
3608
3609 \NewDocumentCommand \trefiii { 0{ } m m m } {
3610   \str_set:Nn \l_tmpa_str { #1 }
3611   \str_if_empty:NTF \l_tmpa_str {
3612     \msg_set:nnn{stex}{warning/deprecated}{
3613       \\\
3614       \c_backslash_str trefiii~is-deprecated! \\\
3615       Please~use~\c_backslash_str STEXsymbol{#2~#3~#4}![#2~#3~#4]~instead!~(in~file~
3616       \stex_path_to_string:N \g_stex_currentfile_seq)
3617       \\\ \\\
3618     }
3619     \msg_warning:nn{stex}{warning/deprecated}
3620     \STEXsymbol { #2~#3~#4 }![ \comp{#2~#3~#4} ]
3621   } {
3622     \msg_set:nnn{stex}{warning/deprecated}{
3623       \\\
3624       \c_backslash_str trefiii~is-deprecated! \\\
3625       Please~use~\c_backslash_str STEXsymbol { #1 }[ #2~#3~#4 ]~instead!~(in~file~
3626       \stex_path_to_string:N \g_stex_currentfile_seq)
3627       \\\ \\\
3628     }
3629     \msg_warning:nn{stex}{warning/deprecated}
3630     \STEXsymbol { #1 }![ \comp{#2~#3~#4} ]
3631   }
3632 }
3633
3634
3635 \NewDocumentCommand \trefiis { 0{ } m m } {
3636   \str_set:Nn \l_tmpa_str { #1 }
3637   \str_if_empty:NTF \l_tmpa_str {
3638     \msg_set:nnn{stex}{warning/deprecated}{

```

```

3639     \
3640     \c_backslash_str trefiis~is-deprecated! \
3641     Please~use~\c_backslash_str STExsymbol{#2~#3}![#2~#3s]~instead!~(in~file~
3642     \stex_path_to_string:N \g_stex_currentfile_seq)
3643     \
3644   }
3645   \msg_warning:nn{stex}{warning/deprecated}
3646   \STExsymbol { #2~#3 }![ \comp{#2~#3s} ]
3647 } {
3648   \msg_set:nnn{stex}{warning/deprecated}{
3649     \
3650     \c_backslash_str trefiis~is-deprecated! \
3651     Please~use~\c_backslash_str STExsymbol { #1 }[ #2~#3s ]~instead!~(in~file~
3652     \stex_path_to_string:N \g_stex_currentfile_seq)
3653     \
3654   }
3655   \msg_warning:nn{stex}{warning/deprecated}
3656   \STExsymbol { #1 }![ \comp{#2~#3s} ]
3657 }
3658 }
3659
3660 \NewDocumentCommand \symvariant { O{} m O{0} m m } {
3661   \msg_set:nnn{stex}{warning/deprecated}{
3662     \
3663     \c_backslash_str symvariant~is-deprecated! \
3664     Please~use~\c_backslash_str notation[#4]{ #2 }~instead!~(in~file~
3665     \stex_path_to_string:N \g_stex_currentfile_seq)
3666     \
3667   }
3668   \msg_warning:nn{stex}{warning/deprecated}
3669
3670   \notation[variant=#4]{#2}{#5}
3671 }
3672
3673 \NewDocumentCommand \mixfixi { O{} m m m } {
3674   \msg_set:nnn{stex}{warning/deprecated}{
3675     \c_backslash_str mixfixi~is-fatally-deprecated!\
3676     Symbol:~\l__stex_term_highlight_uri_str\
3677     Current~file:~\stex_path_to_string:N \g_stex_currentfile_seq
3678   }
3679   \msg_error:nn{stex}{warning/deprecated}
3680 }
3681
3682
3683 \NewDocumentCommand \infix {} {
3684   \msg_set:nnn{stex}{warning/deprecated}{
3685     \c_backslash_str infix~is-fatally-deprecated!\
3686     Symbol:~\l__stex_term_highlight_uri_str\
3687     Current~file:~\stex_path_to_string:N \g_stex_currentfile_seq
3688   }
3689   \msg_error:nn{stex}{warning/deprecated}
3690 }
3691
3692 \let\iprec\infix

```

```

3693
3694 \NewDocumentCommand \inlineex { m } {
3695   \msg_set:nnn{stex}{warning/deprecated}{
3696     \c_backslash_str inlineex~is~deprecated!\\
3697     No~replacement~exists~yet.\\
3698     Current~file:~\stex_path_to_string:N \g_stex_currentfile_seq
3699   }
3700   \msg_warning:nn{stex}{warning/deprecated}
3701   #1
3702 }
3703
3704
3705 \NewDocumentCommand \term { m } {
3706   \msg_set:nnn{stex}{warning/deprecated}{
3707     \c_backslash_str term~is~deprecated!\\
3708     No~replacement~exists~yet.\\
3709     Current~file:~\stex_path_to_string:N \g_stex_currentfile_seq
3710   }
3711   \msg_warning:nn{stex}{warning/deprecated}
3712   #1
3713 }
3714
3715
3716 \NewDocumentCommand \Definame { O{} m } {
3717   \stex_get_symbol:n { #2 }
3718   \str_set:Nx \l_tmpa_str {
3719     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3720   }
3721   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3722   \scalatex_if:TF {
3723     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3724       \l_tmpa_str
3725     }
3726   } {
3727     \@defemph {
3728       \exp_after:wN \stex_capitalize:n \l_tmpa_str
3729     } { \l_stex_get_symbol_uri_str }
3730   }
3731 }
3732
3733 \NewDocumentCommand \Definiendum { O{} m m } {
3734   \stex_get_symbol:n { #2 }
3735   \str_set:Nx \l_tmpa_str {
3736     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3737   }
3738   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3739   \scalatex_if:TF {
3740     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3741       \l_tmpa_str
3742     }
3743   } {
3744     \@defemph {
3745       \exp_after:wN \stex_capitalize:n \l_tmpa_str
3746     } { \l_stex_get_symbol_uri_str }

```



```

3747 }
3748 }
3749
3750 \NewDocumentCommand \Symname { 0{} m }{
3751   \stex_symname_args:n { #1 }
3752   \stex_get_symbol:n { #2 }
3753   \str_set:Nx \l_tmpa_str {
3754     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3755   }
3756   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3757   \exp_args:NNx \use:nn
3758   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }![
3759     \exp_after:wN \stex_capitalize:n \l_tmpa_str
3760     \l_stex_symname_post_str
3761   ] }
3762 }
3763
3764
3765 \seq_gput_right:Nx \g_stex_smsmode_allowedenvs_seq { \tl_to_str:n { mhmodnl } }
3766 \seq_gput_right:Nx \g_stex_smsmode_allowedenvs_seq { \tl_to_str:n { modsig } }
3767 \tl_gput_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\gimport\syml\symii\symiii\symiv\
3768
3769 % omtex:
3770 \cs_new_protected:Npn \lec #1 {
3771   \strut\hfil\strut\hfill(#1)
3772 }
3773 \cs_new_protected:Npn \nlex #1 {
3774   \textcolor{green}{\s1 #1}}
3775 }
3776
3777
3778 </compat>

```