

stex.sty: \TeX 2.0*

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

November 25, 2021

Abstract

TODO

1 Introduction

TODO

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

Contents

1	Introduction	1
2	Manual	3
2.1	Modules	3
2.2	Semantic Macros and Notations	3
2.3	Archives and Imports	7
3	Documentation	8
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
4	Implementation	25
4.1	The \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	90

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

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

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

Example 2

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

$(a \cdot b)$ and $(a \times b)$

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

¹EdNOTE: 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

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

³EDNOTE: “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 suffix¹.

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

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.

¹which 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 {<message>}</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^AT_EXML or S_CA_LT_EX) with attributes:

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

Annotates the HTML generated by `⟨content⟩` with

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

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 module-environment without a header.
----------------------	---

Test 4

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

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

Test 5

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

```
Module 3.1[Bar] (FooBar)
Module path: http://mathhub.info/tests/Foo/Bar/Foo?Bar
Language:
Signature:
Metatheory:
```

\l_stex_all_modules_seq

Stores full URIs for all modules currently in scope.

\STEXModule

\STEXModule {*<fragment>*}

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

`\stex_invoke_module:n`

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

Test 6

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

Module 3.2[STEXModuleTest1]

Module 3.3[STEXModuleTest2]

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

3.4.2 SMS Mode

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

`\g_stex_smsmode_allowedmacros_tl`

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

`\g_stex_smsmode_allowedmacros_escape_tl`

Macros that are executed with the category codes restored.

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

`\g_stex_smsmode_allowedenvs_seq`

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

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

Tests whether SMS mode is currently active.

`\stex_smsmode_set_codes:`

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

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

`\stex_in_smsmode:nn`

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

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

Test 7

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

3.4.3 Imports and Inheritance

`\importmodule`

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

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

Test 8

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

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

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

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

```
Module 3.7[Importtest2]
Meaning: >macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?Foo?foo}<
```

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

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

Test 9

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

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

Module 3.8[UseTest1]

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

Module 3.10[UseTest3]
Meaning: »undefined<
Meaning: »macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?UseTest2?bar}<
All modules: file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?UseTest3, http://mathhub.info/sTeX?Metath
file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?UseTest2
All symbols: http://mathhub.info/sTeX?Metatheory?isa, http://mathhub.info/sTeX?Metatheory?bind, http://mathhub.info/sTeX?Metatheo
http://mathhub.info/sTeX?Metatheory?fromto, http://mathhub.info/sTeX?Metatheory?apply, http://mathhub.info/sTeX?Metatheory?collec
http://mathhub.info/sTeX?Metatheory?seqtype, http://mathhub.info/sTeX?Metatheory?sequence-index, http://mathhub.info/sTeX?Metath
http://mathhub.info/sTeX?Metatheory?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 {<archive-ID>} {<module-path>}</code>
	Determines the URI of a module by splitting <code><module-path></code> into <code><path>?<name></code> . If <code><module-path></code> does <i>not</i> contain a <code>?</code> -character, we consider it to be the <code><name></code> , and <code><path></code> to be empty. If <code><archive-ID></code> is empty, it is automatically set to the ID of the current archive (if one exists).
	1. If <code><archive-ID></code> is empty:
	(a) If <code><path></code> is empty, then <code><name></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><name>.<lang>.tex</code> must exist in the same folder, containing a module <code><name></code> . That module should have the same namespace as the current one.
	(b) If <code><path></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><path></code> is empty, then <code><name></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><name>.<lang>.tex</code> must exist in the top <code>source</code> folder of the archive, containing a module <code><name></code> . That module should lie directly in the namespace of the archive.
	(b) If <code><path></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>{<ns>} {<archive-ID>} {<path>} {<name>}</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.

<hr/> <hr/>	<code>\stex_term_math_arg:nnn</code>	<code>\stex_term_arg:nnn⟨int⟩⟨prec⟩⟨body⟩</code>
		Annotates $\langle body \rangle$ as the $\langle int \rangle$ th argument of the current OMA or OMBIND, with (downwards) argument precedence $\langle prec \rangle$.

<hr/> <hr/>	<code>\stex_term_math_assoc_arg:nnnn</code>	<code>\stex_term_arg:nnn⟨int⟩⟨prec⟩⟨notation⟩⟨body⟩</code>
		Annotates $\langle body \rangle$ as the $\langle int \rangle$ th (associative) <i>sequence</i> argument (as comma-separated list of terms) of the current OMA or OMBIND, with (downwards) argument precedence $\langle prec \rangle$ and associative notation $\langle notation \rangle$.

<hr/>	<code>\infprec</code>	Maximal and minimal notation precedences.
<hr/>	<code>\neginfprec</code>	

<hr/> <hr/>	<code>\dobrackets</code>	<code>\dobrackets {⟨body⟩}</code>
		Puts $\langle body \rangle$ in parentheses; scaled if in display mode unscaled otherwise. Uses the current \S\TeX brackets (by default (and)), which can be changed temporarily using <code>\withbrackets</code> .

<hr/> <hr/>	<code>\withbrackets</code>	<code>\withbrackets ⟨left⟩ ⟨right⟩ {⟨body⟩}</code>
		Temporarily (i.e. within $\langle body \rangle$) sets the brackets used by \S\TeX for automated bracketing (by default (and)) to $\langle left \rangle$ and $\langle right \rangle$. Note that $\langle left \rangle$ and $\langle right \rangle$ need to be allowed after <code>\left</code> and <code>\right</code> in display-mode.

Test 14

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

Module 3.15[MathTest1]
 $((a^b_c))$ and $((a^b_c))$.

Test 15

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

\symdecl[ args=a]{ plus }
\symdecl[ args=a]{ mult }
\notation[prec=50]{ plus }{#1}{#1 \comp+ #2}
\notation[prec=100]{ mult }{#1}{#1 \comp\cdot #2}
$\plus{a,\mult{b,c}}$ and $\mult{a,\plus{\frac{ab}{\frac{ac}}{}}}$
$\displaystyle \plus{a,\mult{b,c}}$ and
\withbrackets[]{$\displaystyle
\mult{a,\plus{\frac{ab}{\frac{ac}}{}}}$}
\end{module}
```

<div> <div> Module 3.16[MathTest2] </div> <div> $((a [b;c,d,e,f]^g)) \text{ and } ((a [b,c]^g)) \text{ and } ((a [b]^c))$ $(a+(b\cdot c)) \text{ and } (a\cdot \frac{a}{b} + \frac{a}{c})$ $(a+(b\cdot c)) \text{ and } (a\cdot \frac{a}{b} + \frac{a}{c})$ $(a+(b\cdot c)) \text{ and } [a\cdot \frac{a}{b} + \frac{a}{c}]$ </div> </div>
--

<u><code>\stex_term_custom:nn</code></u>	<code>\stex_term_custom:nn{<URI>}{<args>}</code>
	Implements custom one-time notation. Invoked by <code>\stex_invoke_symbol:n</code> in text mode, or if followed by <code>*</code> in math mode, or whenever followed by <code>!</code> .

<div> <div> Test 16 </div> <div> <pre> \begin{module}{TextTest} \importmodule{Foo} \bar[some]a[and some]b[and also some]c[here]. \$\bar*[\text{some }]a[\text{ and some }]b[\text{ and also some }]c[\text{ here}]\$. \$\bar![\mathtt{bar}]\$ \bar*{a}*{b}[or just some]c \bar![bar] \bar[or first]*[2]{b}[, then]*[3]{c}[, and finally]a \end{module} </pre> </div> </div>	<div> <div> Module 3.17[TextTest] </div> <div> some a and some b and also some c here. some <i>a</i> and some <i>b</i> and also some <i>c</i> here. or just some c bar or first b, then c, and finally a </div> </div>
--	--

<u><code>\stex_highlight_term:nn</code></u>	<code>\stex_highlight_term:nn{<URI>}{<args>}</code>
	Establishes a context for <code>\comp</code> . Stores the URI in a variable so that <code>\comp</code> knows which symbol governs the current notation.

<u><code>\comp</code></u>	<code>\comp{<args>}</code>
<u><code>\@comp</code></u>	Marks <code><args></code> as a notation component of the current symbol for highlighting, linking, etc.
<u><code>\@defemph</code></u>	

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

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

<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{<symboldoc>}{<symbols>} <text> \end{<symboldoc>}</pre> <p>Declares <i><text></i> to be a (natural language, encyclopaedic) description of $\{<symbols>\}$ (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 ! {\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,ltxcmds}
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^AT_EX_{ML} and S^CA_LT_EX

```
64 \RequirePackage{scalatex}
```

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

```
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`

`\stex_annotate_invisible:n`

`\stex_annotate_invisible:nnn`

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

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

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

```

116     } {
117         \tl_use:N \l__stex_annotate_arg_tl
118     }
119 }
120 \NewDocumentEnvironment{stex_annotate_env} { m m } {
121     \par
122     \scalatex_annotate_HTML_begin:n {
123         property="stex:#1" ~
124         resource="#2"
125     }
126 }{
127     \scalatex_annotate_HTML_end:
128 }
129 }{
130     \latexml_if:TF {
131         \cs_new_protected:Nn \stex_annotate:nnn {
132             \__stex_annotate_checkempty:n { #3 }
133             \mode_if_math:TF {
134                 \cs:w latexml@annotate@math\cs_end:{#1}{#2}{
135                     \tl_use:N \l__stex_annotate_arg_tl
136                 }
137             }{
138                 \cs:w latexml@annotate@text\cs_end:{#1}{#2}{
139                     \tl_use:N \l__stex_annotate_arg_tl
140                 }
141             }
142         }
143         \cs_new_protected:Nn \stex_annotate_invisible:n {
144             \__stex_annotate_checkempty:n { #1 }
145             \mode_if_math:TF {
146                 \cs:w latexml@invisible@math\cs_end:{
147                     \tl_use:N \l__stex_annotate_arg_tl
148                 }
149             } {
150                 \cs:w latexml@invisible@text\cs_end:{
151                     \tl_use:N \l__stex_annotate_arg_tl
152                 }
153             }
154         }
155         \cs_new_protected:Nn \stex_annotate_invisible:nnn {
156             \__stex_annotate_checkempty:n { #3 }
157             \cs:w latexml@annotate@invisible\cs_end:{#1}{#2}{
158                 \tl_use:N \l__stex_annotate_arg_tl
159             }
160         }
161     }
162     \NewDocumentEnvironment{stex_annotate_env} { m m } {
163         \par\begin{latexml@annotateenv}{#1}{#2}
164     }{
165         \end{latexml@annotateenv}
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^AT_EX3-sequences (of the individual path segments, i.e. separated by a /-character) unix-style; i.e. a path is absolute if the sequence starts with an empty entry.

`\stex_path_from_string:Nn`
`\stex_path_from_string:NV`
`\stex_path_from_string:cn`
`\stex_path_from_string:cV`

```
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`
`\c__stex_path_up_str`

. and .., respectively.

```
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_if_empty:NTF \l_tmpa_str {
488     \prop_if_empty:NF \l_stex_current_repository_prop {
489       \prop_get:NnN \l_stex_current_repository_prop { id } \l_tmpa_str
490     }
491   } {
492     \stex_require_repository:n \l_tmpa_str
493   }
494   \str_set:Nx \l_tmpa_str { \c_stex_mathhub_str / #1 / source / #2 }
495   \ifinputref

```

```

496     \input{ \l_tmpa_str }
497 \else
498     \inputreftrue
499     \input{ \l_tmpa_str }
500     \inputreffalse
501 \fi
502 }
503 \NewDocumentCommand \inputref { 0{ } m }{
504     \inputref:nn{ #1 }{ #2 }
505 }

```

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

`\mhpath`

```

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

```

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

`\libinput`

```

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

```

```

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

```

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

4.5 Module System

```

556 <@@=stex_module>

```

`\l_stex_current_module_prop`

```

557 \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`

```

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

```

(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`

```

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

```

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

`\stex_add_to_current_module:n`

`\STEXexport`

```

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

```

(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`

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

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

`\stex_add_import_to_current_module:n`

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

(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`

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


(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:`

```

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

```

(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

```

633 \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`

```

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

```

```

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

```

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

module module arguments:

```

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

```

```

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

```

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

```

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

```

```

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

```

```

813   }{
814     \msg_set:nnn{stex}{error/modulemissing}{
815       No~file~for~signature~module~\l_tmpa_str~found
816     }
817     \msg_error:nn{stex}{error/modulemissing}
818   }
819   \stex_activate_module:n {
820     \l_stex_module_ns_str ? \l_stex_module_name_str
821   }
822   \prop_set_eq:Nc \l_stex_current_module_prop {
823     c_stex_module_
824     \l_stex_module_ns_str ?
825     \l_stex_module_name_str
826     _prop
827   }
828 }
829
830 % metatheory
831 \str_if_empty:NT \l_stex_module_meta_str {
832   \str_set:Nx \l_stex_module_meta_str {
833     \c_stex_metatheory_ns_str ? Metatheory
834   }
835 }
836
837
838 \stex_debug:n{
839   New~module:\\
840   Namespace:~\l_stex_module_ns_str\\
841   Name:~\l_stex_module_name_str\\
842   Language:~\l_stex_module_lang_str\\
843   Signature:~\l_stex_module_sig_str\\
844   Metatheory:~\l_stex_module_meta_str\\
845   File:~\stex_path_to_string:N \g_stex_currentfile_seq
846 }
847
848 \seq_put_right:Nx \l_stex_all_modules_seq {
849   \l_stex_module_ns_str ? \l_stex_module_name_str
850 }
851
852 \seq_gput_right:Nx \g_stex_modules_in_file_seq
853   { \l_stex_module_ns_str ? \l_stex_module_name_str }
854
855 \stex_if_smsmode:TF {
856   \stex_smsmode_set_codes:
857 } {
858   \begin{stex_annotate_env} {theory} {
859     \l_stex_module_ns_str ? \l_stex_module_name_str
860   }
861
862   \stex_annotate_invisible:nnn{header}{} {
863     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
864     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
865     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
866       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}

```

```

867     }
868   }
869 }
870
871 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
872   \exp_args:Nx \STEXexport{
873     \stex_activate_module:n {\l_stex_module_meta_str}
874   }
875 }
876 % TODO: Inherit metatheory for nested modules?
877 }
878 \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}`

```

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

```

(End definition for `_stex_module_end_module:.`)

@module The core environment, with no header

```

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

```

`\stex_modules_heading:` Code for document headers

```

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

```

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

Finally:

```

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

```

4.5.2 SMS Mode

959 $\langle @@=\text{stex_smsmode} \rangle$

$\backslash\text{g_stex_smsmode_allowedmacros_tl}$
 $\backslash\text{g_stex_smsmode_allowedmacros_escape_tl}$
 $\backslash\text{g_stex_smsmode_allowedenvs_seq}$

```

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

(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}

```

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

(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.

```

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

(End definition for $\backslash_stex_smsmode_if_catcodes$: \underline{TF} .)

`\stex_smsmode_set_codes:`

```

996 \cs_new_protected:Nn \stex_smsmode_set_codes: {
997   \stex_if_smsmode:T {
998     \__stex_smsmode_if_catcodes:F {
999       \bool_gset_true:N \g__stex_smsmode_catcode_bool
1000       \exp_after:wN \char_gset_active_eq:NN
1001       \c_backslash_str \__stex_smsmode_cs:
1002       \tex_global:D \char_set_catcode_active:N \
1003       \tex_global:D \char_set_catcode_other: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     }
1009   }
1010 } \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.

```

1011 \cs_new_protected:Nn \__stex_smsmode_unset_codes: {
1012   \__stex_smsmode_if_catcodes:T {
1013     \bool_gset_false:N \g__stex_smsmode_catcode_bool
1014     \exp_after:wN \tex_global:D \exp_after:wN
1015     \char_set_catcode_escape:N \c_backslash_str
1016     \tex_global:D \char_set_catcode_math_toggle:N $
1017     \tex_global:D \char_set_catcode_math_superscript:N ^
1018     \tex_global:D \char_set_catcode_math_subscript:N _
1019     \tex_global:D \char_set_catcode_alignment:N &
1020     \tex_global:D \char_set_catcode_parameter:N ##
1021   }
1022 } \iffalse $ \fi % to make syntax highlighting work again

```

(End definition for `__stex_smsmode_unset_codes:`.)

`\stex_in_smsmode:nn`

```

1023 \cs_new_protected:Nn \stex_in_smsmode:nn {
1024   \vbox_set:Nn \l_tmpa_box {
1025     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1026     \bool_gset_true:N \g__stex_smsmode_bool
1027     \stex_smsmode_set_codes:
1028     #2
1029     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1030     \stex_if_smsmode:F {
1031       \__stex_smsmode_unset_codes:
1032     }
1033   }
1034   \box_clear:N \l_tmpa_box
1035 }

```

(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:

```

1036 \cs_new_protected:Nn \__stex_smsmode_cs: {
1037   \str_clear:N \l_tmpa_str
1038   \peek_analysis_map_inline:n {
1039     % #1: token (one expansion)
1040     % #2: charcode
1041     % #3 catcode
1042     \token_if_eq_charcode:NNTF ##3 B {
1043       % token is a letter
1044       \exp_args:NNNo \str_put_right:Nn \l_tmpa_str { ##1 }
1045     } {
1046       \str_if_empty:NNTF \l_tmpa_str {
1047         % we don't allow (or need) single non-letter CSs
1048         % for now
1049         \peek_analysis_map_break:
1050       } {
1051         \str_if_eq:ontf \l_tmpa_str { begin } {
1052           \peek_analysis_map_break:n {
1053             \exp_after:wN \__stex_smsmode_checkbegin:n ##1
1054           }
1055         } {
1056           \str_if_eq:ontf \l_tmpa_str { end } {
1057             \peek_analysis_map_break:n {
1058               \exp_after:wN \__stex_smsmode_checkend:n ##1
1059             }
1060           } {
1061             \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
1062             \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1063               \g_stex_smsmode_allowedmacros_tl
1064               { \use:c{\l_tmpa_str} } {
1065               \stex_debug:n{Executing~1:~\l_tmpa_str}
1066               \peek_analysis_map_break:n {
1067                 \exp_after:wN \l_tmpa_tl ##1
1068               }
1069             } {
1070               \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1071               \g_stex_smsmode_allowedmacros_escape_tl
1072               { \use:c{\l_tmpa_str} } {
1073               \stex_debug:n{Executing~2:~\l_tmpa_str}
1074               % TODO \__stex_smsmode_rescan_cs:
1075               \exp_after:wN \exp_after:wN \exp_after:wN
1076               \token_if_eq_charcode:NNTF \exp_after:wN \c_backslash_str ##1 {
1077               \peek_analysis_map_break:n {
1078                 \__stex_smsmode_unset_codes:
1079                 \__stex_smsmode_rescan_cs:
1080               }
1081             } {
1082               \peek_analysis_map_break:n {
1083                 \__stex_smsmode_unset_codes:
1084                 \exp_after:wN \l_tmpa_tl ##1
1085               }
1086             }
1087           } {
1088             \peek_analysis_map_break:n { ##1 }
1089           }

```

```

1090     }
1091   }
1092 }
1093 }
1094 }
1095 }
1096 }

```

(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:

```

1097 \cs_new_protected:Nn \_stex_smsmode_rescan_cs: {
1098   \str_clear:N \l_tmpb_str
1099   \peek_analysis_map_inline:n {
1100     \token_if_eq_charcode:NNTF ##3 B {
1101       % token is a letter
1102       \exp_args:NNo \str_put_right:Nn \l_tmpb_str { ##1 }
1103     } {
1104       \peek_analysis_map_break:n {
1105         \exp_after:wN \use:c \exp_after:wN {
1106           \exp_after:wN \l_tmpa_str\exp_after:wN
1107         } \use:c { \l_tmpb_str \exp_after:wN } ##1
1108       }
1109     }
1110   }
1111 }

```

(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.

```

1112 \cs_new_protected:Nn \_stex_smsmode_checkbegin:n {
1113   \str_set:Nn \l_tmpa_str { #1 }
1114   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1115     \_stex_smsmode_unset_codes:
1116     \begin{#1}
1117   }
1118 }

```

(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.

```

1119 \cs_new_protected:Nn \_stex_smsmode_checkend:n {
1120   \str_set:Nn \l_tmpa_str { #1 }
1121   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1122     \end{#1}
1123   }
1124 }

```

(End definition for `_stex_smsmode_checkend:n`.)

4.5.3 Inheritance

1125 <@@=stex_importmodule>

\stex_import_module_uri:nn

```

1126 \cs_new_protected:Nn \stex_import_module_uri:nn {
1127   \str_set:Nx \l__stex_importmodule_archive_str { #1 }
1128   \str_set:Nx \l__stex_importmodule_path_str { #2 }
1129   \str_if_empty:NT \l__stex_importmodule_archive_str {
1130     \prop_if_empty:NF \l_stex_current_repository_prop {
1131       \prop_get:NnN \l_stex_current_repository_prop { id } \l__stex_importmodule_archive_str
1132     }
1133   }
1134
1135   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l__stex_importmodule_path_str }
1136   \seq_pop_right:NN \l_tmpb_seq \l__stex_importmodule_name_str
1137   \str_set:Nx \l__stex_importmodule_path_str { \seq_use:Nn \l_tmpb_seq ? }
1138
1139   \str_if_empty:NTF \l__stex_importmodule_archive_str {
1140     \stex_modules_current_namespace:
1141     \str_if_empty:NF \l__stex_importmodule_path_str {
1142       \str_set:Nx \l_stex_module_ns_str {
1143         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1144       }
1145     }
1146   }{
1147     \stex_require_repository:n \l__stex_importmodule_archive_str
1148     \prop_get:cnN { c_stex_mathhub\l__stex_importmodule_archive_str _manifest_prop } { ns }
1149     \l_stex_module_ns_str
1150     \str_if_empty:NF \l__stex_importmodule_path_str {
1151       \str_set:Nx \l_stex_module_ns_str {
1152         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1153       }
1154     }
1155   }
1156 }
```

(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.

```

1157 \str_new:N \l__stex_importmodule_name_str
1158 \str_new:N \l__stex_importmodule_archive_str
1159 \str_new:N \l__stex_importmodule_path_str
1160 \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>}

```

1161 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1162   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1163     % \stex_debug:n{Arguments: #1, #2, #3, #4}
1164
1165     % archive
1166     \str_set:Nx \l_tmpa_str { #2 }
1167     \str_if_empty:NTF \l_tmpa_str {
```

```

1168     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1169 } {
1170   \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1171   \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1172   \seq_put_right:Nn \l_tmpa_seq { source }
1173 }
1174
1175 % path
1176 \str_set:Nx \l_tmpb_str { #3 }
1177 \str_if_empty:NTF \l_tmpb_str {
1178   \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1179 }
1180 \ltx@ifpackageloaded{babel} {
1181   \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1182     { \language } \l_tmpb_str {
1183     \msg_set:nnn{stex}{error/unknownlanguage}{
1184       Unknown-language-~\language
1185     }
1186     \msg_error:nn{stex}{error/unknownlanguage}
1187   }
1188 } {
1189   \str_clear:N \l_tmpb_str
1190 }
1191
1192 \stex_debug:n{Checking-~\l_tmpa_str.\l_tmpb_str.tex}
1193 \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1194   \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1195 }{
1196   \stex_debug:n{Checking-~\l_tmpa_str.tex}
1197   \IfFileExists{ \l_tmpa_str.tex }{
1198     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1199   }{
1200     % try english as default
1201     \stex_debug:n{Checking-~\l_tmpa_str.en.tex}
1202     \IfFileExists{ \l_tmpa_str.en.tex }{
1203       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1204     }{
1205       \msg_set:nnn{stex}{error/modulemissing}{
1206         No~file~for~module~#1?#4~found
1207       }
1208       \msg_error:nn{stex}{error/modulemissing}
1209     }
1210   }
1211 }
1212
1213 } {
1214   \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1215   \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1216
1217   \ltx@ifpackageloaded{babel} {
1218     \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1219       { \language } \l_tmpb_str {
1220       \msg_set:nnn{stex}{error/unknownlanguage}{
1221         Unknown-language-~\language

```

```

1222     }
1223     \msg_error:nn{stex}{error/unknownlanguage}
1224 }
1225 } {
1226     \str_clear:N \l_tmpb_str
1227 }
1228
1229 \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1230
1231 \stex_debug:n{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1232 \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1233     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1234 }{
1235     \stex_debug:n{Checking~\l_tmpa_str/#4.tex}
1236     \IfFileExists{ \l_tmpa_str/#4.tex }{
1237         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1238     }{
1239         % try english as default
1240         \stex_debug:n{Checking~\l_tmpa_str/#4.en.tex}
1241         \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1242             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1243         }{
1244             \stex_debug:n{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1245             \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1246                 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1247             }{
1248                 \stex_debug:n{Checking~\l_tmpa_str.tex}
1249                 \IfFileExists{ \l_tmpa_str.tex }{
1250                     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1251                 }{
1252                     % try english as default
1253                     \stex_debug:n{Checking~\l_tmpa_str.en.tex}
1254                     \IfFileExists{ \l_tmpa_str.en.tex }{
1255                         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1256                     }{
1257                         \msg_set:nnn{stex}{error/modulemissing}{
1258                             No~file~for~module~#1?#4~found
1259                         }
1260                         \msg_error:nn{stex}{error/modulemissing}
1261                     }
1262                 }
1263             }
1264         }
1265     }
1266 }
1267 }
1268
1269 \seq_set_eq:NN \l_tmpa_seq \g_stex_modules_in_file_seq
1270 \seq_clear:N \g_stex_modules_in_file_seq
1271 % \exp_args:Nnx \use:nn {
1272     \exp_args:No \stex_in_smsmode:nn { \g__stex_importmodule_file_str } {
1273         \seq_clear:N \l_stex_all_modules_seq
1274         \prop_clear:N \l_stex_current_module_prop
1275         \str_set:Nx \l_tmpb_str { #2 }

```

```

1276     \str_if_empty:NF \l_tmpb_str {
1277       \stex_set_current_repository:n { #2 }
1278     }
1279     \stex_debug:n{Loading~\g__stex_importmodule_file_str}
1280     \input { \g__stex_importmodule_file_str }
1281   }
1282 %   }{
1283
1284 %   }
1285   \prop_gput:Noo \g_stex_module_files_prop
1286   \g__stex_importmodule_file_str \g_stex_modules_in_file_seq
1287   \seq_set_eq:NN \g_stex_modules_in_file_seq \l_tmpa_seq
1288
1289   \stex_if_module_exists:nF { #1 ? #4 } {
1290     \msg_set:nnn{stex}{error/modulemissing}{
1291       Module~#1?#4~not~found~in~file~\g__stex_importmodule_file_str
1292     }
1293     \msg_error:nn{stex}{error/modulemissing}
1294   }
1295 }
1296 \stex_activate_module:n { #1 ? #4 }
1297 }

```

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

`\stex_activate_module:n`

```

1298 \cs_new_protected:Nn \stex_activate_module:n {
1299   \stex_debug:n{Activating~module~#1}
1300   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1301     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1302     \prop_item:cn { c_stex_module_#1_prop } { content }
1303   }
1304 }

```

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

`\importmodule`

```

1305 \NewDocumentCommand \importmodule { 0{} m } {
1306   \stex_import_module_uri:nn { #1 } { #2 }
1307   \stex_debug:n{Importing~module:~
1308     \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1309   }
1310   \stex_if_smsmode:F {
1311     \stex_import_require_module:nnnn
1312     { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1313     { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1314     \stex_annotate_invisible:nnn
1315     {import} { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } {}
1316   }
1317   \exp_args:Nx \stex_add_to_current_module:n {
1318     \stex_import_require_module:nnnn
1319     { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1320     { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1321   }
1322   \exp_args:Nx \stex_add_import_to_current_module:n {

```

```

1323 \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1324 }
1325 \stex_smsmode_set_codes:
1326 }

```

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

\usemodule

```

1327 \NewDocumentCommand \usemodule { 0{} m } {
1328 \stex_if_smsmode:F {
1329 \stex_import_module_uri:nn { #1 } { #2 }
1330 \stex_import_require_module:nnnn
1331 { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1332 { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1333 \stex_annotate_invisible:nnn
1334 {usemodule} {\l_stex_module_ns_str ? \l__stex_importmodule_name_str} {}
1335 }
1336 \stex_smsmode_set_codes:
1337 }

```

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

\g_stex_modules_in_file_seq \g_stex_module_files_prop

```

1338 \seq_new:N \g_stex_modules_in_file_seq
1339 \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

```

1340 <@@=stex_symdecl>

```

\l_stex_all_symbols_seq Stores all available symbols

```

1341 \seq_new:N \l_stex_all_symbols_seq

```

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

\STEXsymbol

```

1342 \NewDocumentCommand \STEXsymbol { m } {
1343 \stex_get_symbol:n { #1 }
1344 \exp_args:No
1345 \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1346 }

```

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

symdecl arguments:

```

1347 \keys_define:nn { stex / symdecl } {
1348 name .tl_set:x:N = \l_stex_symdecl_name_str ,
1349 local .bool_set:N = \l_stex_symdecl_local_bool ,
1350 args .tl_set:x:N = \l_stex_symdecl_args_str ,
1351 type .tl_set:N = \l_stex_symdecl_type_tl ,
1352 align .tl_set:N = \l_stex_symdecl_align_str , % TODO(?)
1353 gfc .tl_set:N = \l_stex_symdecl_gfc_str , % TODO(?)
1354 specializes .tl_set:N = \l_stex_symdecl_specializes_str , % TODO(?)

```



```

1355   def          .tl_set:N      = \l_stex_symdecl_definiens_tl
1356 }
1357
1358 \bool_new:N \l_stex_symdecl_make_macro_bool
1359
1360 \cs_new_protected:Nn \__stex_symdecl_args:n {
1361   \str_clear:N \l_stex_symdecl_name_str
1362   \str_clear:N \l_stex_symdecl_args_str
1363   \bool_set_false:N \l_stex_symdecl_local_bool
1364   \tl_clear:N \l_stex_symdecl_type_tl
1365   \tl_clear:N \l_stex_symdecl_definiens_tl
1366
1367   \keys_set:nn { stex /symdecl } { #1 }
1368
1369   \exp_args:NNo \str_set:Nn \l_stex_symdecl_name_str
1370     \l_stex_symdecl_name_str
1371   \exp_args:NNo \str_set:Nn \l_stex_symdecl_args_str
1372     \l_stex_symdecl_args_str
1373 }

```

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

```

1374
1375 \NewDocumentCommand \symdecl { s O{} m } {
1376   \__stex_symdecl_args:n { #2 }
1377   \IfBooleanTF #1 {
1378     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1379   } {
1380     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1381   }
1382   \stex_symdecl_do:n { #3 }
1383   \stex_smsmode_set_codes:
1384 }

```

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

\stex_symdecl_do:n

```

1385 \cs_new_protected:Nn \stex_symdecl_do:n {
1386   \stex_if_in_module:F {
1387     % TODO throw error? some default namespace?
1388   }
1389
1390   \str_if_empty:NT \l_stex_symdecl_name_str {
1391     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1392   }
1393
1394   \prop_if_exist:cT { g_stex_symdecl_
1395     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1396     \prop_item:Nn \l_stex_current_module_prop {name} ?
1397     \l_stex_symdecl_name_str
1398     _prop
1399   }{
1400     % TODO throw error (beware of circular dependencies)
1401   }

```

```

1402 \prop_clear:N \l_tmpa_prop
1403 \prop_put:Nnx \l_tmpa_prop { module } {
1404   \prop_item:Nn \l_stex_current_module_prop {ns} ?
1405   \prop_item:Nn \l_stex_current_module_prop {name}
1406 }
1407 \seq_clear:N \l_tmpa_seq
1408 \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
1409 \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1410 \prop_put:Nno \l_tmpa_prop { local } \l_stex_symdecl_local_bool
1411 \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1412
1413 \exp_args:No \stex_add_constant_to_current_module:n {
1414   \l_stex_symdecl_name_str
1415 }
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     {\tl_to_str:n B} {
1432       \bool_set_false:N \l_tmpa_bool
1433       \int_incr:N \l_tmpb_int
1434     }
1435   }{
1436     \msg_set:nnn{stex}{error/wrongargs}{
1437       args~value~in~symbol~declaration~for~
1438       \prop_item:Nn \l_stex_current_module_prop {ns} ?
1439       \prop_item:Nn \l_stex_current_module_prop {name} ?
1440       \l_stex_symdecl_name_str ~
1441       needs~to~be~
1442       i,~a,~b~or~B,~but~##1~given
1443     }
1444     \msg_error:nn{stex}{error/wrongargs}
1445   }
1446 }
1447 \bool_if:NTF \l_tmpa_bool {
1448   % possibly numeric
1449   \str_if_empty:NTF \l_stex_symdecl_args_str {
1450     \prop_put:Nnn \l_tmpa_prop { args } {}
1451     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1452   }{
1453     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1454     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1455     \str_clear:N \l_tmpa_str

```

```

1456     \int_step_inline:nn \l_tmpa_int {
1457       \str_put_right:Nn \l_tmpa_str i
1458     }
1459     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1460   }
1461 } {
1462   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1463   \prop_put:Nnx \l_tmpa_prop { arity }
1464     { \str_count:N \l_stex_symdecl_args_str }
1465 }
1466 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
1467
1468
1469 % semantic macro
1470
1471 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1472   \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1473     \prop_item:Nn \l_tmpa_prop { module } ?
1474     \prop_item:Nn \l_tmpa_prop { name }
1475   } }
1476
1477   \bool_if:NF \l_stex_symdecl_local_bool {
1478     \exp_args:Nx \stex_add_to_current_module:n {
1479       \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1480         \prop_item:Nn \l_tmpa_prop { module } ?
1481         \prop_item:Nn \l_tmpa_prop { name }
1482       } }
1483     }
1484   }
1485 }
1486
1487 % add to all symbols
1488
1489 \bool_if:NF \l_stex_symdecl_local_bool {
1490   \exp_args:Nx \stex_add_to_current_module:n {
1491     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1492       \prop_item:Nn \l_tmpa_prop { module } ?
1493       \prop_item:Nn \l_tmpa_prop { name }
1494     }
1495   }
1496 }
1497
1498 \stex_debug:n{New~symbol:~
1499   \prop_item:Nn \l_tmpa_prop { module } ?
1500   \prop_item:Nn \l_tmpa_prop { name }^^J
1501   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
1502   Args:~\prop_item:Nn \l_tmpa_prop { args }
1503 }
1504
1505 % circular dependencies require this:
1506
1507 \prop_if_exist:cF {
1508   g_stex_symdecl_
1509   \prop_item:Nn \l_tmpa_prop { module } ?

```

```

1510     \prop_item:Nn \l_tmpa_prop { name }
1511     _prop
1512   } {
1513     \prop_gset_eq:cN {
1514       g_stex_symdecl_
1515       \prop_item:Nn \l_tmpa_prop { module } ?
1516       \prop_item:Nn \l_tmpa_prop { name }
1517       _prop
1518     } \l_tmpa_prop
1519   }
1520
1521   \stex_if_smsmode:TF {
1522     \bool_if:NF \l_stex_symdecl_local_bool {
1523       \exp_args:Nx \stex_addtosms:n {
1524         \prop_gset_from_keyval:cn {
1525           g_stex_symdecl_
1526           \prop_item:Nn \l_tmpa_prop { module } ?
1527           \prop_item:Nn \l_tmpa_prop { name }
1528           _prop
1529         } {
1530           name      = \prop_item:Nn \l_tmpa_prop { name }      ,
1531           module    = \prop_item:Nn \l_tmpa_prop { module }    ,
1532           notations = \prop_item:Nn \l_tmpa_prop { notations } ,
1533           local     = \prop_item:Nn \l_tmpa_prop { local }     ,
1534           type      = \prop_item:Nn \l_tmpa_prop { type }      ,
1535           args      = \prop_item:Nn \l_tmpa_prop { args }      ,
1536           arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
1537           assocs    = \prop_item:Nn \l_tmpa_prop { assocs }
1538         }
1539         \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1540           \prop_item:Nn \l_tmpa_prop { module } ?
1541           \prop_item:Nn \l_tmpa_prop { name }
1542         }
1543       }
1544     }
1545   }{
1546     \exp_args:NNx \seq_put_right:Nn \l_stex_all_symbols_seq {
1547       \prop_item:Nn \l_tmpa_prop { module } ?
1548       \prop_item:Nn \l_tmpa_prop { name }
1549     }
1550     \stex_annotate_invisible:nnn {symdecl} {
1551       \prop_item:Nn \l_tmpa_prop { module } ?
1552       \prop_item:Nn \l_tmpa_prop { name }
1553     } {
1554       \stex_annotate_invisible:nnn{type}{}{\l_stex_symdecl_type_tl$}
1555       \stex_annotate_invisible:nnn{args}{}{
1556         \prop_item:Nn \l_tmpa_prop { args }
1557       }
1558       \stex_annotate_invisible:nnn{macroname}{}{#1}
1559       \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
1560         \stex_annotate_invisible:nnn{definiens}{}{
1561           {\l_stex_symdecl_definiens_tl$}
1562         }
1563       }

```

```

1564 }
1565 }

```

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

`\stex_get_symbol:n`

```

1566 \str_new:N \l_stex_get_symbol_uri_str
1567
1568 \cs_new_protected:Nn \stex_get_symbol:n {
1569   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
1570     \__stex_symdecl_get_symbol_from_cs:n { #1 }
1571   }{
1572     % argument is a string
1573     % is it a command name?
1574     \cs_if_exist:cTF { #1 }{
1575       \cs_set_eq:Nc \l_tmpa_tl { #1 }
1576       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
1577       \str_if_empty:NNTF \l_tmpa_str {
1578         \exp_args:Nx \cs_if_eq:NNTF {
1579           \tl_head:N \l_tmpa_tl
1580         } \stex_invoke_symbol:n {
1581           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
1582         }{
1583           \__stex_symdecl_get_symbol_from_string:n { #1 }
1584         }
1585       } {
1586         \__stex_symdecl_get_symbol_from_string:n { #1 }
1587       }
1588     }{
1589       % argument is not a command name
1590       \__stex_symdecl_get_symbol_from_string:n { #1 }
1591       % \l_stex_all_symbols_seq
1592     }
1593   }
1594 }
1595
1596 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
1597   \bool_set_false:N \l_tmpa_bool
1598   \stex_if_in_module:T {
1599     \prop_get:NnN \l_stex_current_module_prop
1600     { constants } \l_tmpa_seq
1601     \exp_args:NNo \seq_if_in:NnTF \l_tmpa_seq { \l_tmpa_str } {
1602       \bool_set_true:N \l_tmpa_bool
1603       \str_set:Nx \l_stex_get_symbol_uri_str {
1604         \prop_item:Nn \l_stex_current_module_prop { ns } ?
1605         \prop_item:Nn \l_stex_current_module_prop { name } ? #1
1606       }
1607     }
1608   }
1609   \bool_if:NF \l_tmpa_bool {
1610     \tl_set:Nn \l_tmpa_tl {
1611       \msg_set:nnn{stex}{error/unknownsymbol}{
1612         No~symbol~#1~found!
1613       }
1614     }
1615   }

```

```

1614     \msg_error:nn{stex}{error/unknownsymbol}
1615   }
1616   \str_set:Nn \l_tmpa_str { #1 }
1617   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1618   \seq_map_inline:Nn \l_stex_all_symbols_seq {
1619     \str_set:Nn \l_tmpb_str { ##1 }
1620     \str_if_eq:eeT { \l_tmpa_str } {
1621       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1622     } {
1623       \seq_map_break:n {
1624         \tl_set:Nn \l_tmpa_tl {
1625           \str_set:Nn \l_stex_get_symbol_uri_str {
1626             ##1
1627           }
1628         }
1629       }
1630     }
1631   }
1632   \l_tmpa_tl
1633 }
1634 }
1635
1636 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
1637   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
1638   { \tl_tail:N \l_tmpa_tl }
1639   \tl_if_single:NTF \l_tmpa_tl {
1640     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
1641       \exp_after:wN \str_set:Nn \exp_after:wN
1642       \l_stex_get_symbol_uri_str \l_tmpa_tl
1643     }{
1644       % TODO
1645       % tail is not a single group
1646     }
1647   }{
1648     % TODO
1649     % tail is not a single group
1650   }
1651 }

```

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

4.7 Notations

```

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

```

```

1662 \cs_new_protected:Nn \__stex_notation_args:n {
1663   \str_clear:N \l__stex_notation_lang_str
1664   \str_clear:N \l__stex_notation_variant_str
1665   \str_clear:N \l__stex_notation_prec_str
1666   \tl_clear:N \l__stex_notation_op_tl
1667
1668   \keys_set:nn { stex / notation } { #1 }
1669
1670   \str_set:Nx \l__stex_notation_lang_str \l__stex_notation_lang_str
1671   \str_set:Nx \l__stex_notation_variant_str \l__stex_notation_variant_str
1672   \str_set:Nx \l__stex_notation_prec_str \l__stex_notation_prec_str
1673 }

```

\notation

```

1674 \NewDocumentCommand \notation { 0{} m } {
1675   \__stex_notation_args:n { #1 }
1676   \tl_clear:N \l_stex_symdecl_definiens_tl
1677   \stex_get_symbol:n { #2 }
1678   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
1679 }

```

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

\stex_notation_do:nn

```

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

```

```

1711     }
1712   }{
1713     \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
1714     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
1715       \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
1716       \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
1717         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
1718         \l_tmpa_seq {\tl_to_str:n{x} } { \l_tmpa_str }
1719         \seq_map_inline:Nn \l_tmpa_seq {
1720           \seq_put_right:Nn \l_tmpb_seq { ##1 }
1721         }
1722       }
1723       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1724     }{
1725       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1726       \int_compare:nNnTF \l_tmpa_str = 0 {
1727         \exp_args:NNnx
1728         \prop_put:Nno \l_tmpb_prop { opprec }
1729         { \infprec }
1730       }{
1731         \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
1732       }
1733     }
1734   }
1735 }
1736
1737 \seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
1738 \int_step_inline:nn { \l_tmpa_str } {
1739   \seq_pop_left:NNTF \l_tmpa_seq \l_tmpb_str {
1740     \exp_args:NNx
1741     \seq_put_right:Nn \l_tmpb_seq {
1742       \prop_item:Nn \l_tmpb_prop { opprec }
1743     }
1744   }
1745 }
1746
1747 \prop_put:Nno \l_tmpb_prop { argprec } \l_tmpb_seq
1748 \tl_clear:N \l_tmpa_tl
1749
1750 \int_compare:nNnTF \l_tmpa_str = 0 {
1751   \exp_args:NNe
1752   \cs_set:Npn \l__stex_notation_macrocode_cs {
1753     \stex_term_math_oms:nnnn { #1 }
1754     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1755     { \prop_item:Nn \l_tmpb_prop { opprec } }
1756     { \exp_not:n { #2 } }
1757   }
1758   \__stex_notation_final:
1759 }{
1760   \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
1761   \str_if_in:NnTF \l_tmpb_str b {
1762     \exp_args:Nne \use:nn
1763     {
1764       \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs

```



```

1765 \cs_set:Npn \l_tmpa_str } { {
1766   \stex_term_math_omb:nnnn { #1 }
1767   { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1768   { \prop_item:Nn \l_tmpb_prop { opprec } }
1769   { \exp_not:n { #2 } }
1770 } }
1771 }{
1772 \str_if_in:NnTF \l_tmpb_str B {
1773   \exp_args:Nne \use:nn
1774   {
1775     \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1776     \cs_set:Npn \l_tmpa_str } { {
1777       \stex_term_math_omb:nnnn { #1 }
1778       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1779       { \prop_item:Nn \l_tmpb_prop { opprec } }
1780       { \exp_not:n { #2 } }
1781     } }
1782   }{
1783     \exp_args:Nne \use:nn
1784     {
1785       \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1786       \cs_set:Npn \l_tmpa_str } { {
1787         \stex_term_math_oma:nnnn { #1 }
1788         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1789         { \prop_item:Nn \l_tmpb_prop { opprec } }
1790         { \exp_not:n { #2 } }
1791       } }
1792     }
1793   }
1794
1795   \int_zero:N \l_tmpa_int
1796   \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
1797   \prop_get:NnN \l_tmpb_prop { argprecs } \l_tmpa_seq
1798   \__stex_notation_arguments:
1799 }
1800 }

```

(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

```

1801 \cs_new_protected:Nn \__stex_notation_arguments: {
1802   \int_incr:N \l_tmpa_int
1803   \str_if_empty:NNTF \l_tmpa_str {
1804     \__stex_notation_final:
1805   }{
1806     \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
1807     \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
1808     \str_if_eq:VnTF \l_tmpb_str a {
1809       \__stex_notation_argument_assoc:n
1810     }{
1811       \str_if_eq:VnTF \l_tmpb_str B {
1812         \__stex_notation_argument_assoc:n
1813       }{
1814         \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str

```

```

1815     \tl_put_right:Nx \l_tmpa_tl {
1816       { \stex_term_math_arg:nnn
1817         { \int_use:N \l_tmpa_int }
1818         { \l_tmpb_str }
1819         { ####\int_use:N \l_tmpa_int }
1820       }
1821     }
1822     \__stex_notation_arguments:
1823   }
1824 }
1825 }
1826 }

```

(End definition for __stex_notation_arguments:.)

_stex_notation_argument_assoc:n

```

1827 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
1828   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1829   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
1830   \tl_put_right:Nx \l_tmpa_tl {
1831     { \stex_term_math_assoc_arg:nnnn
1832       { \int_use:N \l_tmpa_int }
1833       { \l_tmpb_str }
1834       \exp_args:No \exp_not:n
1835       {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
1836       { ####\int_use:N \l_tmpa_int }
1837     }
1838   }
1839   \__stex_notation_arguments:
1840 }

```

(End definition for __stex_notation_argument_assoc:n.)

_stex_notation_final: Called after processing all notation arguments

```

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

```

```

1861     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1862     _cs
1863   } {
1864     \stex_term_oms:nnn {
1865       \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
1866       \l__stex_notation_lang_str
1867     }{
1868       \l_tmpa_str
1869     }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
1870   }
1871 }
1872
1873
1874
1875 \stex_debug:n{
1876   Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1877   ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
1878   Operator~precedence:~
1879   \prop_item:Nn \l_tmpb_prop { opprec }^^J
1880   Argument~precedences:~
1881   \seq_use:Nn \l_tmpa_seq {,~}^^J
1882   Notation: \cs_meaning:c {
1883     stex_notation_ \l_tmpa_str \c_hash_str
1884     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1885     _cs
1886   }
1887 }
1888
1889 \prop_gset_eq:cN {
1890   g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
1891   \c_hash_str \l__stex_notation_lang_str _prop
1892 } \l_tmpb_prop
1893
1894 \exp_args:Nx
1895 \stex_add_to_current_module:n {
1896   \prop_get:cnN {
1897     g_stex_symdecl_
1898     \prop_item:Nn \l_tmpb_prop { symbol }
1899     _prop
1900   } { notations } \exp_not:N \l_tmpa_seq
1901   \seq_put_right:Nn \exp_not:N \l_tmpa_seq {
1902     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1903   }
1904   \prop_put:cno {
1905     g_stex_symdecl_
1906     \prop_item:Nn \l_tmpb_prop { symbol }
1907     _prop
1908   } { notations } \exp_not:N \l_tmpa_seq
1909 }
1910
1911 \stex_if_smsmode:TF {
1912   \stex_smsmode_set_codes:
1913   \exp_args:Nx \stex_addtosms:n {
1914     \prop_gset_from_keyval:cn {

```

```

1915     g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
1916     \c_hash_str \l__stex_notation_lang_str _prop
1917   } {
1918     symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
1919     language    = \prop_item:Nn \l_tmpb_prop { language }    ,
1920     variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,
1921     opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
1922     argprec     = \prop_item:Nn \l_tmpb_prop { argprec }     ,
1923   }
1924 }
1925 }{
1926   \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
1927   \seq_put_right:Nx \l_tmpa_seq {
1928     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1929   }
1930   \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
1931   \prop_set_eq:cN {
1932     g_stex_symdecl_ \l_tmpa_str _prop
1933   } \l_tmpa_prop
1934
1935   % HTML annotations
1936   \stex_annotate_invisible:nnn { notation }
1937   { \prop_item:Nn \l_tmpb_prop { symbol } } {
1938     \stex_annotate_invisible:nnn { notationfragment }
1939     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
1940   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
1941   \stex_annotate_invisible:nnn { precedence }
1942     { \prop_item:Nn \l_tmpb_prop { opprec } ;
1943     \seq_use:Nn \l_tmpa_seq { x }
1944   }{}
1945
1946   \int_zero:N \l_tmpa_int
1947   \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
1948   \tl_clear:N \l_tmpa_tl
1949   \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
1950     \int_incr:N \l_tmpa_int
1951     \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
1952     \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
1953     \str_if_eq:VnTF \l_tmpb_str a {
1954       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
1955         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
1956         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
1957       } }
1958     }{
1959       \str_if_eq:VnTF \l_tmpb_str B {
1960         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
1961           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
1962           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
1963         } }
1964       }{
1965         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
1966           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
1967         } }
1968       }

```

```

1969     }
1970   }
1971   \stex_annotate_invisible:nnn { notationcomp }{}{
1972     $ \exp_args:Nno \use:nn { \use:c {
1973       stex_notation_ \prop_item:Nn \l_tmpb_prop { symbol }
1974       \c_hash_str \l__stex_notation_variant_str
1975       \c_hash_str \l__stex_notation_lang_str _cs
1976     } } { \l_tmpa_tl } $
1977   }
1978 }
1979 }
1980 }

```

(End definition for _stex_notation_final:.)

\symdef

```

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

```

```

2019
2020 \NewDocumentCommand \symdef { 0{} m } {
2021   \_stex_notation_symdef_args:n { #1 }
2022   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2023   \stex_symdecl_do:n { #2 }
2024   \exp_args:Nx \stex_notation_do:nn {
2025     \prop_item:Nn \l_tmpa_prop { module } ?
2026     \prop_item:Nn \l_tmpa_prop { name }
2027   }
2028 }

```

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

`\stex_invoke_symbol:n` Invokes a semantic macro

```

2029 %\cs_new_protected:Nn \stex_invoke_symbol:n {
2030 %  \peek_charcode_remove:NTF ! {
2031 %    \stex_term_custom:nn { #1 } { }
2032 %  } {
2033 %    \if_mode_math:
2034 %      \exp_after:wN \_stex_notation_invoke_math:n
2035 %    \else:
2036 %      \exp_after:wN \_stex_notation_invoke_text:n
2037 %    \fi: { #1 }
2038 %  }
2039 %}
2040
2041 \cs_new_protected:Nn \stex_invoke_symbol:n {
2042   \if_mode_math:
2043     \exp_after:wN \_stex_notation_invoke_math:n
2044   \else:
2045     \exp_after:wN \_stex_notation_invoke_text:n
2046   \fi: { #1 }
2047 }

```

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

`_stex_notation_invoke_math:n`

```

2048 \cs_new_protected:Nn \_stex_notation_invoke_math:n {
2049   \peek_charcode_remove:NTF ! {
2050     \peek_charcode:NTF [ {
2051       \_stex_notation_invoke_op:nw { #1 }
2052     }{
2053       \_stex_notation_invoke_op:nw { #1 } []
2054     }
2055   }{
2056     \peek_charcode_remove:NTF * {
2057       \_stex_notation_invoke_text:n { #1 }
2058     }{
2059       \peek_charcode:NTF [ {
2060         \_stex_notation_invoke_math:nw { #1 }
2061       }{
2062         \_stex_notation_invoke_math:nw { #1 } []
2063       }
2064     }

```

```

2065 }
2066 }

```

(End definition for `_stex_notation_invoke_math:n`.)

`_stex_notation_invoke_op:nw`

```

2067 \cs_new_protected:Npn \_stex_notation_invoke_op:nw #1 [#2] {
2068   \_stex_notation_args:n { #2 }
2069   \cs_if_exist:cTF {
2070     stex_op_notation_ #1 \c_hash_str
2071     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str _cs
2072   }{
2073     \csname stex_op_notation_ #1 \c_hash_str
2074       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str _cs
2075     \endcsname
2076   }{
2077     % TODO throw error
2078   }
2079 }

```

(End definition for `_stex_notation_invoke_op:nw`.)

`_stex_notation_invoke_math:nw`

```

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

```

```

2112         \msg_error:nn{stex}{error/wrongnotation}
2113     }
2114     }{
2115         \msg_set:nnn{stex}{error/wrongnotation}{
2116             Symbol~#1~has~no~notation~
2117             \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2118         }
2119         \msg_error:nn{stex}{error/wrongnotation}
2120     }
2121 }
2122 }
2123 }

```

(End definition for `__stex_notation_invoke_math:nw`.)

`_stex_notation_invoke_text:n`

```

2124 \cs_new_protected:Nn \__stex_notation_invoke_text:n {
2125     \peek_charcode_remove:NTF ! {
2126         \stex_term_custom:nn { #1 } { }
2127     }{
2128         \prop_set_eq:Nc \l_tmpa_prop {
2129             g_stex_symdecl_ #1 _prop
2130         }
2131         \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2132         \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2133     }
2134 }

```

(End definition for `__stex_notation_invoke_text:n`.)

4.8 Terms

2135 `<@@=stex_term>`

Precedences:

```

\infprec
\neginfprec
\l__stex_term_downprec
2136 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2137 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2138 \int_new:N \l__stex_term_downprec
2139 \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
2140 \tl_set:Nn \l__stex_term_left_bracket_str (
2141 \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

```

2142 \cs_new_protected:Nn \_stex_term_maybe_brackets:nn {
2143     \int_compare:nNnTF { #1 } > \l__stex_term_downprec {
2144         \bool_if:NTF \l_stex_inarray_bool { #2 }{

```



```

2145     \dobrackets { #2 }
2146   }
2147   }{ #2 }
2148 }

```

(End definition for `_stex_term_maybe_brackets:nn`.)

`\dobrackets`

```

2149 %\RequirePackage{scalerel}
2150 \cs_new_protected:Npn \dobrackets #1 {
2151   %\ThisStyle{\if D\m@switch
2152   %   \exp_args:Nnx \use:nn
2153   %   { \exp_after:wN \left\l__stex_term_left_bracket_str #1 }
2154   %   { \exp_not:N\right\l__stex_term_right_bracket_str }
2155   % \else
2156   %   \exp_args:Nnx \use:nn
2157   %   { \l__stex_term_left_bracket_str #1 }
2158   %   { \l__stex_term_right_bracket_str }
2159   %\fi}
2160 }

```

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

`\withbrackets`

```

2161 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2162   \exp_args:Nnx \use:nn
2163   {
2164     \tl_set:Nx \l__stex_term_left_bracket_str { #1 }
2165     \tl_set:Nx \l__stex_term_right_bracket_str { #2 }
2166     #3
2167   }
2168   {
2169     \tl_set:Nn \exp_not:N \l__stex_term_left_bracket_str
2170     { \l__stex_term_left_bracket_str }
2171     \tl_set:Nn \exp_not:N \l__stex_term_right_bracket_str
2172     { \l__stex_term_right_bracket_str }
2173   }
2174 }

```

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

`\STEXinvisible`

```

2175 \cs_new_protected:Npn \STEXinvisible #1 {
2176   \stex_annotate_invisible:n { #1 }
2177 }

```

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

OMDOC terms:

`_stex_term_math_oms:nnnn`

```

2178 \cs_new_protected:Nn \_stex_term_oms:nnn {
2179   \stex_annotate:nnn{ OMID }{ #2 }{
2180     \stex_highlight_term:nn { #1 } { #3 }
2181   }
2182 }

```

```

2183
2184 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2185   \_stex_term_maybe_brackets:nn { #3 }{
2186     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2187   }
2188 }

```

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

`_stex_term_math_oma:nnnn`

```

2189 \cs_new_protected:Nn \_stex_term_oma:nnn {
2190   \stex_annotate:nnn{ OMA }{ #2 }{
2191     \stex_highlight_term:nn { #1 } { #3 }
2192   }
2193 }
2194
2195 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2196   \_stex_term_maybe_brackets:nn { #3 }{
2197     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2198   }
2199 }

```

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

`_stex_term_math_omb:nnnn`

```

2200 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2201   \stex_annotate:nnn{ OMBIND }{ #2 }{
2202     \stex_highlight_term:nn { #1 } { #3 }
2203   }
2204 }
2205
2206 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2207   \_stex_term_maybe_brackets:nn { #3 }{
2208     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2209   }
2210 }

```

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

`_stex_term_math_arg:nnn`

```

2211 \cs_new_protected:Nn \_stex_term_arg:nn {
2212   \stex_unhighlight_term:n {
2213     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2214   }
2215 }
2216 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
2217   \exp_args:Nnx \use:nn
2218     { \int_set:Nn \l__stex_term_downprec { #2 }
2219       \_stex_term_arg:nn { #1 }{ #3 }
2220     }
2221   { \int_set:Nn \exp_not:N \l__stex_term_downprec { \int_use:N \l__stex_term_downprec } }
2222 }

```

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

`\stex_term_math_assoc_arg:nnnn`

```

2223 \cs_new_protected:Nn \stex_term_math_assoc_arg:nnnn {
2224   \seq_set_split:Nnn \l_tmpa_seq , { #4 }
2225   \int_compare:nNnTF { \seq_count:N \l_tmpa_seq } < 2 {
2226     \tl_set:Nn \l_tmpa_tl { #4 }
2227   }{
2228     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
2229     \seq_reverse:N \l_tmpa_seq
2230     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_tl
2231     \tl_set:No \l_tmpa_tl { \l_tmpb_tl }
2232
2233     \seq_map_inline:Nn \l_tmpa_seq {
2234       \exp_args:NNo \tl_set:No \l_tmpa_tl {
2235         \exp_args:Nno
2236         \l_tmpa_cs { ##1 } \l_tmpa_tl
2237       }
2238     }
2239
2240   }
2241   \exp_args:Nnno
2242   \stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2243 }

```

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

`\stex_term_custom:nn`

```

2244 \cs_new_protected:Nn \stex_term_custom:nn {
2245   \str_set:Nn \l__stex_term_custom_uri { #1 }
2246   \str_set:Nn \l_tmpa_str { #2 }
2247   \tl_clear:N \l_tmpa_tl
2248   \int_zero:N \l_tmpa_int
2249   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2250   \__stex_term_custom_loop:
2251 }

```

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

`__stex_term_custom_loop:`

```

2252 \cs_new_protected:Nn \__stex_term_custom_loop: {
2253   \bool_set_false:N \l_tmpa_bool
2254   \bool_while_do:nn {
2255     \str_if_eq_p:ee X {
2256       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2257     }
2258   }{
2259     \int_incr:N \l_tmpa_int
2260   }
2261
2262   \peek_charcode:NTF [ {
2263     % notation/text component
2264     \__stex_term_custom_component:w
2265   } {
2266     \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2267       % all arguments read => finish

```

```

2268     \__stex_term_custom_final:
2269   } {
2270     % arguments missing
2271     \peek_charcode_remove:NTF * {
2272       % invisible, specific argument position or both
2273       \peek_charcode:NTF [ {
2274         % visible specific argument position
2275         \__stex_term_custom_arg:wn
2276       } {
2277         % invisible
2278         \peek_charcode_remove:NTF * {
2279           % invisible specific argument position
2280           \__stex_term_custom_arg_inv:wn
2281         } {
2282           % invisible next argument
2283           \__stex_term_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2284         }
2285       }
2286     } {
2287       % next normal argument
2288       \__stex_term_custom_arg:wn [ \l_tmpa_int + 1 ]
2289     }
2290   }
2291 }
2292 }

```

(End definition for __stex_term_custom_loop:.)

_stex_term_custom_arg_inv:wn

```

2293 \cs_new_protected:Npn \__stex_term_custom_arg_inv:wn [ #1 ] #2 {
2294   \bool_set_true:N \l_tmpa_bool
2295   \__stex_term_custom_arg:wn [ #1 ] { #2 }
2296 }

```

(End definition for _stex_term_custom_arg_inv:wn.)

__stex_term_custom_arg:wn

```

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

```

```

2315         \exp_not:n { { #2 } }
2316     }
2317 }
2318 } {
2319     \tl_put_right:Nx \l_tmpa_tl {
2320         \stex_term_arg:nn { \int_eval:n { #1 } }
2321         \exp_not:n { { #2 } }
2322     }
2323 }
2324
2325 \__stex_term_custom_loop:
2326 }

```

(End definition for __stex_term_custom_arg:wn.)

__stex_term_custom_set_X:n

```

2327 \cs_new_protected:Nn \__stex_term_custom_set_X:n {
2328     \str_set:Nx \l_tmpa_str {
2329         \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
2330         X
2331         \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
2332     }
2333 }

```

(End definition for __stex_term_custom_set_X:n.)

__stex_term_custom_component:

```

2334 \cs_new_protected:Npn \__stex_term_custom_component:w [ #1 ] {
2335     \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
2336     \__stex_term_custom_loop:
2337 }

```

(End definition for __stex_term_custom_component:.)

__stex_term_custom_final:

```

2338 \cs_new_protected:Nn \__stex_term_custom_final: {
2339     \int_compare:nNnTF \l_tmpb_int = 0 {
2340         \exp_args:Nnno \stex_term_oms:nnn
2341     }{
2342         \str_if_in:NnTF \l_tmpa_str {b} {
2343             \exp_args:Nnno \stex_term_ombind:nnn
2344         } {
2345             \exp_args:Nnno \stex_term_oma:nnn
2346         }
2347     }
2348     { \l__stex_term_custom_uri } { \l__stex_term_custom_uri } { \l_tmpa_tl }
2349 }

```

(End definition for __stex_term_custom_final:.)

\symref

\symname

```

2350 \NewDocumentCommand \symref { m m }{
2351     \STEXsymbol{#1}![#2]
2352 }
2353

```

```

2354 \keys_define:nn { stex / symname } {
2355   post      .tl_set_x:N    = \l_stex_symname_post_str
2356 }
2357
2358 \cs_new_protected:Nn \stex_symname_args:n {
2359   \str_clear:N \l_stex_symname_post_str
2360   \keys_set:nn { stex / symname } { #1 }
2361   \exp_args:NNNo \str_set:Nn \l_stex_symname_post_str
2362     \l_stex_symname_post_str
2363 }
2364
2365 \NewDocumentCommand \symname { 0{} m }{
2366   \stex_symname_args:n { #1 }
2367   \stex_get_symbol:n { #2 }
2368   \str_set:Nx \l_tmpa_str {
2369     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
2370   }
2371   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {-}
2372   \exp_args:NNx \use:nn
2373   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }![
2374     \l_tmpa_str \l_stex_symname_post_str
2375   ] }
2376 }

```

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

4.9 Notation Components

```

2377 <@@=stex_notationcomps>

```

`\stex_highlight_term:nn`

```

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

```

```

2402 %   } {
2403     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
2404 %   }
2405 % }
2406 }

```

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

```

\comp
\@comp
\@defemph
2407 \cs_new_protected:Npn \comp #1 {
2408   \str_if_empty:NF \l__stex_notationcomps_highlight_uri_str {
2409     \scalatex_if:TF {
2410       \stex_annotate:nnn { comp }{ \l__stex_notationcomps_highlight_uri_str }{ #1 }
2411     }{
2412       \exp_args:Nnx \@comp { #1 } { \l__stex_notationcomps_highlight_uri_str }
2413     }
2414   }
2415 }
2416
2417 \cs_new_protected:Npn \@comp #1 #2 {
2418   % \pdftooltip {
2419     \textcolor{blue}{#1}
2420   % } { #2 }
2421 }
2422
2423 \cs_new_protected:Npn \@defemph #1 #2 {
2424   % \pdftooltip {
2425     \textbf{\textcolor{magenta}{#1}}
2426   % } { #2 }
2427 }

```

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

\ellipses

```

2428 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix
\parrayline
\parraycell
2429 \bool_new:N \l_stex_inparray_bool
2430 \bool_set_false:N \l_stex_inparray_bool
2431 \NewDocumentCommand \parray { m m } {
2432   \begin{group}
2433     \bool_set_true:N \l_stex_inparray_bool
2434     \begin{array}{#1}
2435       #2
2436     \end{array}
2437   \end{group}
2438 }
2439
2440 \NewDocumentCommand \prmatrix { m } {
2441   \begin{group}
2442     \bool_set_true:N \l_stex_inparray_bool
2443     \begin{matrix}
2444       #1

```

```

2445 \end{matrix}
2446 \endgroup
2447 }
2448
2449 \def \parrayline #1 #2 {
2450   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
2451 }
2452
2453 \def \parraycell #1 {
2454   #1 \bool_if:NT \l_stex_inarray_bool {&}
2455 }

```

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

4.10 Structural Features

```

2456 \@@=stex_features

```

symboldoc

```

2457 \NewDocumentEnvironment{symboldoc}{m}{
2458   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
2459   \seq_clear:N \l_tmpb_seq
2460   \seq_map_inline:Nn \l_tmpa_seq {
2461     \stex_get_symbol:n { ##1 }
2462     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
2463       \l_stex_get_symbol_uri_str
2464     }
2465   }
2466   \par
2467   \exp_args:Nnnx
2468   \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
2469 }{
2470   \end{stex_annotate_env}
2471 }

```

STEXdefinition

```

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

```



```

2490     \l_tmpa_str
2491   }
2492 } {
2493   \@defemph {
2494     \l_tmpa_str
2495   } { \l_stex_get_symbol_uri_str }
2496 }
2497 }
2498
2499 \cs_new_protected:Nn \__stex_features_defi_begin:n {
2500   \let\definiendum\__stex_features_definiendum:w
2501   \let\definame\__stex_features_definame:w
2502   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
2503   \seq_clear:N \l_tmpb_seq
2504   \seq_map_inline:Nn \l_tmpa_seq {
2505     \stex_get_symbol:n { ##1 }
2506     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
2507       \l_stex_get_symbol_uri_str
2508     }
2509   }
2510   \exp_args:Nnnx
2511   \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
2512 }
2513
2514 \cs_new_protected:Nn \__stex_features_defi_end: {
2515   \end{stex_annotate_env}
2516 }
2517
2518 \NewDocumentEnvironment{STEXdefinition}{ m m m }{
2519   \__stex_features_defi_begin:n { #1 }
2520 }{
2521   \__stex_features_defi_end:
2522 }

```

\setSTEXdefinition

```

2523 \cs_new_protected:Npn \setSTEXdefinition #1 {
2524   \AddToHook{env/#1/before}[stex]{\__stex_features_defi_begin:n{}}
2525   \AddToHook{env/#1/after}[stex]{\__stex_features_defi_end:}
2526 }

```

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

structural@feature

```

2527
2528 \NewDocumentEnvironment{structural@feature}{ m m m }{
2529   \stex_if_in_module:F {
2530     \msg_set:nnn{stex}{error/nomodule}{
2531       Structural~Feature~has~to~occur~in~a~module:\\
2532       Feature~#2~of~type~#1\\
2533       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
2534     }
2535     \msg_error:nn{stex}{error/nomodule}
2536   }
2537 }

```

```

2538 \str_set:Nx \l_stex_module_name_str {
2539   \prop_item:Nn \l_stex_current_module_prop
2540     { name } / #2 - feature
2541 }
2542
2543
2544 \str_clear:N \l_tmpa_str
2545 \seq_clear:N \l_tmpa_seq
2546 \tl_clear:N \l_tmpa_tl
2547 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
2548   origname = #2,
2549   name      = \l_stex_module_name_str ,
2550   ns        = \l_stex_module_ns_str ,
2551   imports   = \exp_not:o { \l_tmpa_seq } ,
2552   constants = \exp_not:o { \l_tmpa_seq } ,
2553   content   = \exp_not:o { \l_tmpa_tl } ,
2554   file      = \exp_not:o { \g_stex_currentfile_seq } ,
2555   lang      = \l_stex_module_lang_str ,
2556   sig       = \l_tmpa_str ,
2557   meta      = \l_tmpa_str ,
2558   feature   = #1 ,
2559 }
2560
2561 \stex_if_smsmode:TF {
2562   \stex_smsmode_set_codes:
2563 } {
2564   \begin{stex_annotate_env}{ feature:#1 }{}
2565   \stex_annotate_invisible:nnn{header}{}{ #3 }
2566 }
2567 }{
2568   \str_set:Nx \l_tmpa_str {
2569     c_stex_feature_
2570     \prop_item:Nn \l_stex_current_module_prop { ns } ?
2571     \prop_item:Nn \l_stex_current_module_prop { name }
2572     _prop
2573   }
2574   \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
2575   \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
2576   \stex_if_smsmode:TF {
2577     \exp_args:Nx \stex_addtosms:n {
2578       \prop_gset_from_keyval:cn {
2579         c_stex_feature_
2580         \prop_item:Nn \l_stex_current_module_prop { ns } ?
2581         \prop_item:Nn \l_stex_current_module_prop { name }
2582         _prop
2583       } {
2584         origname = #2,
2585         name      = \prop_item:cn { \l_tmpa_str } { name } ,
2586         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
2587         imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
2588         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
2589         content   = \prop_item:cn { \l_tmpa_str } { content } ,
2590         file      = \prop_item:cn { \l_tmpa_str } { file } ,
2591         lang      = \prop_item:cn { \l_tmpa_str } { lang } ,

```

```

2592         sig      = \prop_item:cn { \l_tmpa_str } { sig } ,
2593         meta      = \prop_item:cn { \l_tmpa_str } { meta } ,
2594         feature    = \prop_item:cn { \l_tmpa_str } { feature }
2595     }
2596 }
2597 } {
2598     \end{stex_annotate_env}
2599 }
2600 }
2601

```

structure

```

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

```

```

2644 \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
2645 }
2646 \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
2647 \exp_args:NNx
2648 \AddToHookNext { env / structure / after }{
2649 \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}}{
2650 \_stex_term_math_oms:nnnn { \l_tmpa_str }{}{0}{}
2651 }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }}{ #2 }
2652 \STEXexport {
2653 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2654 {\prop_item:Nn \l_stex_current_module_prop { origname }}
2655 {\l_tmpa_str}
2656 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2657 {#2}{\l_tmpa_str}
2658 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
2659 % \prop_item:Nn \l_stex_current_module_prop { origname },
2660 % \l_tmpa_str
2661 % }
2662 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
2663 % #2,\l_tmpa_str
2664 % }
2665 % \tl_set:cx { #2 } {
2666 % \stex_invoke_structure:n { \l_tmpa_str }
2667 }
2668 }
2669
2670 \end{structural@feature}
2671 % \g_stex_last_feature_prop
2672 }

```

\instantiate

```

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

```

```

2696         \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
2697         \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
2698         \l_tmpa_tl
2699         \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
2700             \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
2701             \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
2702         }{
2703             \tl_clear:N \l_tmpb_tl
2704         }
2705     }
2706 }{
2707     \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
2708     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
2709         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
2710         \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
2711         \tl_clear:N \l_tmpa_tl
2712     }{
2713         % TODO throw error
2714     }
2715 }
2716 % \l_tmpa_str: name
2717 % \l_tmpa_tl: definiens
2718 % \l_tmpb_tl: notation
2719 \tl_if_empty:NT \l__stex_features_structure_field_str {
2720     % TODO throw error
2721 }
2722 \str_clear:N \l_tmpb_str
2723
2724 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
2725 \seq_map_inline:Nn \l_tmpa_seq {
2726     \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
2727     \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
2728     \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
2729         \seq_map_break:n {
2730             \str_set:Nn \l_tmpb_str { ####1 }
2731         }
2732     }
2733 }
2734 \prop_get:cnN { g_stex_symdecl_ \l_tmpb_str _prop } {args}
2735 \l_tmpb_str
2736
2737 \tl_if_empty:NNTF \l_tmpb_tl {
2738     \tl_if_empty:NF \l_tmpa_tl {
2739         \exp_args:Nx \use:n {
2740             \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fe
2741         }
2742     }
2743 }{
2744     \tl_if_empty:NNTF \l_tmpa_tl {
2745         \exp_args:Nx \use:n {
2746             \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\
2747         }
2748     }
2749 }{

```

```

2750         \exp_args:Nx \use:n {
2751             \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
2752             \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
2753         }
2754     }
2755 }
2756 % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
2757 % \prop_item:Nn \l_stex_current_module_prop {name} ?
2758 % #3/\l__stex_features_structure_field_str
2759 % \par
2760 % \expandafter\present\csname
2761 %     g_stex_symdecl_
2762 %     \prop_item:Nn \l_stex_current_module_prop {ns} ?
2763 %     \prop_item:Nn \l_stex_current_module_prop {name} ?
2764 %     #3/\l__stex_features_structure_field_str
2765 %     _prop
2766 % \endcsname
2767 }
2768
2769 \tl_clear:N \l__stex_features_structure_def_tl
2770
2771 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
2772 \seq_map_inline:Nn \l_tmpa_seq {
2773     \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
2774     \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
2775     \exp_args:Nx \use:n {
2776         \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
2777
2778         }
2779     }
2780
2781     \prop_if_exist:cF {
2782         g_stex_symdecl_
2783         \prop_item:Nn \l_stex_current_module_prop {ns} ?
2784         \prop_item:Nn \l_stex_current_module_prop {name} ?
2785         #3/\l_tmpa_str
2786         _prop
2787     }{
2788         \prop_get:cnN { g_stex_symdecl_ ##1 _prop } {args}
2789         \l_tmpb_str
2790         \exp_args:Nx \use:n {
2791             \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
2792         }
2793     }
2794 }
2795
2796 \symdecl*[type={\STEXsymbol{module-type}}{
2797     \_stex_term_math_oms:nnnn {
2798         \prop_item:Nn \l__stex_features_structure_prop {ns} ?
2799         \prop_item:Nn \l__stex_features_structure_prop {name}
2800     }{}{0}{}
2801 }{}]{#3}
2802
2803 % TODO: -> sms file

```

```

2804
2805 \tl_set:cx{ #3 }{
2806   \stex_invoke_structure:nnn {
2807     \prop_item:Nn \l_stex_current_module_prop {ns} ?
2808     \prop_item:Nn \l_stex_current_module_prop {name} ? #3
2809   } {
2810     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
2811     \prop_item:Nn \l__stex_features_structure_prop {name}
2812   }
2813 }
2814
2815 }

```

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

\stex_invoke_structure:nnn

```

2816 % #1: URI of the instance
2817 % #2: URI of the instantiated module
2818 \cs_new_protected:Nn \stex_invoke_structure:nnn {
2819   \tl_if_empty:nTF{ #3 }{
2820     \prop_set_eq:Nc \l__stex_features_structure_prop {
2821       c_stex_feature_ #2 _prop
2822     }
2823     \tl_clear:N \l_tmpa_tl
2824     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
2825     \seq_map_inline:Nn \l_tmpa_seq {
2826       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
2827       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
2828       \cs_if_exist:cT {
2829         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
2830       }{
2831         \tl_if_empty:NF \l_tmpa_tl {
2832           \tl_put_right:Nn \l_tmpa_tl {,}
2833         }
2834         \tl_put_right:Nx \l_tmpa_tl {
2835           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
2836         }
2837       }
2838     }
2839     \scalatexBREAK
2840     \exp_args:No \mathstrut \l_tmpa_tl
2841   }{
2842     \stex_invoke_symbol:n{#1/#3}
2843   }
2844 }

```

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

4.11 Put these somewhere

\MSC

```

2845 \NewDocumentCommand \MSC {m} {
2846   % TODO
2847 }

```



```

2888 %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
2889 %\notation[lambda]{mapto}{\comp\lambda #1 \comp.\; #2}{#1 \comp, #2}
2890 %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp.\; #2}{#1 \comp, #2}
2891
2892 % function/operator application
2893 \symdecl[args=ia]{apply}
2894 \notation[prec=0;0x\neginfprec,parens]{apply}{#1 \comp( #2 \comp)}{#1 \comp, #2}
2895 \notation[prec=0;0x\neginfprec,lambda]{apply}{#1 \; #2 }{#1 \; #2}
2896
2897 % ‘‘type’’ of all collections (sets, classes, types, kinds)
2898 \symdecl{collection}
2899 \notation[U]{collection}{\comp{\mathcal{U}}}
2900 \notation[set]{collection}{\comp{\textsf{Set}}}
2901
2902 % sequences
2903 \symdecl[args=1]{seqtype}
2904 \notation[kleene]{seqtype}{#1^{\comp\ast}}
2905
2906 \symdef[args=2,li]{sequence-index}{#1_{#2}}
2907 \notation[ui]{sequence-index}{#1^{\#2}}
2908
2909 %\symdef[args=3,li]{sequence-from-to}{#1_{#2}\comp{\,\ellipses},#1_{#3}}
2910 %\notation[ui]{sequence-from-to}{#1^{\#2}\comp{\,\ellipses},#1^{\#3}}
2911 % ^ superceded by \aseqfromto and \livar/\uivar
2912
2913 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{\,\ellipses}}{#1\comp,#2}
2914 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{\,\ellipses\comp,}#2 }{#1\comp,#2}
2915
2916 % letin (‘‘let’’, local definitions, variable substitution)
2917 \symdecl[args=bii]{letin}
2918 \notation[let]{letin}{\comp{\rm let}}\;#1\comp{=}\#2\;\comp{\rm in}}\;#3}
2919 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
2920 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
2921
2922 % structures
2923 \symdecl*[args=1]{module-type}
2924 \notation{module-type}{\mathtt{MOD} #1}
2925 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
2926 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{#1 \comp, #2}
2927
2928 \STEXexport{
2929   \let\nappa\apply
2930   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
2931   \def\livar{\csname sequence-index\endcsname[li]}
2932   \def\uivar{\csname sequence-index\endcsname[ui]}
2933   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
2934   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
2935 }
2936
2937 \end{@module}
2938 \ExplSyntaxOff
2939 </metatheory>

```

4.13 Auxiliary Packages

4.13.1 tikzinput

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

```

2991 \RequirePackage{stex-tikzinput}
2992 }{}
2993 </tikzinput>
2994 <*stex-tikzinput>
2995 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
2996 \RequirePackage{stex}
2997 \RequirePackage{tikzinput}
2998
2999 % TODO
3000
3001 </stex-tikzinput>

```

4.13.2 sTeX1 Compatibility

```

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

```

```

3043 \str_if_eq:NNTF \l_tmpa_str \l_tmpb_str {
3044 \prop_set_eq:NN \l_modsig_old_module_prop \l_stex_current_module_prop
3045 \begin{@module}{modsig-#2}
3046 % \prop_set_eq:NN \l_stex_current_module_prop \l_modsig_old_module_prop
3047 } {
3048 \begin{@module}{#2}
3049 }
3050 } {
3051 \begin{@module}{#2}
3052 }
3053 }{
3054 \end{@module}
3055 \AddToHookNext { env / modsig / after }{
3056 \stex_if_in_module:T {
3057 \prop_get:NnN \l_stex_current_module_prop {name} \l_tmpa_str
3058 \str_set:Nn \l_tmpb_str { #2 }
3059 \str_if_eq:NNT \l_tmpa_str \l_tmpb_str {
3060 % \xdef \g_stex_module_after_group_tl {
3061 \stex_if_smsmode:TF {
3062 \exp_args:Nx
3063 \stex_add_to_current_module:n {
3064 \stex_debug:n{Activating~signature~of~#2}
3065 \exp_not:N \prop_item:cn { c_stex_module_
3066 \prop_item:Nn \l_stex_current_module_prop {ns} ?
3067 \prop_item:Nn \l_stex_current_module_prop {name}
3068 / modsig-#2_prop } { content }
3069 }
3070 }
3071 {
3072 \gdef \g_stex_modsig_after_group_tl {
3073 \stex_activate_module:n {
3074 \prop_item:Nn \l_stex_current_module_prop {ns} ?
3075 \prop_item:Nn \l_stex_current_module_prop {name}
3076 / modsig-#2
3077 }
3078
3079 \exp_args:Nx
3080 \stex_add_to_current_module:n {
3081 \stex_activate_module:n {
3082 \prop_item:Nn \l_stex_current_module_prop {ns} ?
3083 \prop_item:Nn \l_stex_current_module_prop {name}
3084 / modsig-#2
3085 }
3086 }
3087 }
3088 \aftergroup \g_stex_modsig_after_group_tl
3089 }
3090 }
3091 }
3092 }
3093 }
3094
3095 \cs_new_protected:Npn \gimport {
3096 \peek_charcode_remove:NTF * {

```

```

3097     \gimport_do:
3098   } {
3099     \gimport_do:
3100   }
3101 }
3102
3103 \NewDocumentCommand \gimport_do: { 0{} m } {
3104   \msg_set:nnn{stex}{warning/deprecated}{
3105     \\\
3106     \c_backslash_str gimport~is~deprecated! \\\
3107     Please~use~\c_backslash_str importmodule[#1]{#2}~instead!~(in~file~
3108     \stex_path_to_string:N \g_stex_currentfile_seq)
3109     \\\ \\\
3110   }
3111   \msg_warning:nn{stex}{warning/deprecated}
3112   \importmodule[#1]{#2}
3113 }
3114
3115 \cs_new_protected:Npn \guse {
3116   \peek_charcode_remove:NTF * {
3117     \guse_do:
3118   } {
3119     \guse_do:
3120   }
3121 }
3122
3123 \NewDocumentCommand \guse_do: { 0{} m } {
3124   \msg_set:nnn{stex}{warning/deprecated}{
3125     \\\
3126     \c_backslash_str guse~is~deprecated! \\\
3127     Please~use~\c_backslash_str usemodule[#1]{#2}~instead!~(in~file~
3128     \stex_path_to_string:N \g_stex_currentfile_seq)
3129     \\\ \\\
3130   }
3131   \msg_warning:nn{stex}{warning/deprecated}
3132   \usemodule[#1]{#2}
3133 }
3134
3135 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3136
3137 \cs_new_protected:Npn \symi {
3138   \peek_charcode_remove:NTF * {
3139     \symi_do:
3140   } {
3141     \symi_do:
3142   }
3143 }
3144
3145 \NewDocumentCommand \symi_do: { 0{} m } {
3146   \msg_set:nnn{stex}{warning/deprecated}{
3147     \\\
3148     \c_backslash_str symi~is~deprecated! \\\
3149     Please~use~\c_backslash_str symdecl[#1]{#2}~instead!~(in~file~
3150     \stex_path_to_string:N \g_stex_currentfile_seq)

```

```

3151     \\\ \\\
3152   }
3153   \msg_warning:nn{stex}{warning/deprecated}
3154   \symdecl*{#1}{#2}
3155 }
3156
3157 \cs_new_protected:Npn \symii {
3158   \peek_charcode_remove:NTF * {
3159     \symii_do:
3160   } {
3161     \symii_do:
3162   }
3163 }
3164
3165 \NewDocumentCommand \symii_do: { 0{} m m } {
3166   \msg_set:nnn{stex}{warning/deprecated}{
3167     \\\
3168     \c_backslash_str symii~is~deprecated! \\\
3169     Please~use~\c_backslash_str symdecl{#1}{#2-#3}~instead!~(in~file~
3170     \stex_path_to_string:N \g_stex_currentfile_seq)
3171     \\\ \\\
3172   }
3173   \msg_warning:nn{stex}{warning/deprecated}
3174   \symdecl*{#1}{#2-#3}
3175 }
3176
3177 \cs_new_protected:Npn \symiii {
3178   \peek_charcode_remove:NTF * {
3179     \symiii_do:
3180   } {
3181     \symiii_do:
3182   }
3183 }
3184
3185 \NewDocumentCommand \symiii_do: { 0{} m m m } {
3186   \msg_set:nnn{stex}{warning/deprecated}{
3187     \\\
3188     \c_backslash_str symiii~is~deprecated! \\\
3189     Please~use~\c_backslash_str symdecl{#1}{#2-#3-#4}~instead!~(in~file~
3190     \stex_path_to_string:N \g_stex_currentfile_seq)
3191     \\\ \\\
3192   }
3193   \msg_warning:nn{stex}{warning/deprecated}
3194   \symdecl*{#1}{#2-#3-#4}
3195 }
3196
3197 \keys_define:nn { stex / deprec / defi } {
3198   name .tl_set_x:N = \l_tmpa_str
3199 }
3200
3201 \cs_new_protected:Npn \defi {
3202   \peek_charcode_remove:NTF * {
3203     \defi_do:
3204   } {

```

```

3205     \defi_do:
3206   }
3207 }
3208
3209 \NewDocumentCommand \defi_do: { 0{} m } {
3210   \str_clear:N \l_tmpa_str
3211   \keys_set:nn { stex / deprec / defi } { #1 }
3212
3213   \str_if_empty:NTF \l_tmpa_str {
3214     \msg_set:nnn{stex}{warning/deprecated}{
3215       \\\
3216       \c_backslash_str defi-is~deprecated! \\\
3217       Please~use~\c_backslash_str STEXsymbol{#2}![#2]~instead!~(in~file~
3218       \stex_path_to_string:N \g_stex_currentfile_seq)
3219       \\\ \\\
3220     }
3221     \msg_warning:nn{stex}{warning/deprecated}
3222     \STEXsymbol { #2 }![ \comp{#2} ]
3223   } {
3224     \msg_set:nnn{stex}{warning/deprecated}{
3225       \\\
3226       \c_backslash_str defi-is~deprecated! \\\
3227       Please~use~\c_backslash_str STEXsymbol { \l_tmpa_str }[ #2 ]~instead!~(in~file~
3228       \stex_path_to_string:N \g_stex_currentfile_seq)
3229       \\\ \\\
3230     }
3231     \msg_warning:nn{stex}{warning/deprecated}
3232     \exp_args:No \STEXsymbol { \l_tmpa_str }![ \comp{#2} ]
3233   }
3234 }
3235
3236
3237 \cs_new_protected:Npn \Defi {
3238   \peek_charcode_remove:NTF * {
3239     \Defi_do:
3240   } {
3241     \Defi_do:
3242   }
3243 }
3244
3245 \NewDocumentCommand \Defi_do: { 0{} m } {
3246   \str_clear:N \l_tmpa_str
3247   \keys_set:nn { stex / deprec / defi } { #1 }
3248
3249   \str_if_empty:NTF \l_tmpa_str {
3250     \msg_set:nnn{stex}{warning/deprecated}{
3251       \\\
3252       \c_backslash_str Defi-is~deprecated! \\\
3253       Please~use~\c_backslash_str STEXsymbol{#2}![\exp_after:wN \stex_capitalize:n #2]~inste
3254       \stex_path_to_string:N \g_stex_currentfile_seq)
3255       \\\ \\\
3256     }
3257     \msg_warning:nn{stex}{warning/deprecated}
3258     \STEXsymbol { #2 }![ \comp{\exp_after:wN \stex_capitalize:n #2} ]

```

```

3259 } {
3260   \msg_set:nnn{stex}{warning/deprecated}{
3261     \\\
3262     \c_backslash_str Defi~is~deprecated! \\\
3263     Please~use~\c_backslash_str STExsymbol { \l_tmpa_str }[ \exp_after:wN \stex_capitalize
3264     \stex_path_to_string:N \g_stex_currentfile_seq)
3265     \\\ \\\
3266   }
3267   \msg_warning:nn{stex}{warning/deprecated}
3268   \exp_args:No \STExsymbol { \l_tmpa_str }![ \comp{\exp_after:wN \stex_capitalize:n #2} ]
3269 }
3270 }
3271
3272 \cs_new_protected:Npn \adefi {
3273   \peek_charcode_remove:NTF * {
3274     \adefi_do:
3275   } {
3276     \adefi_do:
3277   }
3278 }
3279
3280 \NewDocumentCommand \adefi_do: { 0{} m m } {
3281   \str_clear:N \l_tmpa_str
3282   \keys_set:nn { stex / deprec / defi } { #1 }
3283
3284   \str_if_empty:NTF \l_tmpa_str {
3285     \msg_set:nnn{stex}{warning/deprecated}{
3286       \\\
3287       \c_backslash_str adefi~is~deprecated! \\\
3288       Please~use~\c_backslash_str STExsymbol{#3}![#2]~instead!~(in~file~
3289       \stex_path_to_string:N \g_stex_currentfile_seq)
3290       \\\ \\\
3291     }
3292     \msg_warning:nn{stex}{warning/deprecated}
3293     \STExsymbol { #3 }![ \comp{#2} ]
3294   } {
3295     \msg_set:nnn{stex}{warning/deprecated}{
3296       \\\
3297       \c_backslash_str adefi~is~deprecated! \\\
3298       Please~use~\c_backslash_str STExsymbol { \l_tmpa_str }[ #2 ]~instead!~(in~file~
3299       \stex_path_to_string:N \g_stex_currentfile_seq)
3300       \\\ \\\
3301     }
3302     \msg_warning:nn{stex}{warning/deprecated}
3303     \exp_args:No \STExsymbol { \l_tmpa_str }![ \comp{#2} ]
3304   }
3305 }
3306
3307 \cs_new_protected:Npn \defis {
3308   \peek_charcode_remove:NTF * {
3309     \defis_do:
3310   } {
3311     \defis_do:
3312   }

```



```

3313 }
3314
3315 \NewDocumentCommand \defis_do: { 0{} m } {
3316   \str_clear:N \l_tmpa_str
3317   \keys_set:nn { stex / deprec / defi } { #1 }
3318
3319   \str_if_empty:NTF \l_tmpa_str {
3320     \msg_set:nnn{stex}{warning/deprecated}{
3321       \\\
3322       \c_backslash_str defis-is-deprecated! \\\
3323       Please~use~\c_backslash_str STExsymbol{#2}![#2s]~instead!~(in~file~
3324       \stex_path_to_string:N \g_stex_currentfile_seq)
3325       \\\ \\\
3326     }
3327     \msg_warning:nn{stex}{warning/deprecated}
3328     \STExsymbol { #2 }![ \comp{#2s} ]
3329   } {
3330     \msg_set:nnn{stex}{warning/deprecated}{
3331       \\\
3332       \c_backslash_str defis-is-deprecated! \\\
3333       Please~use~\c_backslash_str STExsymbol { \l_tmpa_str }[ #2s ]~instead!~(in~file~
3334       \stex_path_to_string:N \g_stex_currentfile_seq)
3335       \\\ \\\
3336     }
3337     \msg_warning:nn{stex}{warning/deprecated}
3338     \exp_args:No \STExsymbol { \l_tmpa_str }![ \comp{#2s} ]
3339   }
3340 }
3341
3342 \cs_new_protected:Npn \defii {
3343   \peek_charcode_remove:NTF * {
3344     \defii_do:
3345   } {
3346     \defii_do:
3347   }
3348 }
3349
3350 \NewDocumentCommand \defii_do: { 0{} m m } {
3351   \str_clear:N \l_tmpa_str
3352   \keys_set:nn { stex / deprec / defi } { #1 }
3353   \str_if_empty:NTF \l_tmpa_str {
3354     \msg_set:nnn{stex}{warning/deprecated}{
3355       \\\
3356       \c_backslash_str defii-is-deprecated! \\\
3357       Please~use~\c_backslash_str STExsymbol{#2-#3}![#2~#3]~instead!~(in~file~
3358       \stex_path_to_string:N \g_stex_currentfile_seq)
3359       \\\ \\\
3360     }
3361     \msg_warning:nn{stex}{warning/deprecated}
3362     \STExsymbol { #2-#3 }![ \comp{#2~#3} ]
3363   } {
3364     \msg_set:nnn{stex}{warning/deprecated}{
3365       \\\
3366       \c_backslash_str defii-is-deprecated! \\\

```

```

3367     Please~use~\c_backslash_str STExsymbol { \l_tmpa_str }[ #2~#3 ]~instead!~(in~file~
3368     \stex_path_to_string:N \g_stex_currentfile_seq)
3369     \\\ \\\
3370   }
3371   \msg_warning:nn{stex}{warning/deprecated}
3372   \exp_args:No \STExsymbol { \l_tmpa_str }![ \comp{#2~#3} ]
3373 }
3374 }
3375
3376
3377 \cs_new_protected:Npn \defiis {
3378   \peek_charcode_remove:NTF * {
3379     \defiis_do:
3380   } {
3381     \defiis_do:
3382   }
3383 }
3384
3385 \NewDocumentCommand \defiis_do: { O{} m m } {
3386   \str_clear:N \l_tmpa_str
3387   \keys_set:nn { stex / deprec / defi } { #1 }
3388   \str_if_empty:NTF \l_tmpa_str {
3389     \msg_set:nnn{stex}{warning/deprecated}{
3390       \\\
3391       \c_backslash_str defiis~is~deprecated! \\\
3392       Please~use~\c_backslash_str STExsymbol{#2~#3}![#2~#3s]~instead!~(in~file~
3393       \stex_path_to_string:N \g_stex_currentfile_seq)
3394       \\\ \\\
3395     }
3396     \msg_warning:nn{stex}{warning/deprecated}
3397     \STExsymbol { #2~#3 }![ \comp{#2~#3s} ]
3398   } {
3399     \msg_set:nnn{stex}{warning/deprecated}{
3400       \\\
3401       \c_backslash_str defiis~is~deprecated! \\\
3402       Please~use~\c_backslash_str STExsymbol { \l_tmpa_str }[ #2~#3s ]~instead!~(in~file~
3403       \stex_path_to_string:N \g_stex_currentfile_seq)
3404       \\\ \\\
3405     }
3406     \msg_warning:nn{stex}{warning/deprecated}
3407     \exp_args:No \STExsymbol { \l_tmpa_str }![ \comp{#2~#3s} ]
3408   }
3409 }
3410
3411
3412 \cs_new_protected:Npn \defiii {
3413   \peek_charcode_remove:NTF * {
3414     \defiii_do:
3415   } {
3416     \defiii_do:
3417   }
3418 }
3419
3420 \NewDocumentCommand \defiii_do: { O{} m m m } {

```

```

3421 \str_clear:N \l_tmpa_str
3422 \keys_set:nn { stex / deprec / defi } { #1 }
3423 \str_if_empty:NTF \l_tmpa_str {
3424   \msg_set:nnn{stex}{warning/deprecated}{
3425     \\\
3426     \c_backslash_str defiii~is-deprecated! \\\
3427     Please~use~\c_backslash_str STExsymbol{#2~#3~#4}![#2~#3~#4]~instead!~(in~file~
3428     \stex_path_to_string:N \g_stex_currentfile_seq)
3429     \\\ \\\
3430   }
3431   \msg_warning:nn{stex}{warning/deprecated}
3432   \STExsymbol { #2~#3~#4 }![ \comp{#2~#3~#4} ]
3433 } {
3434   \msg_set:nnn{stex}{warning/deprecated}{
3435     \\\
3436     \c_backslash_str defiii~is-deprecated! \\\
3437     Please~use~\c_backslash_str STExsymbol { \l_tmpa_str }[ #2~#3~#4 ]~instead!~(in~file~
3438     \stex_path_to_string:N \g_stex_currentfile_seq)
3439     \\\ \\\
3440   }
3441   \msg_warning:nn{stex}{warning/deprecated}
3442   \exp_args:No \STExsymbol { \l_tmpa_str }![ \comp{#2~#3~#4} ]
3443 }
3444 }
3445
3446 %\RequirePackage[hyperref]{ntheorem}
3447 %\theoremstyle{plain}
3448 %\RequirePackage{amsthm}
3449
3450 \NewDocumentEnvironment {definition} { 0{ } } {
3451   \begin{STExdefinition}{ }
3452 }{
3453   \end{STExdefinition}
3454 }
3455 \keys_define:nn { stex / omtex } {
3456   title .tl_set_x:N = \l_stex_omtext_title_str
3457 }
3458 \cs_new_protected:Nn \stex_omtext_args:n {
3459   \str_clear:N \l_stex_omtext_title_str
3460   \keys_set:nn { stex / omtex } { #1 }
3461   \exp_args:NNo \str_set:Nn \l_stex_omtext_title_str
3462     \l_stex_omtext_title_str
3463 }
3464 \NewDocumentEnvironment {omtext} { 0{ } } {
3465   \stex_omtext_args:n { #1 }
3466   \paragraph{\l_stex_omtext_title_str}
3467 }{
3468 }
3469
3470 \NewDocumentEnvironment {assertion} { 0{ } } {
3471 }{
3472 }{
3473 }
3474 }

```

```

3475
3476 \NewDocumentCommand \inlinedef { m } {
3477   \begingroup
3478   \let\definiendum\_stex_deprec_definiendum:w
3479   \let\definame\_stex_deprec_definame:w
3480   #1
3481   \endgroup
3482 }
3483
3484 \NewDocumentCommand \inlineass { m } { #1 }
3485
3486 \NewDocumentCommand \trefi { O{} m } {
3487   \str_set:Nn \l_tmpa_str { #1 }
3488   \str_if_empty:NTF \l_tmpa_str {
3489     \msg_set:nnn{stex}{warning/deprecated}{
3490       \\
3491       \c_backslash_str trefi-is-deprecated! \\
3492       Please~use~\c_backslash_str STEXsymbol{#2}![#2]~instead!~(in~file~
3493       \stex_path_to_string:N \g_stex_currentfile_seq)
3494       \\ \\
3495     }
3496     \msg_warning:nn{stex}{warning/deprecated}
3497     \STEXsymbol { #2 }![ \comp{#2} ]
3498   } {
3499     \msg_set:nnn{stex}{warning/deprecated}{
3500       \\
3501       \c_backslash_str trefi-is-deprecated! \\
3502       Please~use~\c_backslash_str STEXsymbol { #1?#2 }[ #2 ]~instead!~(in~file~
3503       \stex_path_to_string:N \g_stex_currentfile_seq)
3504       \\ \\
3505     }
3506     \msg_warning:nn{stex}{warning/deprecated}
3507     \STEXsymbol { #1 }![ \comp{#2} ]
3508   }
3509 }
3510
3511
3512 \NewDocumentCommand \Trefi { O{} m } {
3513   \str_set:Nn \l_tmpa_str { #1 }
3514   \str_if_empty:NTF \l_tmpa_str {
3515     \msg_set:nnn{stex}{warning/deprecated}{
3516       \\
3517       \c_backslash_str Trefi-is-deprecated! \\
3518       Please~use~\c_backslash_str STEXsymbol{#2}![\exp_after:wN \stex_capitalize:n #2]~inste
3519       \stex_path_to_string:N \g_stex_currentfile_seq)
3520       \\ \\
3521     }
3522     \msg_warning:nn{stex}{warning/deprecated}
3523     \STEXsymbol { #2 }![ \comp{\exp_after:wN \stex_capitalize:n #2} ]
3524   } {
3525     \msg_set:nnn{stex}{warning/deprecated}{
3526       \\
3527       \c_backslash_str Trefi-is-deprecated! \\
3528       Please~use~\c_backslash_str STEXsymbol { #1 }[ \exp_after:wN \stex_capitalize:n #2 ]~i

```

```

3529     \stex_path_to_string:N \g_stex_currentfile_seq)
3530     \\\ \
3531   }
3532   \msg_warning:nn{stex}{warning/deprecated}
3533   \STEXsymbol { #1 }![ \comp{\exp_after:wN \stex_capitalize:n #2} ]
3534 }
3535 }
3536
3537 \NewDocumentCommand \trefis { O{} m } {
3538   \str_set:Nn \l_tmpa_str { #1 }
3539   \str_if_empty:NTF \l_tmpa_str {
3540     \msg_set:nnn{stex}{warning/deprecated}{
3541       \\\
3542       \c_backslash_str trefi-is-deprecated! \\\
3543       Please~use~\c_backslash_str STEXsymbol{#2}![#2s]~instead!~(in~file~
3544       \stex_path_to_string:N \g_stex_currentfile_seq)
3545       \\\ \
3546     }
3547     \msg_warning:nn{stex}{warning/deprecated}
3548     \STEXsymbol { #2 }![ \comp{#2s} ]
3549   } {
3550     \msg_set:nnn{stex}{warning/deprecated}{
3551       \\\
3552       \c_backslash_str trefi-is-deprecated! \\\
3553       Please~use~\c_backslash_str STEXsymbol { #1 }[ #2s ]~instead!~(in~file~
3554       \stex_path_to_string:N \g_stex_currentfile_seq)
3555       \\\ \
3556     }
3557     \msg_warning:nn{stex}{warning/deprecated}
3558     \STEXsymbol { #1 }![ \comp{#2s} ]
3559   }
3560 }
3561
3562
3563 \NewDocumentCommand \Trefis { O{} m } {
3564   \str_set:Nn \l_tmpa_str { #1 }
3565   \str_if_empty:NTF \l_tmpa_str {
3566     \msg_set:nnn{stex}{warning/deprecated}{
3567       \\\
3568       \c_backslash_str Trefis-is-deprecated! \\\
3569       Please~use~\c_backslash_str STEXsymbol{#2}![ \exp_after:wN \stex_capitalize:n #2s ]~inst
3570       \stex_path_to_string:N \g_stex_currentfile_seq)
3571       \\\ \
3572     }
3573     \msg_warning:nn{stex}{warning/deprecated}
3574     \STEXsymbol { #2 }![ \comp{\exp_after:wN \stex_capitalize:n #2s} ]
3575   } {
3576     \msg_set:nnn{stex}{warning/deprecated}{
3577       \\\
3578       \c_backslash_str Trefis-is-deprecated! \\\
3579       Please~use~\c_backslash_str STEXsymbol { #1 }[ \exp_after:wN \stex_capitalize:n #2s ]~
3580       \stex_path_to_string:N \g_stex_currentfile_seq)
3581       \\\ \
3582     }

```

```

3583 \msg_warning:nn{stex}{warning/deprecated}
3584 \STEXsymbol { #1 }![ \comp{\exp_after:wN \stex_capitalize:n #2s} ]
3585 }
3586 }
3587
3588 \NewDocumentCommand \trefii { 0{} m m } {
3589 \str_set:Nn \l_tmpa_str { #1 }
3590 \str_if_empty:NTF \l_tmpa_str {
3591 \msg_set:nnn{stex}{warning/deprecated}{
3592 \\\
3593 \c_backslash_str trefii~is-deprecated! \\\
3594 Please~use~\c_backslash_str STEXsymbol{#2~#3}![#2~#3]~instead!~(in~file~
3595 \stex_path_to_string:N \g_stex_currentfile_seq)
3596 \\\ \\\
3597 }
3598 \msg_warning:nn{stex}{warning/deprecated}
3599 \STEXsymbol { #2~#3 }![ \comp{#2~#3} ]
3600 } {
3601 \msg_set:nnn{stex}{warning/deprecated}{
3602 \\\
3603 \c_backslash_str trefii~is-deprecated! \\\
3604 Please~use~\c_backslash_str STEXsymbol { #1 }[ #2~#3 ]~instead!~(in~file~
3605 \stex_path_to_string:N \g_stex_currentfile_seq)
3606 \\\ \\\
3607 }
3608 \msg_warning:nn{stex}{warning/deprecated}
3609 \STEXsymbol { #1 }![ \comp{#2~#3} ]
3610 }
3611 }
3612
3613 \NewDocumentCommand \trefiii { 0{} m m m } {
3614 \str_set:Nn \l_tmpa_str { #1 }
3615 \str_if_empty:NTF \l_tmpa_str {
3616 \msg_set:nnn{stex}{warning/deprecated}{
3617 \\\
3618 \c_backslash_str trefiii~is-deprecated! \\\
3619 Please~use~\c_backslash_str STEXsymbol{#2~#3~#4}![#2~#3~#4]~instead!~(in~file~
3620 \stex_path_to_string:N \g_stex_currentfile_seq)
3621 \\\ \\\
3622 }
3623 \msg_warning:nn{stex}{warning/deprecated}
3624 \STEXsymbol { #2~#3~#4 }![ \comp{#2~#3~#4} ]
3625 } {
3626 \msg_set:nnn{stex}{warning/deprecated}{
3627 \\\
3628 \c_backslash_str trefiii~is-deprecated! \\\
3629 Please~use~\c_backslash_str STEXsymbol { #1 }[ #2~#3~#4 ]~instead!~(in~file~
3630 \stex_path_to_string:N \g_stex_currentfile_seq)
3631 \\\ \\\
3632 }
3633 \msg_warning:nn{stex}{warning/deprecated}
3634 \STEXsymbol { #1 }![ \comp{#2~#3~#4} ]
3635 }
3636 }

```

```

3637
3638
3639 \NewDocumentCommand \treffiis { 0{} m m } {
3640   \str_set:Nn \l_tmpa_str { #1 }
3641   \str_if_empty:NTF \l_tmpa_str {
3642     \msg_set:nnn{stex}{warning/deprecated}{
3643       \\
3644       \c_backslash_str treffiis~is~deprecated! \\
3645       Please~use~\c_backslash_str STEXsymbol{#2~#3}![#2~#3s]~instead!~(in~file~
3646       \stex_path_to_string:N \g_stex_currentfile_seq)
3647       \\ \\
3648     }
3649     \msg_warning:nn{stex}{warning/deprecated}
3650     \STEXsymbol { #2~#3 }![ \comp{#2~#3s} ]
3651   } {
3652     \msg_set:nnn{stex}{warning/deprecated}{
3653       \\
3654       \c_backslash_str treffiis~is~deprecated! \\
3655       Please~use~\c_backslash_str STEXsymbol { #1 }[ #2~#3s ]~instead!~(in~file~
3656       \stex_path_to_string:N \g_stex_currentfile_seq)
3657       \\ \\
3658     }
3659     \msg_warning:nn{stex}{warning/deprecated}
3660     \STEXsymbol { #1 }![ \comp{#2~#3s} ]
3661   }
3662 }
3663
3664 \NewDocumentCommand \symvariant { 0{} m 0{0} m m } {
3665   \msg_set:nnn{stex}{warning/deprecated}{
3666     \\
3667     \c_backslash_str symvariant~is~deprecated! \\
3668     Please~use~\c_backslash_str notation[#4]{ #2 }~instead!~(in~file~
3669     \stex_path_to_string:N \g_stex_currentfile_seq)
3670     \\ \\
3671   }
3672   \msg_warning:nn{stex}{warning/deprecated}
3673
3674   \notation[variant=#4]{#2}{#5}
3675 }
3676
3677 \NewDocumentCommand \mixfixi { 0{} m m m } {
3678   \msg_set:nnn{stex}{warning/deprecated}{
3679     \c_backslash_str mixfixi~is~fatally~deprecated!\\
3680     Symbol:~\l__stex_term_highlight_uri_str\\
3681     Current~file:~\stex_path_to_string:N \g_stex_currentfile_seq
3682   }
3683   \msg_error:nn{stex}{warning/deprecated}
3684 }
3685
3686
3687 \NewDocumentCommand \infix {} {
3688   \msg_set:nnn{stex}{warning/deprecated}{
3689     \c_backslash_str infix~is~fatally~deprecated!\\
3690     Symbol:~\l__stex_term_highlight_uri_str\\

```

```

3691   Current~file:~\stex_path_to_string:N \g_stex_currentfile_seq
3692 }
3693 \msg_error:nn{stex}{warning/deprecated}
3694 }
3695
3696 \let\iprec\infpres
3697
3698 \NewDocumentCommand \inlineex { m } {
3699   \msg_set:nnn{stex}{warning/deprecated}{
3700     \c_backslash_str inlineex~is~deprecated!\\
3701     No~replacement~exists~yet.\\
3702     Current~file:~\stex_path_to_string:N \g_stex_currentfile_seq
3703   }
3704   \msg_warning:nn{stex}{warning/deprecated}
3705   #1
3706 }
3707
3708
3709 \NewDocumentCommand \term { m } {
3710   \msg_set:nnn{stex}{warning/deprecated}{
3711     \c_backslash_str term~is~deprecated!\\
3712     No~replacement~exists~yet.\\
3713     Current~file:~\stex_path_to_string:N \g_stex_currentfile_seq
3714   }
3715   \msg_warning:nn{stex}{warning/deprecated}
3716   #1
3717 }
3718
3719
3720 \NewDocumentCommand \Definame { O{} m } {
3721   \stex_get_symbol:n { #2 }
3722   \str_set:Nx \l_tmpa_str {
3723     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3724   }
3725   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3726   \scalatex_if:TF {
3727     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3728       \l_tmpa_str
3729     }
3730   } {
3731     \@defemph {
3732       \exp_after:wN \stex_capitalize:n \l_tmpa_str
3733     } { \l_stex_get_symbol_uri_str }
3734   }
3735 }
3736
3737 \NewDocumentCommand \Definiendum { O{} m m } {
3738   \stex_get_symbol:n { #2 }
3739   \str_set:Nx \l_tmpa_str {
3740     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3741   }
3742   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3743   \scalatex_if:TF {
3744     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {

```



```

3745 \l_tmpa_str
3746 }
3747 } {
3748 \defemph {
3749 \exp_after:wN \stex_capitalize:n \l_tmpa_str
3750 } { \l_stex_get_symbol_uri_str }
3751 }
3752 }
3753
3754 \NewDocumentCommand \Symname { 0{ } m }{
3755 \stex_symname_args:n { #1 }
3756 \stex_get_symbol:n { #2 }
3757 \str_set:Nx \l_tmpa_str {
3758 \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3759 }
3760 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3761 \exp_args:NNx \use:nn
3762 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }![
3763 \exp_after:wN \stex_capitalize:n \l_tmpa_str
3764 \l_stex_symname_post_str
3765 ] }
3766 }
3767
3768
3769 \seq_gput_right:Nx \g_stex_smsmode_allowedenvs_seq { \tl_to_str:n { mhmodnl } }
3770 \seq_gput_right:Nx \g_stex_smsmode_allowedenvs_seq { \tl_to_str:n { modsig } }
3771 \tl_gput_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\gimport\syml\symii\symiii\symiv\
3772
3773 % omtex:
3774 \cs_new_protected:Npn \lec #1 {
3775 \strut\hfil\strut\hfill(#1)
3776 }
3777 \cs_new_protected:Npn \nlex #1 {
3778 \textcolor{green}{\s1 #1}}
3779 }
3780
3781
3782 </compat>

```