

stex.sty: \TeX 2.0*

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

2021-11-23

Abstract

TODO

1 Introduction

TODO

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

Contents

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	79
4.11	Put these somewhere	87
4.12	Metatheory	87
4.13	Auxiliary Packages	89

2 Manual

2.1 Modules

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

2.2 Semantic Macros and Notations

Semantic macros invoke a formally declared symbol.

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

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

Example 1

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

(ab)

.

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

```
\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

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

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

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

For a macro with arity > 0 , we can refer to the operator *itself* semantically by suffixing the semantic macro with an exclamation point `!` in either text or math mode. 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 $\text{\texttt{\backslash begin{module}\{Foo\}}}$ occurs in a file $\text{\texttt{/path/to/file/Foo[.<lang>].tex}}$ which does not belong to an archive, the namespace is $\text{\texttt{file://path/to/file}}$.
- If the same statement occurs in a file $\text{\texttt{/path/to/file/bar[.<lang>].tex}}$, the namespace is $\text{\texttt{file://path/to/file/bar}}$.

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 $\text{\texttt{\backslash importmodule}}$:

- $\text{\texttt{\backslash importmodule}\{Foo\}}$ 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 $\text{\texttt{Foo[.<lang>].tex}}$ 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>	LaTeX2e and LaTeX3 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`

 $\backslash\text{stex_notation_do:nn}$
 $\backslash\text{stex_notation_do:nn}\langle\text{URI}\rangle\{\langle\text{notations}^+\rangle\}$

Implements the core functionality of $\backslash\text{notation}$, and is called by $\backslash\text{notation}$ and $\backslash\text{symdef}$.

Ultimately stores the notation in the property list $\backslash\text{g_stex_notation_}\langle\text{URI}\rangle\#\langle\text{variant}\rangle\#\langle\text{lang}\rangle_\text{prop}$ 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]

 $\backslash\text{symdef}$
 $\backslash\text{symdef}[\langle\text{args}\rangle]\{\langle\text{symbol}\rangle\}\{\langle\text{notations}^+\rangle\}$

Combines $\backslash\text{symdecl}$ and $\backslash\text{notation}$ 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]
($a+b+c$)

 $\backslash\text{stex_term_math_oms:nnnn}$
 $\backslash\text{stex_term_math_oma:nnnn}$
 $\backslash\text{stex_term_math_omb:nnnn}$
 $\langle\text{URI}\rangle\langle\text{fragment}\rangle\langle\text{precedence}\rangle\langle\text{body}\rangle$

Annotates $\langle\text{body}\rangle$ as an OMDOC-term (OMID, OMA or OMBIND, respectively) with head symbol $\langle\text{URI}\rangle$, generated by the specific notation $\langle\text{fragment}\rangle$ with (upwards) operator precedence $\langle\text{precedence}\rangle$. 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 \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 \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}

```

```

Module 3.16[MathTest2]
  ( $\langle a|b;c,d,e,f \rangle^g$ ) and ( $\langle a|b;c \rangle^g$ ) and ( $\langle a|b \rangle^c$ )
  ( $a+(b\cdot c)$ ) and ( $a\cdot\frac{a}{b}+\frac{a}{c}$ )

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

  ( $a+(b\cdot c)$ ) and [ $a\cdot\frac{a}{b}+\frac{a}{c}$ ]

```

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

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

Test 16

```

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

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

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

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

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

\bar![bar]

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

\end{module}

```

```

Module 3.17[TextTest]
  some a and some b and also some c here.
  some a and some b and also some c here.

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

```

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

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

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

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

`\@defemph`

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`)

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

<code>\ellipses</code>	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 ! {\#1 \comp+ #2 }
]{mM}
\notation[op = U]{mM/universe}{\comp U}
\notation[op = +]{mM/op}{#1 \comp+ #2}

Test: $\mM{op}ab$

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

```

Module 3.18[StructureTest1]

(aob:(M))

file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?StructureTest1/Magma-feature?universe,file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?StructureTest1/Magma-feature?op

macro->stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?StructureTest1?Magma}

Test: (a+b)

Test2: ((U,+))

4 Implementation

4.1 The `sTeX` document class

```

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

```

```

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

```

4.2 Preliminaries

```

13 <*package>
14 \RequirePackage{expl3,l3keys2e}
15 \ProvidesExplPackage{stex}{2021/08/01}{1.9}{bla}

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

```

\sTeX The sTeX logo:

```

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

```

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

Messages

```

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

```

\stex_debug:n Debug mode

```

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

```

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

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

(End definition for `\c__stex_sms_iow`.)

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

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

4.2.1 L^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`

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_lat_ex).

The p_df_lat_ex-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     \NewDocumentEnvironment{stex_annotate_env} { m m } {
162       \par\begin{latexml@annotateenv}{#1}{#2}
163     }{
164       \end{latexml@annotateenv}
165     }
166   }{
167     \cs_new_protected:Nn \stex_annotate:nnn {#3}
168     \cs_new_protected:Nn \stex_annotate_invisible:n {}
169     \cs_new_protected:Nn \stex_annotate_invisible:nnn {}

```

```

170 \NewDocumentEnvironment{stex_annotate_env} { m m } {\par}{\}
171 }
172 }

```

(End definition for `\stex_annotate:nnn`, `\stex_annotate_invisible:n`, and `\stex_annotate_invisible:nnn`. These functions are documented on page 9.)

4.2.3 Languages

```

173 <@@=stex_language>

```

`\c_stex_languages_prop`

`\c_stex_language_abbrevs_prop`

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

```

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

```

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

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

```

199 \clist_if_empty:NF \c_stex_languages_clist {
200   \clist_clear:N \l_tmpa_clist
201   \clist_map_inline:Nn \c_stex_languages_clist {
202     \prop_get:NnNTF \c_stex_languages_prop { #1 } \l_tmpa_str {
203       \clist_put_right:No \l_tmpa_clist \l_tmpa_str
204     } {
205       \msg_set:nnn{stex}{error/unknownlanguage}{
206         Unknown~language~\l_tmpa_str
207       }
208       \msg_error:nn{stex}{error/unknownlanguage}
209     }
210   }
211   \stex_debug:n {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
212   \RequirePackage[\clist_use:Nn \l_tmpa_clist ,]{babel}
213 }

```

4.3 Files, Paths and URIs

214 `<@@=stex_path>`

4.3.1 Generic Path Handling

We treat paths as L^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 . and .., respectively.
\c__stex_path_up_str
241 \str_const:Nn \c__stex_path_dot_str {.}
242 \str_const:Nn \c__stex_path_up_str {...}

```

(End definition for `\c__stex_path_dot_str` and `\c__stex_path_up_str`.)

`\stex_path_canonicalize:N` Canonicalizes the path provided; in particular, resolves . and .. path segments.

```

243 \cs_new_protected:Nn \stex_path_canonicalize:N {
244   \seq_if_empty:NF #1 {
245     \seq_clear:N \l_tmpa_seq
246     \seq_get_left:NN #1 \l_tmpa_tl
247     \str_if_empty:NT \l_tmpa_tl {

```

```

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

```

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

`\stex_path_if_absolute_p:N`
`\stex_path_if_absolute:NTF`

```

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

```

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

4.3.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

290 \str_new:N\l_stex_kpsewhich_return_str
291 \cs_new_protected:Nn \stex_kpsewhich:n {

```



```

292 \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
293 \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
294 \tl_trim_spaces:N \l_stex_kpsewhich_return_str
295 }

```

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

We determine the PWD

`\c_stex_pwd_seq`
`\c_stex_pwd_str`

```

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

```

(End definition for `\c_stex_pwd_seq` and `\c_stex_pwd_str`. These variables are documented on page 10.)

4.3.3 File Hooks and Tracking

```

305 <@@=stex_files>

```

We introduce hooks for file inputs that keep track of the absolute paths of files used. This will be useful to keep track of modules, their archives, namespaces etc.

Note that the absolute paths are only accurate in `\input`-statements for paths relative to the PWD, so they shouldn't be relied upon in any other setting than for \TeX -purposes.

`\g__stex_files_stack` keeps track of file changes

```

306 \seq_gclear_new:N\g__stex_files_stack

```

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`

```

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

```

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

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

```

310 \seq_gclear_new:N\g_stex_currentfile_seq
311 \AddToHook{file/before}{
312   \stex_path_from_string:Nn\g_stex_currentfile_seq{\CurrentFilePath}
313   \stex_path_if_absolute:NTF\g_stex_currentfile_seq{
314     \exp_args:NNe\seq_put_right:Nn\g_stex_currentfile_seq{\CurrentFile}
315   }{
316     \stex_path_from_string:Nn\g_stex_currentfile_seq{
317       \c_stex_pwd_str/\CurrentFilePath/\CurrentFile
318     }
319   }

```

```

320 \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
321 \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
322 }
323 \AddToHook{file/after}{
324   \seq_if_empty:NF\g__stex_files_stack{
325     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
326   }
327   \seq_if_empty:NTF\g__stex_files_stack{
328     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
329   }{
330     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
331     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
332   }
333 }

```

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

4.4 MathHub Repositories

```

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

```

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

```

\__stex_mathhub_do_manifest:n
355 \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
356   \str_set:Nx \l_tmpa_str { #1 }
357   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
358     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
359     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
360     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
361     \__stex_mathhub_find_manifest:N \l_tmpa_seq
362     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {

```

```

363     \msg_set:nnn{stex}{error/norepository}{
364       No~archive~#1~found~in~
365       \stex_path_to_string:N \c_stex_mathhub_str
366     }
367     \msg_error:nn{stex}{error/norepository}
368   } {
369     \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
370   }
371 }
372 }

```

(End definition for __stex_mathhub_do_manifest:n.)

\l_stex_mathhub_manifest_file_seq

```

373 \str_new:N\l__stex_mathhub_manifest_file_seq

```

(End definition for \l_stex_mathhub_manifest_file_seq.)

__stex_mathhub_find_manifest:N Attempts to find the MANIFEST.MF in some file path and stores its path in \l__stex_mathhub_manifest_file_seq:

```

374 \cs_new_protected:Nn \__stex_mathhub_find_manifest:N {
375   \seq_set_eq:NN\l_tmpa_seq #1
376   \bool_set_true:N\l_tmpa_bool
377   \bool_while_do:Nn \l_tmpa_bool {
378     \seq_if_empty:NTF \l_tmpa_seq {
379       \bool_set_false:N\l_tmpa_bool
380     }{
381       \file_if_exist:nTF{
382         \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
383       }{
384         \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
385         \bool_set_false:N\l_tmpa_bool
386       }{
387         \file_if_exist:nTF{
388           \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
389         }{
390           \seq_put_right:Nn\l_tmpa_seq{META-INF}
391           \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
392           \bool_set_false:N\l_tmpa_bool
393         }{
394           \file_if_exist:nTF{
395             \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
396           }{
397             \seq_put_right:Nn\l_tmpa_seq{meta-inf}
398             \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
399             \bool_set_false:N\l_tmpa_bool
400           }{
401             \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
402           }
403         }
404       }
405     }
406   }
407   \seq_set_eq:NN\l__stex_mathhub_manifest_file_seq\l_tmpa_seq
408 }

```

(End definition for `_stex_mathhub_find_manifest:N`.)

`\c_stex_mathhub_manifest_ior` File variable used for MANIFEST-files

409 `\ior_new:N \c__stex_mathhub_manifest_ior`

(End definition for `\c_stex_mathhub_manifest_ior`.)

`_stex_mathhub_parse_manifest:n` Stores the entries in manifest file in the corresponding property list:

```

410 \cs_new_protected:Nn \_stex_mathhub_parse_manifest:n {
411   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
412   \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
413   \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
414     \str_set:Nn \l_tmpa_str {#1}
415     \exp_args:NNoo \seq_set_split:Nnn
416       \l_tmpb_seq \c_colon_str \l_tmpa_str
417     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
418       \exp_args:NNe \str_set:Nn \l_tmpb_tl {
419         \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
420       }
421       \exp_args:No \str_case:nnTF \l_tmpa_tl {
422         {id} {
423           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
424             { id } \l_tmpb_tl
425         }
426         {narration-base} {
427           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
428             { narr } \l_tmpb_tl
429         }
430         {source-base} {
431           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
432             { ns } \l_tmpb_tl
433         }
434         {ns} {
435           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
436             { ns } \l_tmpb_tl
437         }
438         {dependencies} {
439           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
440             { deps } \l_tmpb_tl
441         }
442       }{}{}
443     }{}
444   }
445   \ior_close:N \c__stex_mathhub_manifest_ior
446 }

```

(End definition for `_stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

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

```

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

`\stex_require_repository:n`

```

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

```

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

`\l_stex_current_repository_prop`

Current MathHub repository

```

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

```

(End definition for `\l_stex_current_repository_prop`. This variable is documented on page 11.)

`\inputref`

```

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

```

```

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

```

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

`\libinput`

```

508 \cs_new_protected:Npn \libinput #1 {
509   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
510     \msg_set:nnn{stex}{error/norepository}{
511       \c_backslash_str libinput~needs~to~be~called~in~an~archive
512     }
513     \msg_error:nn{stex}{error/norepository}
514   }
515   \bool_set_false:N \l_tmpa_bool
516   \tl_clear:N \l_tmpa_tl
517   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
518   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
519   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str
520   \seq_pop_left:NNT \l_tmpb_seq \l_tmpb_str {
521     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
522     \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
523       / meta-inf / lib / #1.tex}{
524       \bool_set_true:N \l_tmpa_bool
525       \tl_put_right:Nx \l_tmpa_tl {
526         \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
527           / meta-inf / lib / #1.tex}
528       }
529     }{}
530   }
531   \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
532     / \l_tmpa_str / lib / #1.tex
533   }{
534     \bool_set_true:N \l_tmpa_bool
535     \tl_put_right:Nx \l_tmpa_tl {
536       \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
537         / \l_tmpa_str / lib / #1.tex}
538     }
539   }{}
540   \bool_if:NF \l_tmpa_bool {
541     \msg_set:nnn{stex}{error/nofile}{
542       \c_backslash_str libinput~no~file~#1.tex~found!
543     }
544     \msg_error:nn{stex}{error/nofile}
545   }

```

```

546 \l_tmpa_tl
547 }

```

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

4.5 Module System

```

548 <@@=stex_module>

```

\l_stex_current_module_prop

```

549 \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

```

550 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
551   \prop_if_empty:NTF \l_stex_current_module_prop
552   \prg_return_false: \prg_return_true:
553 }

```

(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

```

554 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
555   \prop_if_exist:cTF { c_stex_module_#1_prop }
556   \prg_return_true: \prg_return_false:
557 }

```

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

\stex_add_to_current_module:n

\STEXexport

```

558 \cs_new_protected:Nn \stex_add_to_current_module:n {
559   \prop_get:NnN \l_stex_current_module_prop { content } \l_tmpa_tl
560   \tl_put_right:Nn \l_tmpa_tl { #1 }
561   \prop_put:Nno \l_stex_current_module_prop { content } { \l_tmpa_tl }
562 }
563 \NewDocumentCommand \STEXexport { m }{
564   \stex_smsmode_set_codes:
565   \stex_add_to_current_module:n { #1 }
566   #1
567 }

```

(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

```

568 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
569   \str_set:Nx \l_tmpa_str { #1 }
570   \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
571   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
572   \prop_put:Nno \l_stex_current_module_prop { constants } \l_tmpa_seq
573 }

```

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

`\stex_add_import_to_current_module:n`

```

574 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
575   \str_set:Nx \l_tmpa_str { #1 }
576   \prop_get:NnN \l_stex_current_module_prop { imports } \l_tmpa_seq
577   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
578   \prop_put:Nno \l_stex_current_module_prop { imports } \l_tmpa_seq
579 }

```

(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 580 \str_new:N \l_stex_modules_ns_str

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

```

(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:`

```

610 \cs_new_protected:Nn \stex_modules_current_namespace: {
611   \prop_get:NnNTF \l_stex_current_repository_prop { ns } \l_tmpa_str {
612     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
613   }{
614     % split off file extension
615     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq

```



```

616 \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
617 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
618 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
619 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
620 \str_set:Nx \l_stex_modules_ns_str {
621   file:/\stex_path_to_string:N \l_tmpa_seq
622 }
623 }
624 }

```

(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

```

625 \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`

```

626 \NewDocumentCommand \STEXModule { m } {
627   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
628   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
629   \tl_set:Nn \l_tmpa_tl {
630     \msg_set:nnn{stex}{error/unknownmodule}{
631       No~module~#1~found!
632     }
633     \msg_error:nn{stex}{error/unknownmodule}
634   }
635   \seq_map_inline:Nn \l_stex_all_modules_seq {
636     \str_set:Nn \l_tmpb_str { ##1 }
637     \str_if_eq:eeT { \l_tmpa_str } {
638       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
639     } {
640       \seq_map_break:n {
641         \tl_set:Nn \l_tmpa_tl {
642           \stex_invoke_module:n { ##1 }
643         }
644       }
645     }
646   }
647   \l_tmpa_tl
648 }

649
650 \cs_new_protected:Nn \stex_invoke_module:n {
651   \stex_debug:n{Invoking~module~#1}
652   \peek_charcode_remove:NTF ! {
653     \__stex_module_invoke_uri:nN { #1 }
654   } {
655     \peek_charcode_remove:NTF ? {
656       \__stex_module_invoke_symbol:nn { #1 }
657     } {
658       \msg_set:nnn{stex}{error/syntax}{
659         Syntax~error:~?~or~!~expected~after~

```

```

660         \c_backslash_str STEXModule{#1}
661     }
662     \msg_error:nn{stex}{error/syntax}
663 }
664 }
665 }
666
667 \cs_new_protected:Nn \__stex_module_invoke_uri:nN {
668     \str_set:Nn #2 { #1 }
669 }
670
671 \cs_new_protected:Nn \__stex_module_invoke_symbol:nn {
672     \stex_invoke_symbol:n{#1?#2}
673 }

```

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

module module arguments:

```

674 \keys_define:nn { stex / module } {
675     title          .tl_set_x:N = \l_stex_module_title_str ,
676     ns             .tl_set_x:N = \l_stex_module_ns_str ,
677     lang           .tl_set_x:N = \l_stex_module_lang_str ,
678     sig            .tl_set_x:N = \l_stex_module_sig_str ,
679     creators       .tl_set_x:N = \l_stex_module_creators_str ,
680     contributors   .tl_set_x:N = \l_stex_module_contributors_str ,
681     meta           .tl_set_x:N = \l_stex_module_meta_str
682 }
683
684 % module parameters here? In the body?
685
686 \cs_new_protected:Nn \__stex_module_args:n {
687     \str_clear:N \l_stex_module_title_str
688     \str_clear:N \l_stex_module_ns_str
689     \str_clear:N \l_stex_module_lang_str
690     \str_clear:N \l_stex_module_sig_str
691     \str_clear:N \l_stex_module_creators_str
692     \str_clear:N \l_stex_module_contributors_str
693     \str_clear:N \l_stex_module_meta_str
694     \keys_set:nn { stex / module } { #1 }
695     \exp_args:NNo \str_set:Nn \l_stex_module_title_str
696         \l_stex_module_title_str
697     \exp_args:NNo \str_set:Nn \l_stex_module_ns_str
698         \l_stex_module_ns_str
699     \exp_args:NNo \str_set:Nn \l_stex_module_lang_str
700         \l_stex_module_lang_str
701     \exp_args:NNo \str_set:Nn \l_stex_module_sig_str
702         \l_stex_module_sig_str
703     \exp_args:NNo \str_set:Nn \l_stex_module_meta_str
704         \l_stex_module_meta_str
705     \exp_args:NNo \str_set:Nn \l_stex_module_creators_str
706         \l_stex_module_creators_str
707     \exp_args:NNo \str_set:Nn \l_stex_module_contributors_str
708         \l_stex_module_contributors_str
709 }

```

```

\__stex_module_begin_module: implements \begin{module}

710 \cs_new_protected:Nn \__stex_module_begin_module: {
711   % Nested module?
712   \stex_if_in_module:TF {
713     % Nested module
714     \prop_get:NnN \l_stex_current_module_prop
715       { ns } \l_stex_module_ns_str
716     \str_set:Nx \l_stex_module_name_str {
717       \prop_item:Nn \l_stex_current_module_prop
718         { name } / \l_stex_module_name_str
719     }
720   }{
721     % not nested:
722     \str_if_empty:NT \l_stex_module_ns_str {
723       \stex_modules_current_namespace:
724       \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
725       \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
726         / {\l_stex_module_ns_str}
727       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
728       \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
729         \str_set:Nx \l_stex_module_ns_str {
730           \stex_path_to_string:N \l_tmpa_seq
731         }
732       }
733     }
734   }
735
736   % language
737   \str_if_empty:NT \l_stex_module_lang_str {
738     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
739     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
740     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
741     \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
742     \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
743       \stex_debug:n {Language~\l_stex_module_lang_str~
744         inferred~from~file~name}
745       \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
746     }
747   }
748
749   \str_if_empty:NF \l_stex_module_lang_str {
750     \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
751       \l_tmpa_str {
752       \ltx@ifpackageloaded{babel}{
753         \exp_args:Nx \selectlanguage { \l_tmpa_str }
754       }{}
755     } {
756       \msg_set:nnn{stex}{error/unknownlanguage}{
757         Unknown~language~\l_tmpa_str
758       }
759       \msg_error:nn{stex}{error/unknownlanguage}
760     }
761   }
762

```

```

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

```

```

817     \l_stex_module_name_str
818     _prop
819   }
820 }
821
822 % metatheory
823 \str_if_empty:NT \l_stex_module_meta_str {
824   \str_set:Nx \l_stex_module_meta_str {
825     \c_stex_metatheory_ns_str ? Metatheory
826   }
827 }
828
829
830 \stex_debug:n{
831   New~module:\\
832   Namespace:~\l_stex_module_ns_str\\
833   Name:~\l_stex_module_name_str\\
834   Language:~\l_stex_module_lang_str\\
835   Signature:~\l_stex_module_sig_str\\
836   Metatheory:~\l_stex_module_meta_str\\
837   File:~\stex_path_to_string:N \g_stex_currentfile_seq
838 }
839
840 \seq_put_right:Nx \l_stex_all_modules_seq {
841   \l_stex_module_ns_str ? \l_stex_module_name_str
842 }
843
844 \seq_gput_right:Nx \g_stex_modules_in_file_seq
845   { \l_stex_module_ns_str ? \l_stex_module_name_str }
846
847 \stex_if_smsmode:TF {
848   \stex_smsmode_set_codes:
849 } {
850   \begin{stex_annotate_env} {theory} {
851     \l_stex_module_ns_str ? \l_stex_module_name_str
852   }
853
854   \stex_annotate_invisible:nnn{header}{} {
855     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
856     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
857     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
858       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
859     }
860   }
861 }
862
863 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
864   \exp_args:Nx \STEXexport{
865     \stex_activate_module:n { \l_stex_module_meta_str }
866   }
867 }
868 % TODO: Inherit metatheory for nested modules?
869 }
870 \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}

```

871 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
872 \cs_new_protected:Nn \_stex_module_end_module: {
873   \str_set:Nx \l_tmpa_str {
874     c_stex_module_
875     \prop_item:Nn \l_stex_current_module_prop { ns } ?
876     \prop_item:Nn \l_stex_current_module_prop { name }
877     _prop
878   }
879   %^^A \prop_new:c { \l_tmpa_str }
880   \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
881   \stex_debug:n{Closing-module~\prop_item:Nn \l_stex_current_module_prop { name }}
882   \stex_if_smsmode:TF {
883     \exp_args:Nx \stex_addtosms:n {
884       \prop_gset_from_keyval:cn {
885         c_stex_module_
886         \prop_item:Nn \l_stex_current_module_prop { ns } ?
887         \prop_item:Nn \l_stex_current_module_prop { name }
888         _prop
889       } {
890         name      = \prop_item:cn { \l_tmpa_str } { name } ,
891         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
892         imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
893         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
894         content   = \prop_item:cn { \l_tmpa_str } { content } ,
895         file      = \prop_item:cn { \l_tmpa_str } { file } ,
896         lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
897         sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
898         meta      = \prop_item:cn { \l_tmpa_str } { meta }
899       }
900     }
901   }{
902     \end{stex_annotate_env}
903   }
904 }

```

(End definition for _stex_module_end_module:.)

@module The core environment, with no header

```

905 \NewDocumentEnvironment { @module } { 0{} m } {
906   \str_set:Nx \l_stex_module_name_str { #2 }
907   \par
908   \_stex_module_args:n { #1 }
909   \_stex_module_begin_module:
910 } {
911   \_stex_module_end_module:
912 }

```

\stex_modules_heading: Code for document headers

```

913 \cs_if_exist:NTF \thesection {
914   \newcounter{module}[section]

```

```

915 }{
916   \newcounter{module}
917 }
918
919 \bool_if:NT \c_stex_showmods_bool {
920   \latexml_if:F { \RequirePackage{mdframed} }
921 }
922
923 \cs_new_protected:Nn \stex_modules_heading: {
924   \stepcounter{module}
925   \par
926   \bool_if:NT \c_stex_showmods_bool {
927     \noindent{\textbf{Module} ~
928       \cs_if_exist:NT \thesection {\thesection.}
929       \themodule ~ [\l_stex_module_name_str]
930     }
931     % TODO references
932     % \sref@label@id{Module \thesection.\themodule [\module@name]]}%
933     \str_if_empty:NTF \l_stex_module_title_str {
934       }{
935         \quad(\l_stex_module_title_str)\hfill
936       }\par
937     }
938   }

```

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

Finally:

```

939 \NewDocumentEnvironment { module } { 0{} m } {
940   \bool_if:NT \c_stex_showmods_bool {
941     \begin{mdframed}
942   }
943   \begin{@module}[\#1]{\#2}
944   \stex_modules_heading:
945   }{
946     \end{@module}
947     \bool_if:NT \c_stex_showmods_bool {
948       \end{mdframed}
949     }
950   }

```

4.5.2 SMS Mode

```

951 <@@=stex_smsmode>

```

```

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq
952 \tl_new:N \g_stex_smsmode_allowedmacros_tl
953 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
954 \seq_new:N \g_stex_smsmode_allowedenvs_seq
955
956 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
957   \makeatletter
958   \makeatother
959   \ExplSyntaxOn
960   \ExplSyntaxOff

```

```

961 }
962
963 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
964   \symdef
965   \importmodule
966   \notation
967   \symdecl
968   \STEXexport
969 }
970
971 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
972   \tl_to_str:n {
973     module,
974     @module
975   }
976 }

```

(End definition for `\g_stex_smsmode_allowedmacros_tl`, `\g_stex_smsmode_allowedmacros_escape_tl`, and `\g_stex_smsmode_allowedenvs_seq`. These variables are documented on page 15.)

```

\stex_if_smsmode_p:
\stex_if_smsmode:TF
977 \bool_new:N \g__stex_smsmode_bool
978 \bool_set_false:N \g__stex_smsmode_bool
979 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
980   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
981 }

```

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

```

\__stex_smsmode_if_catcodes_p: Checks whether the SMS mode category code scheme is active.
\__stex_smsmode_if_catcodes:TF
982 \bool_new:N \g__stex_smsmode_catcode_bool
983 \bool_set_false:N \g__stex_smsmode_catcode_bool
984 \prg_new_conditional:Nnn \__stex_smsmode_if_catcodes: { p, T, F, TF } {
985   \bool_if:NTF \g__stex_smsmode_catcode_bool
986     \prg_return_true: \prg_return_false:
987 }

```

(End definition for `__stex_smsmode_if_catcodes:TF`.)

```

\stex_smsmode_set_codes:
988 \cs_new_protected:Nn \stex_smsmode_set_codes: {
989   \stex_if_smsmode:T {
990     \__stex_smsmode_if_catcodes:F {
991       \bool_gset_true:N \g__stex_smsmode_catcode_bool
992       \exp_after:wN \char_gset_active_eq:NN
993       \c_backslash_str \__stex_smsmode_cs:
994       \tex_global:D \char_set_catcode_active:N \
995       \tex_global:D \char_set_catcode_other:N $
996       \tex_global:D \char_set_catcode_other:N ^
997       \tex_global:D \char_set_catcode_other:N _
998       \tex_global:D \char_set_catcode_other:N &
999       \tex_global:D \char_set_catcode_other:N ##
1000     }
1001   }
1002 } \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.

```

1003 \cs_new_protected:Nn \__stex_smsmode_unset_codes: {
1004   \__stex_smsmode_if_catcodes:T {
1005     \bool_gset_false:N \g__stex_smsmode_catcode_bool
1006     \exp_after:wN \tex_global:D \exp_after:wN
1007     \char_set_catcode_escape:N \c_backslash_str
1008     \tex_global:D \char_set_catcode_math_toggle:N $
1009     \tex_global:D \char_set_catcode_math_superscript:N ^
1010     \tex_global:D \char_set_catcode_math_subscript:N _
1011     \tex_global:D \char_set_catcode_alignment:N &
1012     \tex_global:D \char_set_catcode_parameter:N ##
1013   }
1014 } \iffalse $ \fi % to make syntax highlighting work again

```

(End definition for `__stex_smsmode_unset_codes:`.)

`\stex_in_smsmode:nn`

```

1015 \cs_new_protected:Nn \stex_in_smsmode:nn {
1016   \vbox_set:Nn \l_tmpa_box {
1017     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1018     \bool_gset_true:N \g__stex_smsmode_bool
1019     \stex_smsmode_set_codes:
1020     #2
1021     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1022     \stex_if_smsmode:F {
1023       \__stex_smsmode_unset_codes:
1024     }
1025   }
1026   \box_clear:N \l_tmpa_box
1027 }

```

(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:

```

1028 \cs_new_protected:Nn \__stex_smsmode_cs: {
1029   \str_clear:N \l_tmpa_str
1030   \peek_analysis_map_inline:n {
1031     % #1: token (one expansion)
1032     % #2: charcode
1033     % #3 catcode
1034     \token_if_eq_charcode:NNTF ##3 B {
1035       % token is a letter
1036       \exp_args:NNo \str_put_right:Nn \l_tmpa_str { ##1 }
1037     } {
1038       \str_if_empty:NTF \l_tmpa_str {
1039         % we don't allow (or need) single non-letter CSs
1040         % for now
1041         \peek_analysis_map_break:
1042       }{
1043         \str_if_eq:onTF \l_tmpa_str { begin } {
1044           \peek_analysis_map_break:n {

```

```

1045         \exp_after:wN \__stex_smsmode_checkbegin:n ##1
1046     }
1047 } {
1048     \str_if_eq:onTF \l_tmpa_str { end } {
1049         \peek_analysis_map_break:n {
1050             \exp_after:wN \__stex_smsmode_checkend:n ##1
1051         }
1052     } {
1053         \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
1054         \exp_args:NNNo \exp_args:NNo \tl_if_in:NnTF
1055         \g_stex_smsmode_allowedmacros_tl
1056         { \use:c{\l_tmpa_str} } {
1057             \stex_debug:n{Executing~1:~\l_tmpa_str}
1058             \peek_analysis_map_break:n {
1059                 \exp_after:wN \l_tmpa_tl ##1
1060             }
1061         } {
1062             \exp_args:NNNo \exp_args:NNo \tl_if_in:NnTF
1063             \g_stex_smsmode_allowedmacros_escape_tl
1064             { \use:c{\l_tmpa_str} } {
1065                 \stex_debug:n{Executing~2:~\l_tmpa_str}
1066                 % TODO \__stex_smsmode_rescan_cs:
1067                 \exp_after:wN \exp_after:wN \exp_after:wN
1068                 \token_if_eq_charcode:NNTF \exp_after:wN \c_backslash_str ##1 {
1069                     \peek_analysis_map_break:n {
1070                         \__stex_smsmode_unset_codes:
1071                         \__stex_smsmode_rescan_cs:
1072                     }
1073                 } {
1074                     \peek_analysis_map_break:n {
1075                         \__stex_smsmode_unset_codes:
1076                         \exp_after:wN \l_tmpa_tl ##1
1077                     }
1078                 }
1079             } {
1080                 \peek_analysis_map_break:n { ##1 }
1081             }
1082         }
1083     }
1084 }
1085 }
1086 }
1087 }
1088 }

```

(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:

```

1089 \cs_new_protected:Nn \__stex_smsmode_rescan_cs: {
1090     \str_clear:N \l_tmppb_str
1091     \peek_analysis_map_inline:n {
1092         \token_if_eq_charcode:NNTF ##3 B {
1093             % token is a letter

```

```

1094     \exp_args:NNo \str_put_right:Nn \l_tmpb_str { ##1 }
1095   } {
1096     \peek_analysis_map_break:n {
1097       \exp_after:wN \use:c \exp_after:wN {
1098         \exp_after:wN \l_tmpa_str\exp_after:wN
1099       } \use:c { \l_tmpb_str \exp_after:wN } ##1
1100     }
1101   }
1102 }
1103 }

```

(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.

```

1104 \cs_new_protected:Nn \__stex_smsmode_checkbegin:n {
1105   \str_set:Nn \l_tmpa_str { #1 }
1106   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1107     \__stex_smsmode_unset_codes:
1108     \begin{#1}
1109   }
1110 }

```

(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.

```

1111 \cs_new_protected:Nn \__stex_smsmode_checkend:n {
1112   \str_set:Nn \l_tmpa_str { #1 }
1113   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1114     \end{#1}
1115   }
1116 }

```

(End definition for `__stex_smsmode_checkend:n`.)

4.5.3 Inheritance

```

1117 <@@=stex_importmodule>

```

`\stex_import_module_uri:nn`

```

1118 \cs_new_protected:Nn \stex_import_module_uri:nn {
1119   \str_set:Nx \l__stex_importmodule_archive_str { #1 }
1120   \str_set:Nx \l__stex_importmodule_path_str { #2 }
1121   \str_if_empty:NT \l__stex_importmodule_archive_str {
1122     \prop_if_empty:NF \l_stex_current_repository_prop {
1123       \prop_get:NnN \l_stex_current_repository_prop { id } \l__stex_importmodule_archive_str
1124     }
1125   }
1126
1127   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l__stex_importmodule_path_str }
1128   \seq_pop_right:NN \l_tmpb_seq \l__stex_importmodule_name_str
1129   \str_set:Nx \l__stex_importmodule_path_str { \seq_use:Nn \l_tmpb_seq ? }
1130
1131   \str_if_empty:NTF \l__stex_importmodule_archive_str {
1132     \stex_modules_current_namespace:
1133     \str_if_empty:NF \l__stex_importmodule_path_str {

```

```

1134     \str_set:Nx \l_stex_module_ns_str {
1135       \l_stex_module_ns_str / \l__stex_importmodule_path_str
1136     }
1137   }
1138 }{
1139   \stex_require_repository:n \l__stex_importmodule_archive_str
1140   \prop_get:cnN { c_stex_mathhub_\l__stex_importmodule_archive_str _manifest_prop } { ns }
1141   \l_stex_module_ns_str
1142   \str_if_empty:NF \l__stex_importmodule_path_str {
1143     \str_set:Nx \l_stex_module_ns_str {
1144       \l_stex_module_ns_str / \l__stex_importmodule_path_str
1145     }
1146   }
1147 }
1148 }

```

(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`.

```

1149 \str_new:N \l__stex_importmodule_name_str
1150 \str_new:N \l__stex_importmodule_archive_str
1151 \str_new:N \l__stex_importmodule_path_str
1152 \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>}
1153 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1154   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1155     % \stex_debug:n{Arguments: #1, #2, #3, #4}
1156
1157     % archive
1158     \str_set:Nx \l_tmpa_str { #2 }
1159     \str_if_empty:NTF \l_tmpa_str {
1160       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1161     } {
1162       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1163       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1164       \seq_put_right:Nn \l_tmpa_seq { source }
1165     }
1166
1167     % path
1168     \str_set:Nx \l_tmpb_str { #3 }
1169     \str_if_empty:NTF \l_tmpb_str {
1170       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1171     }
1172     \ltx@ifpackageloaded{babel} {
1173       \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1174         { \language } \l_tmpb_str {
1175         \msg_set:nnn{stex}{error/unknownlanguage}{
1176           Unknown-language-\language
1177         }
1178         \msg_error:nn{stex}{error/unknownlanguage}
1179       }

```

```

1180 } {
1181   \str_clear:N \l_tmpb_str
1182 }
1183
1184 \stex_debug:n{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1185 \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1186   \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1187 }{
1188   \stex_debug:n{Checking~\l_tmpa_str.tex}
1189   \IfFileExists{ \l_tmpa_str.tex }{
1190     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1191   }{
1192     % try english as default
1193     \stex_debug:n{Checking~\l_tmpa_str.en.tex}
1194     \IfFileExists{ \l_tmpa_str.en.tex }{
1195       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1196     }{
1197       \msg_set:nnn{stex}{error/modulemissing}{
1198         No~file~for~module~#1?#4~found
1199       }
1200       \msg_error:nn{stex}{error/modulemissing}
1201     }
1202   }
1203 }
1204
1205 } {
1206   \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1207   \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1208
1209   \ltx@ifpackageloaded{babel} {
1210     \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1211       { \language } \l_tmpb_str {
1212         \msg_set:nnn{stex}{error/unknownlanguage}{
1213           Unknown~language~\language
1214         }
1215         \msg_error:nn{stex}{error/unknownlanguage}
1216       }
1217   } {
1218     \str_clear:N \l_tmpb_str
1219   }
1220
1221   \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1222
1223   \stex_debug:n{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1224   \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1225     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1226   }{
1227     \stex_debug:n{Checking~\l_tmpa_str/#4.tex}
1228     \IfFileExists{ \l_tmpa_str/#4.tex }{
1229       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1230     }{
1231       % try english as default
1232       \stex_debug:n{Checking~\l_tmpa_str/#4.en.tex}
1233       \IfFileExists{ \l_tmpa_str/#4.en.tex }{

```

```

1234         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1235     }{
1236         \stex_debug:n{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1237         \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1238             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1239         }{
1240             \stex_debug:n{Checking~\l_tmpa_str.tex}
1241             \IfFileExists{ \l_tmpa_str.tex }{
1242                 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1243             }{
1244                 % try english as default
1245                 \stex_debug:n{Checking~\l_tmpa_str.en.tex}
1246                 \IfFileExists{ \l_tmpa_str.en.tex }{
1247                     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1248                 }{
1249                     \msg_set:nnn{stex}{error/modulemissing}{
1250                         No~file~for~module~#1?#4~found
1251                     }
1252                     \msg_error:nn{stex}{error/modulemissing}
1253                 }
1254             }
1255         }
1256     }
1257 }
1258 }
1259 }
1260
1261 \seq_set_eq:NN \l_tmpa_seq \g_stex_modules_in_file_seq
1262 \seq_clear:N \g_stex_modules_in_file_seq
1263 % \exp_args:Nnx \use:nn {
1264     \exp_args:No \stex_in_smsmode:nn { \g__stex_importmodule_file_str } {
1265         \seq_clear:N \l_stex_all_modules_seq
1266         \prop_clear:N \l_stex_current_module_prop
1267         \str_set:Nx \l_tmpb_str { #2 }
1268         \str_if_empty:NF \l_tmpb_str {
1269             \stex_set_current_repository:n { #2 }
1270         }
1271         \stex_debug:n{Loading~\g__stex_importmodule_file_str}
1272         \input { \g__stex_importmodule_file_str }
1273     }
1274 % }{
1275
1276 % }
1277 \prop_gput:Noo \g_stex_module_files_prop
1278 \g__stex_importmodule_file_str \g_stex_modules_in_file_seq
1279 \seq_set_eq:NN \g_stex_modules_in_file_seq \l_tmpa_seq
1280
1281 \stex_if_module_exists:nF { #1 ? #4 } {
1282     \msg_set:nnn{stex}{error/modulemissing}{
1283         Module~#1?#4~not~found~in~file~\g__stex_importmodule_file_str
1284     }
1285     \msg_error:nn{stex}{error/modulemissing}
1286 }
1287 }

```

```

1288 \stex_activate_module:n { #1 ? #4 }
1289 }

```

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

`\stex_activate_module:n`

```

1290 \cs_new_protected:Nn \stex_activate_module:n {
1291   \stex_debug:n{Activating~module~#1}
1292   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1293     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1294     \prop_item:cn { c_stex_module_#1_prop } { content }
1295   }
1296 }

```

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

`\importmodule`

```

1297 \NewDocumentCommand \importmodule { 0{} m } {
1298   \stex_import_module_uri:nn { #1 } { #2 }
1299   \stex_debug:n{Importing~module:~
1300     \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1301   }
1302   \stex_if_smsmode:F {
1303     \stex_import_require_module:nnnn
1304     { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1305     { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1306     \stex_annotate_invisible:nnn
1307     {import} { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } {}
1308   }
1309   \exp_args:Nx \stex_add_to_current_module:n {
1310     \stex_import_require_module:nnnn
1311     { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1312     { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1313   }
1314   \exp_args:Nx \stex_add_import_to_current_module:n {
1315     \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1316   }
1317   \stex_smsmode_set_codes:
1318 }

```

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

`\usemodule`

```

1319 \NewDocumentCommand \usemodule { 0{} m } {
1320   \stex_if_smsmode:F {
1321     \stex_import_module_uri:nn { #1 } { #2 }
1322     \stex_import_require_module:nnnn
1323     { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1324     { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1325     \stex_annotate_invisible:nnn
1326     {usemodule} { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } {}
1327   }
1328   \stex_smsmode_set_codes:
1329 }

```

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

`\g_stex_modules_in_file_seq`
`\g_stex_module_files_prop`

```
1330 \seq_new:N \g_stex_modules_in_file_seq
1331 \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

```
1332 <@@=stex_symdecl>
```

`\l_stex_all_symbols_seq` Stores all available symbols

```
1333 \seq_new:N \l_stex_all_symbols_seq
```

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

`\STEXsymbol`

```
1334 \NewDocumentCommand \STEXsymbol { m } {
1335   \stex_get_symbol:n { #1 }
1336   \exp_args:No
1337   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1338 }
```

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

symdecl arguments:

```
1339 \keys_define:nn { stex / symdecl } {
1340   name      .tl_set:x:N = \l_stex_symdecl_name_str ,
1341   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1342   args      .tl_set:x:N = \l_stex_symdecl_args_str ,
1343   type      .tl_set:N   = \l_stex_symdecl_type_tl ,
1344   align     .tl_set:N   = \l_stex_symdecl_align_str , % TODO(?)
1345   gfc       .tl_set:N   = \l_stex_symdecl_gfc_str , % TODO(?)
1346   specializes .tl_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
1347   def       .tl_set:N   = \l_stex_symdecl_definiens_tl
1348 }
1349
1350 \bool_new:N \l_stex_symdecl_make_macro_bool
1351
1352 \cs_new_protected:Nn \__stex_symdecl_args:n {
1353   \str_clear:N \l_stex_symdecl_name_str
1354   \str_clear:N \l_stex_symdecl_args_str
1355   \bool_set_false:N \l_stex_symdecl_local_bool
1356   \tl_clear:N \l_stex_symdecl_type_tl
1357   \tl_clear:N \l_stex_symdecl_definiens_tl
1358 }
1359
1360 \keys_set:nn { stex / symdecl } { #1 }
1361
1362 \exp_args:NNo \str_set:Nn \l_stex_symdecl_name_str
1363   \l_stex_symdecl_name_str
1364 \exp_args:NNo \str_set:Nn \l_stex_symdecl_args_str
1365   \l_stex_symdecl_args_str
1366 }
```


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

```

1366
1367 \NewDocumentCommand \symdecl { s O{} m } {
1368   \_stex_symdecl_args:n { #2 }
1369   \IfBooleanTF #1 {
1370     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1371   } {
1372     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1373   }
1374   \stex_symdecl_do:n { #3 }
1375   \stex_smsmode_set_codes:
1376 }

```

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

`\stex_symdecl_do:n`

```

1377 \cs_new_protected:Nn \stex_symdecl_do:n {
1378   \stex_if_in_module:F {
1379     % TODO throw error? some default namespace?
1380   }
1381
1382   \str_if_empty:NT \l_stex_symdecl_name_str {
1383     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1384   }
1385
1386   \prop_if_exist:cT { g_stex_symdecl_
1387     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1388     \prop_item:Nn \l_stex_current_module_prop {name} ?
1389     \l_stex_symdecl_name_str
1390     _prop
1391   }{
1392     % TODO throw error (beware of circular dependencies)
1393   }
1394
1395   \prop_clear:N \l_tmpa_prop
1396   \prop_put:Nnx \l_tmpa_prop { module } {
1397     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1398     \prop_item:Nn \l_stex_current_module_prop {name}
1399   }
1400   \seq_clear:N \l_tmpa_seq
1401   \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
1402   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1403   \prop_put:Nno \l_tmpa_prop { local } \l_stex_symdecl_local_bool
1404   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1405
1406   \exp_args:No \stex_add_constant_to_current_module:n {
1407     \l_stex_symdecl_name_str
1408   }
1409
1410   % arity/args
1411   \int_zero:N \l_tmpb_int
1412
1413   \bool_set_true:N \l_tmpa_bool

```

```

1414 \str_map_inline:Nn \l_stex_symdecl_args_str {
1415   \token_case_meaning:NnF ##1 {
1416     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1417     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1418     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1419     {\tl_to_str:n a} {
1420       \bool_set_false:N \l_tmpa_bool
1421       \int_incr:N \l_tmpb_int
1422     }
1423     {\tl_to_str:n B} {
1424       \bool_set_false:N \l_tmpa_bool
1425       \int_incr:N \l_tmpb_int
1426     }
1427   }{
1428     \msg_set:nnn{stex}{error/wrongargs}{
1429       args~value~in~symbol~declaration~for~
1430       \prop_item:Nn \l_stex_current_module_prop {ns} ?
1431       \prop_item:Nn \l_stex_current_module_prop {name} ?
1432       \l_stex_symdecl_name_str ~
1433       needs~to~be~
1434       i,~a,~b~or~B,~but~##1~given
1435     }
1436     \msg_error:nn{stex}{error/wrongargs}
1437   }
1438 }
1439 \bool_if:NTF \l_tmpa_bool {
1440   % possibly numeric
1441   \str_if_empty:NTF \l_stex_symdecl_args_str {
1442     \prop_put:Nnn \l_tmpa_prop { args } {}
1443     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1444   }{
1445     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1446     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1447     \str_clear:N \l_tmpa_str
1448     \int_step_inline:nn \l_tmpa_int {
1449       \str_put_right:Nn \l_tmpa_str i
1450     }
1451     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1452   }
1453 } {
1454   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1455   \prop_put:Nnx \l_tmpa_prop { arity }
1456   { \str_count:N \l_stex_symdecl_args_str }
1457 }
1458 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
1459
1460
1461 % semantic macro
1462
1463 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1464   \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1465     \prop_item:Nn \l_tmpa_prop { module } ?
1466     \prop_item:Nn \l_tmpa_prop { name }
1467   } }

```

```

1468
1469 \bool_if:NF \l_stex_symdecl_local_bool {
1470   \exp_args:Nx \stex_add_to_current_module:n {
1471     \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1472       \prop_item:Nn \l_tmpa_prop { module } ?
1473       \prop_item:Nn \l_tmpa_prop { name }
1474     } }
1475   }
1476 }
1477 }
1478
1479 % add to all symbols
1480
1481 \bool_if:NF \l_stex_symdecl_local_bool {
1482   \exp_args:Nx \stex_add_to_current_module:n {
1483     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1484       \prop_item:Nn \l_tmpa_prop { module } ?
1485       \prop_item:Nn \l_tmpa_prop { name }
1486     }
1487   }
1488 }
1489
1490 \stex_debug:n{New~symbol:~
1491   \prop_item:Nn \l_tmpa_prop { module } ?
1492   \prop_item:Nn \l_tmpa_prop { name }^^J
1493   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
1494   Args:~\prop_item:Nn \l_tmpa_prop { args }
1495 }
1496
1497 % circular dependencies require this:
1498
1499 \prop_if_exist:cF {
1500   g_stex_symdecl_
1501   \prop_item:Nn \l_tmpa_prop { module } ?
1502   \prop_item:Nn \l_tmpa_prop { name }
1503   _prop
1504 } {
1505   \prop_gset_eq:cN {
1506     g_stex_symdecl_
1507     \prop_item:Nn \l_tmpa_prop { module } ?
1508     \prop_item:Nn \l_tmpa_prop { name }
1509     _prop
1510   } \l_tmpa_prop
1511 }
1512
1513 \stex_if_smsmode:TF {
1514   \bool_if:NF \l_stex_symdecl_local_bool {
1515     \exp_args:Nx \stex_addtosms:n {
1516       \prop_gset_from_keyval:cn {
1517         g_stex_symdecl_
1518         \prop_item:Nn \l_tmpa_prop { module } ?
1519         \prop_item:Nn \l_tmpa_prop { name }
1520         _prop
1521       } {

```

```

1522     name      = \prop_item:Nn \l_tmpa_prop { name }      ,
1523     module    = \prop_item:Nn \l_tmpa_prop { module }    ,
1524     notations = \prop_item:Nn \l_tmpa_prop { notations }  ,
1525     local     = \prop_item:Nn \l_tmpa_prop { local }      ,
1526     type      = \prop_item:Nn \l_tmpa_prop { type }       ,
1527     args      = \prop_item:Nn \l_tmpa_prop { args }       ,
1528     arity     = \prop_item:Nn \l_tmpa_prop { arity }      ,
1529     assocs    = \prop_item:Nn \l_tmpa_prop { assocs }     ,
1530   }
1531   \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1532     \prop_item:Nn \l_tmpa_prop { module } ?
1533     \prop_item:Nn \l_tmpa_prop { name }
1534   }
1535 }
1536 }
1537 }{
1538   \exp_args:NNx \seq_put_right:Nn \l_stex_all_symbols_seq {
1539     \prop_item:Nn \l_tmpa_prop { module } ?
1540     \prop_item:Nn \l_tmpa_prop { name }
1541   }
1542   \stex_annotate_invisible:nnn {symdecl} {
1543     \prop_item:Nn \l_tmpa_prop { module } ?
1544     \prop_item:Nn \l_tmpa_prop { name }
1545   } {
1546     \stex_annotate_invisible:nnn{type}{}{${\l_stex_symdecl_type_tl$}
1547     \stex_annotate_invisible:nnn{args}{}{
1548       \prop_item:Nn \l_tmpa_prop { args }
1549     }
1550     \stex_annotate_invisible:nnn{macroname}{}{#1}
1551     \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
1552       \stex_annotate_invisible:nnn{definiens}{}{
1553         ${\l_stex_symdecl_definiens_tl$}
1554       }
1555     }
1556   }
1557 }

```

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

`\stex_get_symbol:n`

```

1558 \str_new:N \l_stex_get_symbol_uri_str
1559
1560 \cs_new_protected:Nn \stex_get_symbol:n {
1561   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
1562     \__stex_symdecl_get_symbol_from_cs:n { #1 }
1563   }{
1564     % argument is a string
1565     % is it a command name?
1566     \cs_if_exist:cTF { #1 }{
1567       \cs_set_eq:Nc \l_tmpa_tl { #1 }
1568       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
1569       \str_if_empty:NNTF \l_tmpa_str {
1570         \exp_args:Nx \cs_if_eq:NNTF {
1571           \tl_head:N \l_tmpa_tl

```

```

1572     } \stex_invoke_symbol:n {
1573       \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
1574     }{
1575       \__stex_symdecl_get_symbol_from_string:n { #1 }
1576     }
1577   } {
1578     \__stex_symdecl_get_symbol_from_string:n { #1 }
1579   }
1580 }{
1581   % argument is not a command name
1582   \__stex_symdecl_get_symbol_from_string:n { #1 }
1583   % \l_stex_all_symbols_seq
1584 }
1585 }
1586 }
1587
1588 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
1589   \prop_get:NnN \l_stex_current_module_prop
1590   { constants } \l_tmpa_seq
1591   \seq_if_in:NnTF \l_tmpa_seq { #1 } {
1592     \str_set:Nx \l_stex_get_symbol_uri_str {
1593       \prop_item:Nn \l_stex_current_module_prop { ns } ?
1594       \prop_item:Nn \l_stex_current_module_prop { name } ? #1
1595     }
1596   } {
1597     \tl_set:Nn \l_tmpa_tl {
1598       \msg_set:nnn{stex}{error/unknownsymbol}{
1599         No~symbol~#1~found!
1600       }
1601       \msg_error:nn{stex}{error/unknownsymbol}
1602     }
1603     \str_set:Nn \l_tmpa_str { #1 }
1604     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1605     \seq_map_inline:Nn \l_stex_all_symbols_seq {
1606       \str_set:Nn \l_tmpb_str { ##1 }
1607       \str_if_eq:eeT { \l_tmpa_str } {
1608         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1609       } {
1610         \seq_map_break:n {
1611           \tl_set:Nn \l_tmpa_tl {
1612             \str_set:Nn \l_stex_get_symbol_uri_str {
1613               ##1
1614             }
1615           }
1616         }
1617       }
1618     }
1619     \l_tmpa_tl
1620   }
1621 }
1622
1623 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
1624   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
1625   { \tl_tail:N \l_tmpa_tl }

```

```

1626 \tl_if_single:NTF \l_tmpa_tl {
1627   \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
1628     \exp_after:wN \str_set:Nn \exp_after:wN
1629       \l_stex_get_symbol_uri_str \l_tmpa_tl
1630   }{
1631     % TODO
1632     % tail is not a single group
1633   }
1634 }{
1635   % TODO
1636   % tail is not a single group
1637 }
1638 }

```

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

4.7 Notations

```

1639 <@=stex_notation>

notation arguments:
1640 \keys_define:nn { stex / notation } {
1641   lang .tl_set_x:N = \l__stex_notation_lang_str ,
1642   variant .tl_set_x:N = \l__stex_notation_variant_str ,
1643   prec .tl_set_x:N = \l__stex_notation_prec_str ,
1644   op .tl_set:N = \l__stex_notation_op_tl ,
1645   unknown .code:n = \str_set:Nx
1646     \l__stex_notation_variant_str \l_keys_key_str
1647 }
1648
1649 \cs_new_protected:Nn \__stex_notation_args:n {
1650   \str_clear:N \l__stex_notation_lang_str
1651   \str_clear:N \l__stex_notation_variant_str
1652   \str_clear:N \l__stex_notation_prec_str
1653   \tl_clear:N \l__stex_notation_op_tl
1654
1655   \keys_set:nn { stex / notation } { #1 }
1656
1657   \str_set:Nx \l__stex_notation_lang_str \l__stex_notation_lang_str
1658   \str_set:Nx \l__stex_notation_variant_str \l__stex_notation_variant_str
1659   \str_set:Nx \l__stex_notation_prec_str \l__stex_notation_prec_str
1660 }

```

`\notation`

```

1661 \NewDocumentCommand \notation { 0{ } m } {
1662   \__stex_notation_args:n { #1 }
1663   \tl_clear:N \l_stex_symdecl_definiens_tl
1664   \stex_get_symbol:n { #2 }
1665   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
1666 }

```

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

`\stex_notation_do:nn`

```

1667 \cs_new_protected:Nn \stex_notation_do:nn {

```

```

1668 \prop_set_eq:Nc \l_tmpa_prop {
1669   g_stex_symdecl_ #1 _prop
1670 }
1671
1672 \prop_clear:N \l_tmpb_prop
1673 \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
1674 \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
1675 \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
1676
1677 % precedences
1678 \seq_clear:N \l_tmpb_seq
1679 \exp_args:NNno
1680 \str_if_empty:NTF \l__stex_notation_prec_str {
1681   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1682   \int_compare:nNnTF \l_tmpa_str = 0 {
1683     \exp_args:NNnx
1684     \prop_put:Nno \l_tmpb_prop { opprec }
1685     { \infprec }
1686   }{
1687     \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
1688   }
1689 } {
1690   \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
1691     \exp_args:NNnx
1692     \prop_put:Nno \l_tmpb_prop { opprec }
1693     { \infprec }
1694     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1695     \int_step_inline:nn { \l_tmpa_str } {
1696       \exp_args:NNx
1697       \seq_put_right:Nn \l_tmpb_seq { \neginfprec }
1698     }
1699   }{
1700     \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
1701     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
1702       \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
1703       \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
1704         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
1705         \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
1706         \seq_map_inline:Nn \l_tmpa_seq {
1707           \seq_put_right:Nn \l_tmpb_seq { ##1 }
1708         }
1709       }
1710       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1711     }{
1712       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1713       \int_compare:nNnTF \l_tmpa_str = 0 {
1714         \exp_args:NNnx
1715         \prop_put:Nno \l_tmpb_prop { opprec }
1716         { \infprec }
1717       }{
1718         \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
1719       }
1720     }
1721   }

```

```

1722 }
1723
1724 \seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
1725 \int_step_inline:nn { \l_tmpa_str } {
1726   \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
1727     \exp_args:NNx
1728     \seq_put_right:Nn \l_tmpb_seq {
1729       \prop_item:Nn \l_tmpb_prop { opprec }
1730     }
1731   }
1732 }
1733
1734 \prop_put:Nno \l_tmpb_prop { argprec } \l_tmpb_seq
1735 \tl_clear:N \l_tmpa_tl
1736
1737 \int_compare:nNnTF \l_tmpa_str = 0 {
1738   \exp_args:NNe
1739   \cs_set:Npn \l__stex_notation_macrocode_cs {
1740     \stex_term_math_oms:nnnn { #1 }
1741     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1742     { \prop_item:Nn \l_tmpb_prop { opprec } }
1743     { \exp_not:n { #2 } }
1744   }
1745   \__stex_notation_final:
1746 }{
1747   \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
1748   \str_if_in:NnTF \l_tmpb_str b {
1749     \exp_args:Nne \use:nn
1750     {
1751       \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1752       \cs_set:Npn \l_tmpa_str { { {
1753         \stex_term_math_omb: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     }{
1759       \str_if_in:NnTF \l_tmpb_str B {
1760         \exp_args:Nne \use:nn
1761         {
1762           \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1763           \cs_set:Npn \l_tmpa_str { { {
1764             \stex_term_math_omb:nnnn { #1 }
1765             { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1766             { \prop_item:Nn \l_tmpb_prop { opprec } }
1767             { \exp_not:n { #2 } }
1768           } }
1769         }{
1770           \exp_args:Nne \use:nn
1771           {
1772             \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1773             \cs_set:Npn \l_tmpa_str { { {
1774               \stex_term_math_oma:nnnn { #1 }
1775               { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }

```



```

1776         { \prop_item:Nn \l_tmpb_prop { opprec } }
1777         { \exp_not:n { #2 } }
1778     } }
1779 }
1780 }
1781
1782 \int_zero:N \l_tmpa_int
1783 \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
1784 \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
1785 \__stex_notation_arguments:
1786 }
1787 }

```

(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

```

1788 \cs_new_protected:Nn \__stex_notation_arguments: {
1789   \int_incr:N \l_tmpa_int
1790   \str_if_empty:NTF \l_tmpa_str {
1791     \__stex_notation_final:
1792   }{
1793     \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
1794     \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
1795     \str_if_eq:VnTF \l_tmpb_str a {
1796       \__stex_notation_argument_assoc:n
1797     }{
1798       \str_if_eq:VnTF \l_tmpb_str B {
1799         \__stex_notation_argument_assoc:n
1800       }{
1801         \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1802         \tl_put_right:Nx \l_tmpa_tl {
1803           { \stex_term_math_arg:nnn
1804             { \int_use:N \l_tmpa_int }
1805             { \l_tmpb_str }
1806             { ###\int_use:N \l_tmpa_int }
1807           }
1808         }
1809         \__stex_notation_arguments:
1810       }
1811     }
1812   }
1813 }

```

(End definition for __stex_notation_arguments:.)

__stex_notation_argument_assoc:n

```

1814 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
1815   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1816   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
1817   \tl_put_right:Nx \l_tmpa_tl {
1818     { \stex_term_math_assoc_arg:nnnn
1819       { \int_use:N \l_tmpa_int }
1820       { \l_tmpb_str }
1821       \exp_args:No \exp_not:n

```

```

1822     {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
1823     { ####\int_use:N \l_tmpa_int }
1824   }
1825 }
1826 \__stex_notation_arguments:
1827 }

```

(End definition for __stex_notation_argument_assoc:n.)

__stex_notation_final: Called after processing all notation arguments

```

1828 \cs_new_protected:Nn \__stex_notation_final: {
1829   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
1830   \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
1831   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
1832   \exp_args:Nne \use:nn
1833   {
1834     \cs_generate_from_arg_count:cNnn {
1835       stex_notation_ \l_tmpa_str \c_hash_str
1836       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1837       _cs
1838     }
1839     \cs_gset:Npn \l_tmpb_str { { {
1840       \exp_after:wN \exp_after:wN \exp_after:wN
1841       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
1842       { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
1843     } } }
1844
1845     \tl_if_empty:NF \l__stex_notation_op_tl {
1846       \cs_gset:cpx {
1847         stex_op_notation_ \l_tmpa_str \c_hash_str
1848         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1849         _cs
1850       } {
1851         \stex_term oms:nnn {
1852           \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
1853           \l__stex_notation_lang_str
1854         }{
1855           \l_tmpa_str
1856         }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } } }
1857     }
1858   }
1859
1860
1861
1862   \stex_debug:n{
1863     Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1864     ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
1865     Operator~precedence:~
1866     \prop_item:Nn \l_tmpb_prop { opprec }^^J
1867     Argument~precedences:~
1868     \seq_use:Nn \l_tmpa_seq { ,~ }^^J
1869     Notation: \cs_meaning:c {
1870       stex_notation_ \l_tmpa_str \c_hash_str
1871       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str

```

```

1872     _cs
1873   }
1874 }
1875
1876 \prop_gset_eq:cN {
1877   g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
1878   \c_hash_str \l__stex_notation_lang_str _prop
1879 } \l_tmpb_prop
1880
1881 \exp_args:Nx
1882 \stex_add_to_current_module:n {
1883   \prop_get:cnN {
1884     g_stex_symdecl_
1885     \prop_item:Nn \l_tmpb_prop { symbol }
1886     _prop
1887   } { notations } \exp_not:N \l_tmpa_seq
1888   \seq_put_right:Nn \exp_not:N \l_tmpa_seq {
1889     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1890   }
1891   \prop_put:cno {
1892     g_stex_symdecl_
1893     \prop_item:Nn \l_tmpb_prop { symbol }
1894     _prop
1895   } { notations } \exp_not:N \l_tmpa_seq
1896 }
1897
1898 \stex_if_smsmode:TF {
1899   \stex_smsmode_set_codes:
1900   \exp_args:Nx \stex_addtosms:n {
1901     \prop_gset_from_keyval:cn {
1902       g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
1903       \c_hash_str \l__stex_notation_lang_str _prop
1904     } {
1905       symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
1906       language    = \prop_item:Nn \l_tmpb_prop { language }    ,
1907       variant     = \prop_item:Nn \l_tmpb_prop { variant }      ,
1908       opprec      = \prop_item:Nn \l_tmpb_prop { opprec }       ,
1909       argprecs    = \prop_item:Nn \l_tmpb_prop { argprecs }     ,
1910     }
1911   }
1912 }{
1913   \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
1914   \seq_put_right:Nx \l_tmpa_seq {
1915     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1916   }
1917   \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
1918   \prop_set_eq:cN {
1919     g_stex_symdecl_ \l_tmpa_str _prop
1920   } \l_tmpa_prop
1921
1922   % HTML annotations
1923   \stex_annotate_invisible:nnn { notation }
1924   { \prop_item:Nn \l_tmpb_prop { symbol } } {
1925     \stex_annotate_invisible:nnn { notationfragment }

```

```

1926     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
1927 \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
1928 \stex_annotate_invisible:nnn { precedence }
1929   { \prop_item:Nn \l_tmpb_prop { opprec };
1930     \seq_use:Nn \l_tmpa_seq { x }
1931   }{}
1932
1933 \int_zero:N \l_tmpa_int
1934 \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
1935 \tl_clear:N \l_tmpa_tl
1936 \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
1937   \int_incr:N \l_tmpa_int
1938   \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
1939   \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
1940   \str_if_eq:VnTF \l_tmpb_str a {
1941     \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
1942       \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
1943       \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
1944     } }
1945   }{
1946     \str_if_eq:VnTF \l_tmpb_str B {
1947       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
1948         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
1949         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
1950       } }
1951     }{
1952       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
1953         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
1954       } }
1955     }
1956   }
1957 }
1958 \stex_annotate_invisible:nnn { notationcomp }{}{
1959   $ \exp_args:Nno \use:nn { \use:c {
1960     stex_notation_ \prop_item:Nn \l_tmpb_prop { symbol }
1961     \c_hash_str \l__stex_notation_variant_str
1962     \c_hash_str \l__stex_notation_lang_str _cs
1963   } } { \l_tmpa_tl } $
1964 }
1965 }
1966 }
1967 }

```

(End definition for _stex_notation_final:.)

\symdef

```

1968 \keys_define:nn { stex / symdef } {
1969   name .tl_set_x:N = \l_stex_symdecl_name_str ,
1970   local .bool_set:N = \l_stex_symdecl_local_bool ,
1971   args .tl_set_x:N = \l_stex_symdecl_args_str ,
1972   type .tl_set:N = \l_stex_symdecl_type_tl ,
1973   def .tl_set:N = \l_stex_symdecl_definiens_tl ,
1974   op .tl_set:N = \l__stex_notation_op_tl ,
1975   lang .tl_set_x:N = \l__stex_notation_lang_str ,

```

```

1976   variant .tl_set_x:N = \l__stex_notation_variant_str ,
1977   prec    .tl_set_x:N = \l__stex_notation_prec_str ,
1978   unknown .code:n      = \str_set:Nx
1979           \l__stex_notation_variant_str \l_keys_key_str
1980 }
1981
1982 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
1983   \str_clear:N \l_stex_symdecl_name_str
1984   \str_clear:N \l_stex_symdecl_args_str
1985   \bool_set_false:N \l_stex_symdecl_local_bool
1986   \tl_clear:N \l_stex_symdecl_type_tl
1987   \tl_clear:N \l_stex_symdecl_definiens_tl
1988   \str_clear:N \l__stex_notation_lang_str
1989   \str_clear:N \l__stex_notation_variant_str
1990   \str_clear:N \l__stex_notation_prec_str
1991   \tl_clear:N \l__stex_notation_op_tl
1992
1993   \keys_set:nn { stex /symdef } { #1 }
1994
1995   \exp_args:NNo \str_set:Nn \l_stex_symdecl_name_str
1996     \l_stex_symdecl_name_str
1997   \exp_args:NNo \str_set:Nn \l_stex_symdecl_args_str
1998     \l_stex_symdecl_args_str
1999   \exp_args:NNo \str_set:Nn \l__stex_notation_lang_str
2000     \l__stex_notation_lang_str
2001   \exp_args:NNo \str_set:Nn \l__stex_notation_variant_str
2002     \l__stex_notation_variant_str
2003   \exp_args:NNo \str_set:Nn \l__stex_notation_prec_str
2004     \l__stex_notation_prec_str
2005 }
2006
2007 \NewDocumentCommand \symdef { O{} m } {
2008   \__stex_notation_symdef_args:n { #1 }
2009   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2010   \stex_symdecl_do:n { #2 }
2011   \exp_args:Nx \stex_notation_do:nn {
2012     \prop_item:Nn \l_tmpa_prop { module } ?
2013     \prop_item:Nn \l_tmpa_prop { name }
2014   }
2015 }

```

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

`\stex_invoke_symbol:n` Invokes a semantic macro

```

2016 %\cs_new_protected:Nn \stex_invoke_symbol:n {
2017 %  \peek_charcode_remove:NTF ! {
2018 %    \stex_term_custom:nn { #1 } { }
2019 %  } {
2020 %    \if_mode_math:
2021 %      \exp_after:wN \__stex_notation_invoke_math:n
2022 %    \else:
2023 %      \exp_after:wN \__stex_notation_invoke_text:n
2024 %    \fi: { #1 }
2025 %  }

```

```

2026 %}
2027
2028 \cs_new_protected:Nn \stex_invoke_symbol:n {
2029   \if_mode_math:
2030     \exp_after:wN \__stex_notation_invoke_math:n
2031   \else:
2032     \exp_after:wN \__stex_notation_invoke_text:n
2033   \fi: { #1 }
2034 }

```

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

`__stex_notation_invoke_math:n`

```

2035 \cs_new_protected:Nn \__stex_notation_invoke_math:n {
2036   \peek_charcode_remove:NTF ! {
2037     \peek_charcode:NTF [ {
2038       \__stex_notation_invoke_op:nw { #1 }
2039     }{
2040       \__stex_notation_invoke_op:nw { #1 } []
2041     }
2042   }{
2043     \peek_charcode_remove:NTF * {
2044       \__stex_notation_invoke_text:n { #1 }
2045     }{
2046       \peek_charcode:NTF [ {
2047         \__stex_notation_invoke_math:nw { #1 }
2048       }{
2049         \__stex_notation_invoke_math:nw { #1 } []
2050       }
2051     }
2052   }
2053 }

```

(End definition for `__stex_notation_invoke_math:n`.)

`__stex_notation_invoke_op:nw`

```

2054 \cs_new_protected:Npn \__stex_notation_invoke_op:nw #1 [#2] {
2055   \__stex_notation_args:n { #2 }
2056   \cs_if_exist:cTF {
2057     stex_op_notation_ #1 \c_hash_str
2058     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str _cs
2059   }{
2060     \csname stex_op_notation_ #1 \c_hash_str
2061       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str _cs
2062     \endcsname
2063   }{
2064     % TODO throw error
2065   }
2066 }

```

(End definition for `__stex_notation_invoke_op:nw`.)

`__stex_notation_invoke_math:nw`

```

2067 \cs_new_protected:Npn \__stex_notation_invoke_math:nw #1 [#2] {
2068   \__stex_notation_args:n { #2 }

```

```

2069 \prop_set_eq:Nc \l_tmpa_prop {
2070   g_stex_symdecl_ #1 _prop
2071 }
2072 \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
2073 \seq_if_empty:NTF \l_tmpa_seq {
2074   \msg_set:nnn{stex}{error/nonotations}{
2075     Symbol~#1~used,~but~has~no~notations!
2076   }
2077   \msg_error:nn{stex}{error/nonotations}
2078 } {
2079   \seq_if_in:NxTF \l_tmpa_seq
2080   { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2081     \use:c{
2082       stex_notation_ #1 \c_hash_str
2083       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2084       _cs
2085     }
2086   }{
2087     \str_if_empty:NTF \l__stex_notation_variant_str {
2088       \str_if_empty:NTF \l__stex_notation_lang_str {
2089         \seq_get_left:NN \l_tmpa_seq \l_tmpa_str
2090         \use:c{
2091           stex_notation_ #1 \c_hash_str \l_tmpa_str
2092           _cs
2093         }
2094       }{
2095         \msg_set:nnn{stex}{error/wrongnotation}{
2096           Symbol~#1~has~no~notation~
2097           \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2098         }
2099         \msg_error:nn{stex}{error/wrongnotation}
2100       }
2101     }{
2102       \msg_set:nnn{stex}{error/wrongnotation}{
2103         Symbol~#1~has~no~notation~
2104         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2105       }
2106       \msg_error:nn{stex}{error/wrongnotation}
2107     }
2108   }
2109 }
2110 }

```

(End definition for __stex_notation_invoke_math:nw.)

__stex_notation_invoke_text:n

```

2111 \cs_new_protected:Nn \__stex_notation_invoke_text:n {
2112   \peek_charcode_remove:NTF ! {
2113     \stex_term_custom:nn { #1 } { }
2114   }{
2115     \prop_set_eq:Nc \l_tmpa_prop {
2116       g_stex_symdecl_ #1 _prop
2117     }
2118     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str

```

```

2119 \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2120 }
2121 }

```

(End definition for `_stex_notation_invoke_text:n`.)

4.8 Terms

```

2122 <@@=stex_term>

```

Precedences:

```

\infprec
\neginfprec
\l__stex_term_downprec
2123 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2124 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2125 \int_new:N \l__stex_term_downprec
2126 \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
2127 \tl_set:Nn \l__stex_term_left_bracket_str (
2128 \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

```

2129 \cs_new_protected:Nn \_stex_term_maybe_brackets:nn {
2130 \int_compare:nNnTF { #1 } > \l__stex_term_downprec {
2131 \bool_if:NTF \l_stex_inarray_bool { #2 }{
2132 \dobrackets { #2 }
2133 }
2134 }{ #2 }
2135 }

```

(End definition for `_stex_term_maybe_brackets:nn`.)

`\dobrackets`

```

2136 %\RequirePackage{scalerel}
2137 \cs_new_protected:Npn \dobrackets #1 {
2138 %\ThisStyle{\if D\m@switch
2139 % \exp_args:Nnx \use:nn
2140 % { \exp_after:wN \left\l__stex_term_left_bracket_str #1 }
2141 % { \exp_not:N\right\l__stex_term_right_bracket_str }
2142 % \else
2143 % \exp_args:Nnx \use:nn
2144 % { \l__stex_term_left_bracket_str #1 }
2145 % { \l__stex_term_right_bracket_str }
2146 %\fi}
2147 }

```

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

\withbrackets

```
2148 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2149   \exp_args:Nnx \use:nn
2150   {
2151     \tl_set:Nx \l__stex_term_left_bracket_str { #1 }
2152     \tl_set:Nx \l__stex_term_right_bracket_str { #2 }
2153     #3
2154   }
2155   {
2156     \tl_set:Nn \exp_not:N \l__stex_term_left_bracket_str
2157     {\l__stex_term_left_bracket_str}
2158     \tl_set:Nn \exp_not:N \l__stex_term_right_bracket_str
2159     {\l__stex_term_right_bracket_str}
2160   }
2161 }
```

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

\STEXinvisible

```
2162 \cs_new_protected:Npn \STEXinvisible #1 {
2163   \stex_annotate_invisible:n { #1 }
2164 }
```

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

OMDOC terms:

_stex_term_math_oms:nnnn

```
2165 \cs_new_protected:Nn \_stex_term_oms:nnn {
2166   \stex_annotate:nnn{ OMID }{ #2 }{
2167     \stex_highlight_term:nn { #1 } { #3 }
2168   }
2169 }
2170
2171 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2172   \_stex_term_maybe_brackets:nn { #3 }{
2173     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2174   }
2175 }
```

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

_stex_term_math_oma:nnnn

```
2176 \cs_new_protected:Nn \_stex_term_oma:nnn {
2177   \stex_annotate:nnn{ OMA }{ #2 }{
2178     \stex_highlight_term:nn { #1 } { #3 }
2179   }
2180 }
2181
2182 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2183   \_stex_term_maybe_brackets:nn { #3 }{
2184     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2185   }
2186 }
```

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

`_stex_term_math_omb:nnnn`

```

2187 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2188   \stex_annotate:nnn{ OMBIND }{ #2 }{
2189     \stex_highlight_term:nn { #1 } { #3 }
2190   }
2191 }
2192
2193 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2194   \__stex_term_maybe_brackets:nn { #3 }{
2195     \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2196   }
2197 }

```

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

`_stex_term_math_arg:nnn`

```

2198 \cs_new_protected:Nn \_stex_term_arg:nn {
2199   \stex_unhighlight_term:n {
2200     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2201   }
2202 }
2203 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
2204   \exp_args:Nnx \use:nn
2205   { \int_set:Nn \l__stex_term_downprec { #2 }
2206     \stex_term_arg:nn { #1 }{ #3 }
2207   }
2208   { \int_set:Nn \exp_not:N \l__stex_term_downprec { \int_use:N \l__stex_term_downprec } }
2209 }

```

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

`_stex_term_math_assoc_arg:nnnn`

```

2210 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
2211   \seq_set_split:Nnn \l_tmpa_seq , { #4 }
2212   \int_compare:nNnTF { \seq_count:N \l_tmpa_seq } < 2 {
2213     \tl_set:Nn \l_tmpa_tl { #4 }
2214   }{
2215     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
2216     \seq_reverse:N \l_tmpa_seq
2217     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_tl
2218     \tl_set:No \l_tmpa_tl { \l_tmpb_tl }
2219
2220     \seq_map_inline:Nn \l_tmpa_seq {
2221       \exp_args:NNo \tl_set:No \l_tmpa_tl {
2222         \exp_args:Nno
2223         \l_tmpa_cs { ##1 } \l_tmpa_tl
2224       }
2225     }
2226
2227   }
2228   \exp_args:Nnno
2229   \_stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2230 }

```

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

`\stex_term_custom:nn`

```

2231 \cs_new_protected:Nn \stex_term_custom:nn {
2232   \str_set:Nn \l__stex_term_custom_uri { #1 }
2233   \str_set:Nn \l_tmpa_str { #2 }
2234   \tl_clear:N \l_tmpa_tl
2235   \int_zero:N \l_tmpa_int
2236   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2237   \__stex_term_custom_loop:
2238 }

```

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

`__stex_term_custom_loop:`

```

2239 \cs_new_protected:Nn \__stex_term_custom_loop: {
2240   \bool_set_false:N \l_tmpa_bool
2241   \bool_while_do:nn {
2242     \str_if_eq_p:ee X {
2243       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2244     }
2245   }{
2246     \int_incr:N \l_tmpa_int
2247   }
2248
2249   \peek_charcode:NTF [ {
2250     % notation/text component
2251     \__stex_term_custom_component:w
2252   } {
2253     \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2254       % all arguments read => finish
2255       \__stex_term_custom_final:
2256     } {
2257       % arguments missing
2258       \peek_charcode_remove:NTF * {
2259         % invisible, specific argument position or both
2260         \peek_charcode:NTF [ {
2261           % visible specific argument position
2262           \__stex_term_custom_arg:wn
2263         } {
2264           % invisible
2265           \peek_charcode_remove:NTF * {
2266             % invisible specific argument position
2267             \__stex_term_custom_arg_inv:wn
2268           } {
2269             % invisible next argument
2270             \__stex_term_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2271           }
2272         }
2273       } {
2274         % next normal argument
2275         \__stex_term_custom_arg:wn [ \l_tmpa_int + 1 ]
2276       }
2277     }
2278   }
2279 }

```

(End definition for _stex_term_custom_loop:.)

_stex_term_custom_arg_inv:wn

```

2280 \cs_new_protected:Npn \_stex_term_custom_arg_inv:wn [ #1 ] #2 {
2281   \bool_set_true:N \l_tmpa_bool
2282   \_stex_term_custom_arg:wn [ #1 ] { #2 }
2283 }

```

(End definition for _stex_term_custom_arg_inv:wn.)

_stex_term_custom_arg:wn

```

2284 \cs_new_protected:Npn \_stex_term_custom_arg:wn [ #1 ] #2 {
2285   \str_set:Nx \l_tmpb_str {
2286     \str_item:Nn \l_tmpa_str { #1 }
2287   }
2288   \str_case:VnTF \l_tmpb_str {
2289     { X } { } % TODO throw error ?
2290     { i } { \_stex_term_custom_set_X:n { #1 } }
2291     { b } { \_stex_term_custom_set_X:n { #1 } }
2292     { a } { \_stex_term_custom_set_X:n { #1 } } % TODO ?
2293     { B } { \_stex_term_custom_set_X:n { #1 } } % TODO ?
2294   }{}{
2295     % TODO throw error
2296   }
2297
2298   \bool_if:nTF \l_tmpa_bool {
2299     \tl_put_right:Nx \l_tmpa_tl {
2300       \stex_annotate_invisible:n {
2301         \_stex_term_arg:nn { \int_eval:n { #1 } }
2302         \exp_not:n { { #2 } }
2303       }
2304     }
2305   } {
2306     \tl_put_right:Nx \l_tmpa_tl {
2307       \_stex_term_arg:nn { \int_eval:n { #1 } }
2308       \exp_not:n { { #2 } }
2309     }
2310   }
2311
2312   \_stex_term_custom_loop:
2313 }

```

(End definition for _stex_term_custom_arg:wn.)

_stex_term_custom_set_X:n

```

2314 \cs_new_protected:Nn \_stex_term_custom_set_X:n {
2315   \str_set:Nx \l_tmpa_str {
2316     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
2317     X
2318     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
2319   }
2320 }

```

(End definition for _stex_term_custom_set_X:n.)

`_stex_term_custom_component:`

```

2321 \cs_new_protected:Npn \_stex_term_custom_component:w [ #1 ] {
2322   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
2323   \_stex_term_custom_loop:
2324 }

```

(End definition for _stex_term_custom_component:.)

`_stex_term_custom_final:`

```

2325 \cs_new_protected:Nn \_stex_term_custom_final: {
2326   \int_compare:nNnTF \l_tmpb_int = 0 {
2327     \exp_args:Nnno \_stex_term oms:nnn
2328   }{
2329     \str_if_in:NnTF \l_tmpa_str {b} {
2330       \exp_args:Nnno \_stex_term ombind:nnn
2331     } {
2332       \exp_args:Nnno \_stex_term oma:nnn
2333     }
2334   }
2335   { \l__stex_term_custom_uri } { \l__stex_term_custom_uri } { \l_tmpa_tl }
2336 }

```

(End definition for _stex_term_custom_final:.)

`\symref`

`\symname`

```

2337 \NewDocumentCommand \symref { m m }{
2338   \STEXsymbol{#1}!{#2}
2339 }
2340
2341 \keys_define:nn { stex / symname } {
2342   post .tl_set_x:N = \l_stex_symname_post_str
2343 }
2344
2345 \cs_new_protected:Nn \stex_symname_args:n {
2346   \str_clear:N \l_stex_symname_post_str
2347   \keys_set:nn { stex / symname } { #1 }
2348   \exp_args:NNo \str_set:Nn \l_stex_symname_post_str
2349     \l_stex_symname_post_str
2350 }
2351
2352 \NewDocumentCommand \symname { 0{ } m }{
2353   \stex_symname_args:n { #1 }
2354   \stex_get_symbol:n { #2 }
2355   \str_set:Nx \l_tmpa_str {
2356     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
2357   }
2358   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
2359   \exp_args:NNx \use:nn
2360   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }![
2361     \l_tmpa_str \l_stex_symname_post_str
2362   ] }
2363 }

```

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

4.9 Notation Components

2364 $\langle @@=\text{stex_notationcomps} \rangle$

$\backslash\text{stex_highlight_term:nn}$

```

2365 \latexml_if:F {
2366   \scalatex_if:F{
2367     \RequirePackage{pdfcomment}
2368   }
2369 }
2370
2371 \str_new:N \l__stex_notationcomps_highlight_uri_str
2372 \cs_new_protected:Nn \stex_highlight_term:nn {
2373   \exp_args:Nnx
2374   \use:nn {
2375     \str_set:Nx \l__stex_notationcomps_highlight_uri_str { #1 }
2376     #2
2377   } {
2378     \str_set:Nx \exp_not:N \l__stex_notationcomps_highlight_uri_str
2379     { \l__stex_notationcomps_highlight_uri_str }
2380   }
2381 }
2382
2383 \cs_new_protected:Nn \stex_unhighlight_term:n {
2384   % \latexml_if:TF {
2385   %   #1
2386   % } {
2387   %   \scalatex_if:TF {
2388   %     #1
2389   %   } {
2390   %     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
2391   %   }
2392   % }
2393 }
```

(End definition for $\backslash\text{stex_highlight_term:nn}$. This function is documented on page 24.)

$\backslash\text{comp}$
 $\backslash\text{@comp}$
 $\backslash\text{@defemph}$

```

2394 \cs_new_protected:Npn \comp #1 {
2395   \str_if_empty:NF \l__stex_notationcomps_highlight_uri_str {
2396     \scalatex_if:TF {
2397       \stex_annotate:nnn { comp }{ \l__stex_notationcomps_highlight_uri_str }{ #1 }
2398     }{
2399       \exp_args:Nnx \@comp { #1 } { \l__stex_notationcomps_highlight_uri_str }
2400     }
2401   }
2402 }
2403
2404 \cs_new_protected:Npn \@comp #1 #2 {
2405   \pdftooltip {
2406     \textcolor{blue}{#1}
2407   } { #2 }
2408 }
2409
2410 \cs_new_protected:Npn \@defemph #1 #2 {
```

```

2411 \pdftooltip {
2412   \textbf{\textcolor{magenta}{#1}}
2413 } { #2 }
2414 }

```

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

`\ellipses`

```

2415 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix
\parrayline
\parraycell
2416 \bool_new:N \l_stex_inarray_bool
2417 \bool_set_false:N \l_stex_inarray_bool
2418 \NewDocumentCommand \parray { m m } {
2419   \begingroup
2420   \bool_set_true:N \l_stex_inarray_bool
2421   \begin{array}{#1}
2422     #2
2423   \end{array}
2424   \endgroup
2425 }
2426
2427 \NewDocumentCommand \prmatrix { m } {
2428   \begingroup
2429   \bool_set_true:N \l_stex_inarray_bool
2430   \begin{matrix}
2431     #1
2432   \end{matrix}
2433   \endgroup
2434 }
2435
2436 \def \parrayline #1 #2 {
2437   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
2438 }
2439
2440 \def \parraycell #1 {
2441   #1 \bool_if:NT \l_stex_inarray_bool {&}
2442 }

```

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

4.10 Structural Features

```

2443 \<@@=stex_features>

```

`symboldoc`

```

2444 \NewDocumentEnvironment{symboldoc}{ m }{
2445   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
2446   \seq_clear:N \l_tmpb_seq
2447   \seq_map_inline:Nn \l_tmpa_seq {
2448     \stex_get_symbol:n { ##1 }
2449     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
2450       \l_stex_get_symbol_uri_str

```

```

2451     }
2452   }
2453   \par
2454   \exp_args:Nnnx
2455   \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
2456   ){
2457     \end{stex_annotate_env}
2458   }

```

STEXdefinition

```

2459
2460 \NewDocumentCommand \__stex_features_definiendum:w { 0{} m m } {
2461   \stex_get_symbol:n { ##2 }
2462   \scalatex_if:TF {
2463     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { ##3 }
2464   } {
2465     \exp_args:Nnx \@defemph { ##3 } { \l_stex_get_symbol_uri_str }
2466   }
2467 }
2468 \NewDocumentCommand \__stex_features_definame:w { 0{} m } {
2469   % TODO: root
2470   \stex_get_symbol:n { ##2 }
2471   \str_set:Nx \l_tmpa_str {
2472     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
2473   }
2474   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {-}
2475   \scalatex_if:TF {
2476     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
2477       \l_tmpa_str
2478     }
2479   } {
2480     \@defemph {
2481       \l_tmpa_str
2482     } { \l_stex_get_symbol_uri_str }
2483   }
2484 }
2485
2486 \cs_new_protected:Nn \__stex_features_defi_begin:n {
2487   \let\definiendum\__stex_features_definiendum:w
2488   \let\definame\__stex_features_definame:w
2489   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
2490   \seq_clear:N \l_tmpb_seq
2491   \seq_map_inline:Nn \l_tmpa_seq {
2492     \stex_get_symbol:n { ##1 }
2493     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
2494       \l_stex_get_symbol_uri_str
2495     }
2496   }
2497   \exp_args:Nnnx
2498   \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
2499 }
2500
2501 \cs_new_protected:Nn \__stex_features_defi_end: {
2502   \end{stex_annotate_env}

```



```

2503 }
2504
2505 \NewDocumentEnvironment{STEXdefinition}{ m }{
2506   \_stex_features_defi_begin:n { #1 }
2507 }{
2508   \_stex_features_defi_end:
2509 }

```

\setSTEXdefinition

```

2510 \cs_new_protected:Npn \setSTEXdefinition #1 {
2511   \AddToHook{env/#1/before}[stex]{\_stex_features_defi_begin:n{}}
2512   \AddToHook{env/#1/after}[stex]{\_stex_features_defi_end:}
2513 }

```

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

structural@feature

```

2514
2515 \NewDocumentEnvironment{structural@feature}{ m m m }{
2516   \stex_if_in_module:F {
2517     \msg_set:nnn{stex}{error/nomodule}{
2518       Structural~Feature~has~to~occur~in~a~module:\\
2519       Feature~#2~of~type~#1\\
2520       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
2521     }
2522     \msg_error:nn{stex}{error/nomodule}
2523   }
2524
2525   \str_set:Nx \l_stex_module_name_str {
2526     \prop_item:Nn \l_stex_current_module_prop
2527       { name } / #2 - feature
2528   }
2529
2530
2531   \str_clear:N \l_tmpa_str
2532   \seq_clear:N \l_tmpa_seq
2533   \tl_clear:N \l_tmpa_tl
2534   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
2535     origname = #2,
2536     name     = \l_stex_module_name_str ,
2537     ns       = \l_stex_module_ns_str ,
2538     imports  = \exp_not:o { \l_tmpa_seq } ,
2539     constants = \exp_not:o { \l_tmpa_seq } ,
2540     content  = \exp_not:o { \l_tmpa_tl } ,
2541     file     = \exp_not:o { \g_stex_currentfile_seq } ,
2542     lang     = \l_stex_module_lang_str ,
2543     sig      = \l_tmpa_str ,
2544     meta     = \l_tmpa_str ,
2545     feature  = #1 ,
2546   }
2547
2548   \stex_if_smsmode:TF {
2549     \stex_smsmode_set_codes:
2550   } {

```

```

2551     \begin{stex_annotate_env}{ feature:#1 }{
2552     \stex_annotate_invisible:nnn{header}{-}{ #3 }
2553   }
2554   }{
2555     \str_set:Nx \l_tmpa_str {
2556       c_stex_feature_
2557       \prop_item:Nn \l_stex_current_module_prop { ns } ?
2558       \prop_item:Nn \l_stex_current_module_prop { name }
2559       _prop
2560     }
2561     \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
2562     \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
2563     \stex_if_smsmode:TF {
2564       \exp_args:Nx \stex_addtosms:n {
2565         \prop_gset_from_keyval:cn {
2566           c_stex_feature_
2567           \prop_item:Nn \l_stex_current_module_prop { ns } ?
2568           \prop_item:Nn \l_stex_current_module_prop { name }
2569           _prop
2570         } {
2571           origname = #2,
2572           name      = \prop_item:cn { \l_tmpa_str } { name } ,
2573           ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
2574           imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
2575           constants = \prop_item:cn { \l_tmpa_str } { constants } ,
2576           content   = \prop_item:cn { \l_tmpa_str } { content } ,
2577           file      = \prop_item:cn { \l_tmpa_str } { file } ,
2578           lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
2579           sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
2580           meta      = \prop_item:cn { \l_tmpa_str } { meta } ,
2581           feature   = \prop_item:cn { \l_tmpa_str } { feature }
2582         }
2583       }
2584     } {
2585       \end{stex_annotate_env}
2586     }
2587   }
2588

```

structure

```

2589
2590 \prop_new:N \l_stex_all_structures_prop
2591
2592 \keys_define:nn { stex / features / structure } {
2593   name .tl_set_x:N = \l__stex_features_structure_name_str ,
2594 }
2595
2596 \cs_new_protected:Nn \__stex_features_structure_args:n {
2597   \str_clear:N \l__stex_features_structure_name_str
2598   \keys_set:nn { stex / features / structure } { #1 }
2599   \exp_args:NNo \str_set:Nn \l__stex_features_structure_name_str
2600     \l__stex_features_structure_name_str
2601 }
2602

```

```

2603 %\stex_new_feature:nnnn { structure } { 0{ } m } {
2604 % \__stex_features_structure_args:n { ##1 }
2605 % \str_if_empty:NT \l__stex_features_structure_name_str {
2606 % \str_set:Nx \l__stex_features_structure_name_str { ##2 }
2607 % }
2608 %} {
2609 %
2610 %}
2611
2612 \NewDocumentEnvironment{structure}{ 0{ } m }{
2613 \__stex_features_structure_args:n { #1 }
2614 \str_if_empty:NT \l__stex_features_structure_name_str {
2615 \str_set:Nx \l__stex_features_structure_name_str { #2 }
2616 }
2617 \exp_args:Nnnx
2618 \begin{structural@feature}{ structure }
2619 { \l__stex_features_structure_name_str }{}
2620 \seq_clear:N \l_tmpa_seq
2621 \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
2622
2623 }{
2624 \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
2625 \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
2626 \str_set:Nx \l_tmpa_str {
2627 \prop_item:Nn \l_stex_current_module_prop { ns } ?
2628 \prop_item:Nn \l_stex_current_module_prop { name }
2629 }
2630 \seq_map_inline:Nn \l_tmpa_seq {
2631 \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
2632 }
2633 \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
2634 \exp_args:Nnx
2635 \AddToHookNext { env / structure / after }{
2636 \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}}{
2637 \stex_term_math_oms:nnnn { \l_tmpa_str }{}{0}{}
2638 }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
2639 \STEXexport {
2640 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2641 {\prop_item:Nn \l_stex_current_module_prop { origname }}
2642 {\l_tmpa_str}
2643 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2644 {#2}{\l_tmpa_str}
2645 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
2646 % \prop_item:Nn \l_stex_current_module_prop { origname },
2647 % \l_tmpa_str
2648 % }
2649 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
2650 % #2,\l_tmpa_str
2651 % }
2652 % \tl_set:cx { #2 } {
2653 % \stex_invoke_structure:n { \l_tmpa_str }
2654 % }
2655 % }
2656

```

```

2657 \end{structural@feature}
2658 % \g_stex_last_feature_prop
2659 }

\instantiate

2660 \seq_new:N \l__stex_features_structure_field_seq
2661 \str_new:N \l__stex_features_structure_field_str
2662 \str_new:N \l__stex_features_structure_def_tl
2663 \prop_new:N \l__stex_features_structure_prop
2664 \NewDocumentCommand \instantiate { m O{} m }{
2665   \stex_smsmode_set_codes:
2666   \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
2667   \prop_set_eq:Nc \l__stex_features_structure_prop {
2668     c_stex_feature_\l_tmpa_str _prop
2669   }
2670   \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
2671   \seq_map_inline:Nn \l__stex_features_structure_field_seq {
2672     \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
2673     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
2674       \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
2675       \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
2676       {!} \l_tmpa_tl
2677       \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
2678         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
2679         \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
2680         \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
2681       }{
2682         \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
2683         \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
2684         \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
2685         \l_tmpa_tl
2686         \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
2687           \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
2688           \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
2689         }{
2690           \tl_clear:N \l_tmpb_tl
2691         }
2692       }
2693     }{
2694       \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
2695       \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
2696         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
2697         \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
2698         \tl_clear:N \l_tmpa_tl
2699       }{
2700         % TODO throw error
2701       }
2702     }
2703     % \l_tmpa_str: name
2704     % \l_tmpa_tl: definiens
2705     % \l_tmpb_tl: notation
2706     \tl_if_empty:NT \l__stex_features_structure_field_str {
2707       % TODO throw error
2708     }

```

```

2709 \str_clear:N \l_tmpb_str
2710
2711 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
2712 \seq_map_inline:Nn \l_tmpa_seq {
2713   \seq_set_split:Nnn \l_tmpb_seq ? { ###1 }
2714   \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
2715   \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
2716     \seq_map_break:n {
2717       \str_set:Nn \l_tmpb_str { ###1 }
2718     }
2719   }
2720 }
2721 \prop_get:cnN { g_stex_symdecl_ \l_tmpb_str _prop } {args}
2722 \l_tmpb_str
2723
2724 \tl_if_empty:NTF \l_tmpb_tl {
2725   \tl_if_empty:NF \l_tmpa_tl {
2726     \exp_args:Nx \use:n {
2727       \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fe
2728     }
2729   }
2730 }{
2731   \tl_if_empty:NTF \l_tmpa_tl {
2732     \exp_args:Nx \use:n {
2733       \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\
2734     }
2735   }
2736 }{
2737   \exp_args:Nx \use:n {
2738     \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
2739     \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
2740   }
2741 }
2742 }
2743 % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
2744 % \prop_item:Nn \l_stex_current_module_prop {name} ?
2745 % #3/\l__stex_features_structure_field_str
2746 % \par
2747 % \expandafter\present\csname
2748 %   g_stex_symdecl_
2749 %   \prop_item:Nn \l_stex_current_module_prop {ns} ?
2750 %   \prop_item:Nn \l_stex_current_module_prop {name} ?
2751 %   #3/\l__stex_features_structure_field_str
2752 %   _prop
2753 % \endcsname
2754 }
2755
2756 \tl_clear:N \l__stex_features_structure_def_tl
2757
2758 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
2759 \seq_map_inline:Nn \l_tmpa_seq {
2760   \seq_set_split:Nnn \l_tmpb_seq ? { #1 }
2761   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
2762   \exp_args:Nx \use:n {

```

```

2763     \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
2764
2765     }
2766   }
2767
2768   \prop_if_exist:cF {
2769     g_stex_symdecl_
2770     \prop_item:Nn \l_stex_current_module_prop {ns} ?
2771     \prop_item:Nn \l_stex_current_module_prop {name} ?
2772     #3/\l_tmpa_str
2773   _prop
2774   }{
2775     \prop_get:cnN { g_stex_symdecl_ ##1 _prop } {args}
2776     \l_tmpb_str
2777     \exp_args:Nx \use:n {
2778       \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
2779     }
2780   }
2781 }
2782
2783 \symdecl*[type={\STEXsymbol{module-type}}{
2784   \_stex_term_math_oms:nnnn {
2785     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
2786     \prop_item:Nn \l__stex_features_structure_prop {name}
2787   }{}{0}{}
2788   }}{#3}
2789
2790 % TODO: -> sms file
2791
2792 \tl_set:cx{ #3 }{
2793   \stex_invoke_structure:nnn {
2794     \prop_item:Nn \l_stex_current_module_prop {ns} ?
2795     \prop_item:Nn \l_stex_current_module_prop {name} ? #3
2796   } {
2797     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
2798     \prop_item:Nn \l__stex_features_structure_prop {name}
2799   }
2800 }
2801
2802 }

```

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

\stex_invoke_structure:nnn

```

2803 % #1: URI of the instance
2804 % #2: URI of the instantiated module
2805 \cs_new_protected:Nn \stex_invoke_structure:nnn {
2806   \tl_if_empty:nTF{ #3 }{
2807     \prop_set_eq:Nc \l__stex_features_structure_prop {
2808       c_stex_feature_ #2 _prop
2809     }
2810     \tl_clear:N \l_tmpa_tl
2811     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
2812     \seq_map_inline:Nn \l_tmpa_seq {

```

```

2813 \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
2814 \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
2815 \cs_if_exist:cT {
2816   stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
2817 }{
2818   \tl_if_empty:NF \l_tmpa_tl {
2819     \tl_put_right:Nn \l_tmpa_tl {,}
2820   }
2821   \tl_put_right:Nx \l_tmpa_tl {
2822     \stex_invoke_symbol:n {#1/\l_tmpa_str}!
2823   }
2824 }
2825 }
2826 \scalatexBREAK
2827 \exp_args:No \mathstruct \l_tmpa_tl
2828 }{
2829   \stex_invoke_symbol:n{#1/#3}
2830 }
2831 }

```

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

4.11 Put these somewhere

`\MSC`

```

2832 \NewDocumentCommand \MSC {m} {
2833   % TODO
2834 }

```

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

```

2835 \@ifpackageloaded{tikzinput}{
2836   \RequirePackage{stex-tikzinput}
2837 }{}
2838
2839 \AddToHook{begindocument}{
2840   \input{stex-metatheory}
2841 }
2842 \</package>

```

4.12 Metatheory

The default meta theory for an \TeX module. Contains symbols so ubiquitous, that it is virtually impossible to describe any flexiformal content without them, or that are required to annotate even the most primitive symbols with meaningful (foundation-independent) “type”-annotations, or required for basic structuring principles (theorems, definitions).

Foundations should ideally instantiate these symbols with their formal counterparts, e.g. `isa` corresponds to a typing operation in typed setting, or the \in -operator in set-theoretic contexts; `bind` corresponds to a universal quantifier in (n th-order) logic, or a Π in dependent type theories.

```

2843 <*metatheory>
2844 \ExplSyntaxOn
2845 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}

```

```

2846 \begin{@module}[ns=\c_stex_metatheory_ns_str,meta=NONE]{Metatheory}
2847 \ExplSyntaxOff
2848
2849 % is-a (a:A, a \in A, a is an A, etc.)
2850 \symdecl[args=ai]{isa}
2851 \notation[typed]{isa}{#1 \comp: #2}{#1 \comp, #2}
2852 \notation[in]{isa}{#1 \comp\in #2}{#1 \comp, #2}
2853 \notation[pred]{isa}{#2\comp(#1 \comp)}{#1 \comp, #2}
2854
2855 % bind (\forall, \Pi, \lambda etc.)
2856 \symdecl[args=Bi]{bind}
2857 \notation[forall]{bind}{\comp\forall #1.;#2}{#1 \comp, #2}
2858 \notation[\Pi]{bind}{\comp\prod_{#1}#2}{#1 \comp, #2}
2859 \notation[depfun]{bind}{\comp( #1 \comp{ }\; \to\; ) #2}{#1 \comp, #2}
2860
2861 % dummy variable
2862 \symdecl{dummyvar}
2863 \notation[underscore]{dummyvar}{\comp\_}
2864 \notation[dot]{dummyvar}{\comp\cdot}
2865 \notation[dot]{dummyvar}{\comp\cdot}
2866 \notation[dash]{dummyvar}{\comp{\rm --}}
2867
2868 %fromto (function space, Hom-set, implication etc.)
2869 \symdecl[args=ai]{fromto}
2870 \notation[xarrow]{fromto}{#1 \comp\to #2}{#1 \comp\times #2}
2871 \notation[arrow]{fromto}{#1 \comp\to #2}{#1 \comp\to #2}
2872
2873 % mapto (lambda etc.)
2874 %\symdecl[args=Bi]{mapto}
2875 %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
2876 %\notation[lambda]{mapto}{\comp\lambda #1 \comp.; #2}{#1 \comp, #2}
2877 %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp.; #2}{#1 \comp, #2}
2878
2879 % function/operator application
2880 \symdecl[args=ia]{apply}
2881 \notation[prec=0;0x\neginfprec,parens]{apply}{#1 \comp( #2 \comp )}{#1 \comp, #2}
2882 \notation[prec=0;0x\neginfprec,lambdas]{apply}{#1 \; #2 }{#1 \; #2}
2883
2884 % ‘‘type’’ of all collections (sets, classes, types, kinds)
2885 \symdecl{collection}
2886 \notation[U]{collection}{\comp{\mathcal{U}}}
2887 \notation[set]{collection}{\comp{\textsf{Set}}}
2888
2889 % sequences
2890 \symdecl[args=1]{seqtype}
2891 \notation[kleene]{seqtype}{#1^{\comp\ast}}
2892
2893 \symdef[args=2,li]{sequence-index}{#1_{#2}}
2894 \notation[ui]{sequence-index}{#1^{\comp\ast}}
2895
2896 %\symdef[args=3,li]{sequence-from-to}{#1_{#2}\comp{\,\ellipses\,}#1_{#3}}
2897 %\notation[ui]{sequence-from-to}{#1^{\comp\ast}_{#2}\comp{\,\ellipses\,}#1^{\comp\ast}_{#3}}
2898 % ^ superceded by \aseqfromto and \livar/\uivar
2899

```



```

2900 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{,\ellipses}}{#1\comp,#2}
2901 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{,\ellipses\comp,}#2 }{#1\comp,#2}
2902
2903 % letin (‘‘let’’, local definitions, variable substitution)
2904 \symdecl[args=bii]{letin}
2905 \notation[let]{letin}{\comp{{\rm let}}\;#1\comp{=}#2\;\comp{{\rm in}}\;#3}
2906 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
2907 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
2908
2909 % structures
2910 \symdecl*[args=1]{module-type}
2911 \notation{module-type}{\mathtt{MOD} #1}
2912 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
2913 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{#1 \comp, #2}
2914
2915 \STEXexport{
2916   \let\nappa\apply
2917   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
2918   \def\livar{\csname sequence-index\endcsname[li]}
2919   \def\uivar{\csname sequence-index\endcsname[ui]}
2920   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
2921   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
2922 }
2923
2924 \end{@module}
2925 \ExplSyntaxOff
2926 </metatheory>

```

4.13 Auxiliary Packages

4.13.1 tikzinput

```

2927 <*tikzinput>
2928 <@=tikzinput>
2929 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
2930 \RequirePackage{l3keys2e}
2931
2932 \keys_define:nn { tikzinput } {
2933   image .bool_set:N = \c_tikzinput_image_bool
2934 }
2935
2936 \ProcessKeysOptions { tikzinput }
2937
2938 \bool_if:NTF \c_tikzinput_image_bool {
2939   \RequirePackage{graphicx}
2940
2941   \providecommand\usetikzlibrary[]{}
2942   \newcommand\tikzinput[2] []{\includegraphics[#1]{#2}}
2943 }{
2944   \RequirePackage{tikz}
2945   \RequirePackage{standalone}
2946
2947   \newcommand \tikzinput [2] [] {
2948     \setkeys{Gin}{#1}

```

```

2949 \ifx \Gin@width \Gin@exclamation
2950 \ifx \Gin@height \Gin@exclamation
2951 \input { #2 }
2952 \else
2953 \resizebox{!}{ \Gin@height }{
2954 \input { #2 }
2955 }
2956 \fi
2957 \else
2958 \ifx \Gin@height \Gin@exclamation
2959 \resizebox{ \Gin@width }{!}{
2960 \input { #2 }
2961 }
2962 \else
2963 \resizebox{ \Gin@width }{ \Gin@height }{
2964 \input { #2 }
2965 }
2966 \fi
2967 \fi
2968 }
2969 }
2970
2971 \newcommand \ctikzinput [2] [] {
2972 \begin{center}
2973 \tikzinput [#1] {#2}
2974 \end{center}
2975 }
2976
2977 \@ifpackageloaded{stex}{
2978 \RequirePackage{stex-tikzinput}
2979 }{}
2980 \</tikzinput>
2981 \< *stex-tikzinput>
2982 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
2983 \RequirePackage{stex}
2984 \RequirePackage{tikzinput}
2985
2986 % TODO
2987
2988 \</stex-tikzinput>

```

4.13.2 sTeX1 Compatibility

```

2989 \< *smglom>
2990 \RequirePackage{expl3,l3keys2e}
2991 \ProvidesExplClass{smglom}{2021/08/01}{1.9}{sTeX1 compatibility}
2992 \LoadClass[border=1px,varwidth]{standalone}
2993 \setlength\textwidth{15cm}
2994 %\g@addto@macro{\@parboxrestore}{\setlength\parskip{\baselineskip}}
2995 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
2996 \ProcessOptions
2997
2998 \RequirePackage{stex-compatibility}
2999 \</smglom>
3000

```

```

3001 <*compat>
3002 <@@=stex_deprec>
3003 \ProvidesExplPackage{stex-compatibility}{2021/08/01}{1.9}{bla}
3004 \RequirePackage[lang={de,en,ro,tr,fr}]{stex}
3005
3006 \NewDocumentEnvironment { mhmodnl } { 0{} m m } {
3007   \msg_set:nnn{stex}{warning/deprecated}{
3008     \
3009     Environment~mhmodnl~is~deprected! \
3010     Please~update~module~#2~in~file~
3011     \stex_path_to_string:N \g_stex_currentfile_seq!
3012     \
3013   }
3014   \msg_warning:nn{stex}{warning/deprecated}
3015
3016   \begin{module}[#1,lang=#3]{#2}
3017     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
3018     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
3019     \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
3020     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
3021     \input { \stex_path_to_string:N \l_tmpa_seq / \l_tmpa_str.tex }
3022   } {
3023     \end{module}
3024   }
3025
3026 \NewDocumentEnvironment { modsig } { 0{} m } {
3027   \stex_if_in_module:TF {
3028     \prop_get:NnN \l_stex_current_module_prop {name} \l_tmpa_str
3029     \str_set:Nn \l_tmpb_str { #2 }
3030     \str_if_eq:NNTF \l_tmpa_str \l_tmpb_str {
3031       \prop_set_eq:NN \l_modsig_old_module_prop \l_stex_current_module_prop
3032       \begin{@module}{modsig-#2}
3033         % \prop_set_eq:NN \l_stex_current_module_prop \l_modsig_old_module_prop
3034       } {
3035         \begin{@module}{#2}
3036       }
3037     } {
3038       \begin{@module}{#2}
3039     }
3040   }{
3041     \end{@module}
3042     \AddToHookNext { env / modsig / after }{
3043       \stex_if_in_module:T {
3044         \prop_get:NnN \l_stex_current_module_prop {name} \l_tmpa_str
3045         \str_set:Nn \l_tmpb_str { #2 }
3046         \str_if_eq:NNT \l_tmpa_str \l_tmpb_str {
3047           % \xdef \g_stex_module_after_group_tl {
3048             \stex_if_smsmode:TF {
3049               \exp_args:Nx
3050               \stex_add_to_current_module:n {
3051                 \stex_debug:n{Activating~signature~of~#2}
3052                 \exp_not:N \prop_item:cn { c_stex_module_
3053                   \prop_item:Nn \l_stex_current_module_prop {ns} ?
3054                   \prop_item:Nn \l_stex_current_module_prop {name}

```

```

3055         / modsig-#2_prop } { content }
3056     }
3057 }
3058 {
3059     \gdef \g_stex_modsig_after_group_tl {
3060         \stex_activate_module:n {
3061             \prop_item:Nn \l_stex_current_module_prop {ns} ?
3062             \prop_item:Nn \l_stex_current_module_prop {name}
3063             / modsig-#2
3064         }
3065
3066         \exp_args:Nx
3067         \stex_add_to_current_module:n {
3068             \stex_activate_module:n {
3069                 \prop_item:Nn \l_stex_current_module_prop {ns} ?
3070                 \prop_item:Nn \l_stex_current_module_prop {name}
3071                 / modsig-#2
3072             }
3073         }
3074     }
3075     \aftergroup \g_stex_modsig_after_group_tl
3076 }
3077 }
3078 }
3079 }
3080 }
3081
3082 \cs_new_protected:Npn \gimport {
3083     \peek_charcode_remove:NTF * {
3084         \gimport_do:
3085     } {
3086         \gimport_do:
3087     }
3088 }
3089
3090 \NewDocumentCommand \gimport_do: { O{} m } {
3091     \msg_set:nnn{stex}{warning/deprecated}{
3092         \\\
3093         \c_backslash_str gimport~is~deprecated! \\\
3094         Please~use~\c_backslash_str importmodule[#1]{#2}~instead!~(in~file~
3095         \stex_path_to_string:N \g_stex_currentfile_seq)
3096         \\\ \\\
3097     }
3098     \msg_warning:nn{stex}{warning/deprecated}
3099     \importmodule[#1]{#2}
3100 }
3101
3102 \cs_new_protected:Npn \guse {
3103     \peek_charcode_remove:NTF * {
3104         \guse_do:
3105     } {
3106         \guse_do:
3107     }
3108 }

```

```

3109
3110 \NewDocumentCommand \guse_do: { 0{ } m } {
3111   \msg_set:nnn{stex}{warning/deprecated}{
3112     \
3113     \c_backslash_str guse~is~deprecated! \
3114     Please~use~\c_backslash_str usemodule[#1]{#2}~instead!~(in~file~
3115     \stex_path_to_string:N \g_stex_currentfile_seq)
3116     \
3117   }
3118   \msg_warning:nn{stex}{warning/deprecated}
3119   \usemodule[#1]{#2}
3120 }
3121
3122 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3123
3124 \cs_new_protected:Npn \symi {
3125   \peek_charcode_remove:NTF * {
3126     \symi_do:
3127   } {
3128     \symi_do:
3129   }
3130 }
3131
3132 \NewDocumentCommand \symi_do: { 0{ } m } {
3133   \msg_set:nnn{stex}{warning/deprecated}{
3134     \
3135     \c_backslash_str symi~is~deprecated! \
3136     Please~use~\c_backslash_str symdecl[#1]{#2}~instead!~(in~file~
3137     \stex_path_to_string:N \g_stex_currentfile_seq)
3138     \
3139   }
3140   \msg_warning:nn{stex}{warning/deprecated}
3141   \symdecl*{#1}{#2}
3142 }
3143
3144 \cs_new_protected:Npn \symii {
3145   \peek_charcode_remove:NTF * {
3146     \symii_do:
3147   } {
3148     \symii_do:
3149   }
3150 }
3151
3152 \NewDocumentCommand \symii_do: { 0{ } m m } {
3153   \msg_set:nnn{stex}{warning/deprecated}{
3154     \
3155     \c_backslash_str symii~is~deprecated! \
3156     Please~use~\c_backslash_str symdecl[#1]{#2-#3}~instead!~(in~file~
3157     \stex_path_to_string:N \g_stex_currentfile_seq)
3158     \
3159   }
3160   \msg_warning:nn{stex}{warning/deprecated}
3161   \symdecl*{#1}{#2-#3}
3162 }

```

```

3163
3164 \cs_new_protected:Npn \symiii {
3165   \peek_charcode_remove:NTF * {
3166     \symiii_do:
3167   } {
3168     \symiii_do:
3169   }
3170 }
3171
3172 \NewDocumentCommand \symiii_do: { 0{} m m m } {
3173   \msg_set:nnn{stex}{warning/deprecated}{
3174     \\\
3175     \c_backslash_str symiii-is~deprecated! \\\
3176     Please~use~\c_backslash_str symdecl[#1]{#2-#3-#4}~instead!~(in~file~
3177     \stex_path_to_string:N \g_stex_currentfile_seq)
3178     \\\ \\\
3179   }
3180   \msg_warning:nn{stex}{warning/deprecated}
3181   \symdecl*{#1}{#2-#3-#4}
3182 }
3183
3184 \keys_define:nn { stex / deprec / defi } {
3185   name .tl_set_x:N = \l_tmpa_str
3186 }
3187
3188 \cs_new_protected:Npn \defi {
3189   \peek_charcode_remove:NTF * {
3190     \defi_do:
3191   } {
3192     \defi_do:
3193   }
3194 }
3195
3196 \NewDocumentCommand \defi_do: { 0{} m } {
3197   \str_clear:N \l_tmpa_str
3198   \keys_set:nn { stex / deprec / defi } { #1 }
3199
3200   \str_if_empty:NTF \l_tmpa_str {
3201     \msg_set:nnn{stex}{warning/deprecated}{
3202       \\\
3203       \c_backslash_str defi-is~deprecated! \\\
3204       Please~use~\c_backslash_str STExsymbol{#2}![#2]~instead!~(in~file~
3205       \stex_path_to_string:N \g_stex_currentfile_seq)
3206       \\\ \\\
3207     }
3208     \msg_warning:nn{stex}{warning/deprecated}
3209     \STExsymbol { #2 }![ \comp{#2} ]
3210   } {
3211     \msg_set:nnn{stex}{warning/deprecated}{
3212       \\\
3213       \c_backslash_str defi-is~deprecated! \\\
3214       Please~use~\c_backslash_str STExsymbol { \l_tmpa_str }[ #2 ]~instead!~(in~file~
3215       \stex_path_to_string:N \g_stex_currentfile_seq)
3216       \\\ \\\

```

```

3217     }
3218     \msg_warning:nn{stex}{warning/deprecated}
3219     \exp_args:No \STEXsymbol { \l_tmpa_str }![ \comp{#2} ]
3220   }
3221 }
3222
3223
3224 \cs_new_protected:Npn \Defi {
3225   \peek_charcode_remove:NTF * {
3226     \Defi_do:
3227   } {
3228     \Defi_do:
3229   }
3230 }
3231
3232 \NewDocumentCommand \Defi_do: { 0{} m } {
3233   \str_clear:N \l_tmpa_str
3234   \keys_set:nn { stex / deprec / defi } { #1 }
3235
3236   \str_if_empty:NTF \l_tmpa_str {
3237     \msg_set:nnn{stex}{warning/deprecated}{
3238       \l_
3239       \c_backslash_str Defi-is~deprecated! \l_
3240       Please~use~\c_backslash_str STEXsymbol{#2}![ \exp_after:wN \stex_capitalize:n #2]~inste
3241       \stex_path_to_string:N \g_stex_currentfile_seq)
3242     \l_ \l_
3243   }
3244   \msg_warning:nn{stex}{warning/deprecated}
3245   \STEXsymbol { #2 }![ \comp{\exp_after:wN \stex_capitalize:n #2} ]
3246 } {
3247   \msg_set:nnn{stex}{warning/deprecated}{
3248     \l_
3249     \c_backslash_str Defi-is~deprecated! \l_
3250     Please~use~\c_backslash_str STEXsymbol { \l_tmpa_str }[ \exp_after:wN \stex_capitalize
3251     \stex_path_to_string:N \g_stex_currentfile_seq)
3252     \l_ \l_
3253   }
3254   \msg_warning:nn{stex}{warning/deprecated}
3255   \exp_args:No \STEXsymbol { \l_tmpa_str }![ \comp{\exp_after:wN \stex_capitalize:n #2} ]
3256 }
3257 }
3258
3259 \cs_new_protected:Npn \adefi {
3260   \peek_charcode_remove:NTF * {
3261     \adefi_do:
3262   } {
3263     \adefi_do:
3264   }
3265 }
3266
3267 \NewDocumentCommand \adefi_do: { 0{} m m } {
3268   \str_clear:N \l_tmpa_str
3269   \keys_set:nn { stex / deprec / defi } { #1 }
3270

```

```

3271 \str_if_empty:NTF \l_tmpa_str {
3272   \msg_set:nnn{stex}{warning/deprecated}{
3273     \\\
3274     \c_backslash_str adefi-is-deprecated! \\\
3275     Please~use~\c_backslash_str STEXsymbol{#3}![#2]~instead!~(in~file~
3276     \stex_path_to_string:N \g_stex_currentfile_seq)
3277     \\\
3278   }
3279   \msg_warning:nn{stex}{warning/deprecated}
3280   \STEXsymbol { #3 }![ \comp{#2} ]
3281 } {
3282   \msg_set:nnn{stex}{warning/deprecated}{
3283     \\\
3284     \c_backslash_str adefi-is-deprecated! \\\
3285     Please~use~\c_backslash_str STEXsymbol { \l_tmpa_str }[ #2 ]~instead!~(in~file~
3286     \stex_path_to_string:N \g_stex_currentfile_seq)
3287     \\\
3288   }
3289   \msg_warning:nn{stex}{warning/deprecated}
3290   \exp_args:No \STEXsymbol { \l_tmpa_str }![ \comp{#2} ]
3291 }
3292 }
3293
3294 \cs_new_protected:Npn \defis {
3295   \peek_charcode_remove:NTF * {
3296     \defis_do:
3297   } {
3298     \defis_do:
3299   }
3300 }
3301
3302 \NewDocumentCommand \defis_do: { 0{} m } {
3303   \str_clear:N \l_tmpa_str
3304   \keys_set:nn { stex / deprec / defi } { #1 }
3305
3306   \str_if_empty:NTF \l_tmpa_str {
3307     \msg_set:nnn{stex}{warning/deprecated}{
3308       \\\
3309       \c_backslash_str defis-is-deprecated! \\\
3310       Please~use~\c_backslash_str STEXsymbol{#2}![#2s]~instead!~(in~file~
3311       \stex_path_to_string:N \g_stex_currentfile_seq)
3312       \\\
3313     }
3314     \msg_warning:nn{stex}{warning/deprecated}
3315     \STEXsymbol { #2 }![ \comp{#2s} ]
3316   } {
3317     \msg_set:nnn{stex}{warning/deprecated}{
3318       \\\
3319       \c_backslash_str defis-is-deprecated! \\\
3320       Please~use~\c_backslash_str STEXsymbol { \l_tmpa_str }[ #2s ]~instead!~(in~file~
3321       \stex_path_to_string:N \g_stex_currentfile_seq)
3322       \\\
3323     }
3324     \msg_warning:nn{stex}{warning/deprecated}

```



```

3325 \exp_args:No \STEXsymbol { \l_tmpa_str }![ \comp{#2s} ]
3326 }
3327 }
3328
3329 \cs_new_protected:Npn \defii {
3330 \peek_charcode_remove:NTF * {
3331 \defii_do:
3332 } {
3333 \defii_do:
3334 }
3335 }
3336
3337 \NewDocumentCommand \defii_do: { O{} m m } {
3338 \str_clear:N \l_tmpa_str
3339 \keys_set:nn { stex / deprec / defi } { #1 }
3340 \str_if_empty:NTF \l_tmpa_str {
3341 \msg_set:nnn{stex}{warning/deprecated}{
3342 \
3343 \c_backslash_str defii-is-deprecated! \
3344 Please~use~\c_backslash_str STEXsymbol{#2~#3}![#2~#3]~instead!~(in~file~
3345 \stex_path_to_string:N \g_stex_currentfile_seq)
3346 \
3347 }
3348 \msg_warning:nn{stex}{warning/deprecated}
3349 \STEXsymbol { #2~#3 }![ \comp{#2~#3} ]
3350 } {
3351 \msg_set:nnn{stex}{warning/deprecated}{
3352 \
3353 \c_backslash_str defii-is-deprecated! \
3354 Please~use~\c_backslash_str STEXsymbol { \l_tmpa_str }[ #2~#3 ]~instead!~(in~file~
3355 \stex_path_to_string:N \g_stex_currentfile_seq)
3356 \
3357 }
3358 \msg_warning:nn{stex}{warning/deprecated}
3359 \exp_args:No \STEXsymbol { \l_tmpa_str }![ \comp{#2~#3} ]
3360 }
3361 }
3362
3363
3364 \cs_new_protected:Npn \defiis {
3365 \peek_charcode_remove:NTF * {
3366 \defiis_do:
3367 } {
3368 \defiis_do:
3369 }
3370 }
3371
3372 \NewDocumentCommand \defiis_do: { O{} m m } {
3373 \str_clear:N \l_tmpa_str
3374 \keys_set:nn { stex / deprec / defi } { #1 }
3375 \str_if_empty:NTF \l_tmpa_str {
3376 \msg_set:nnn{stex}{warning/deprecated}{
3377 \
3378 \c_backslash_str defiis-is-deprecated! \

```

```

3379     Please~use~\c_backslash_str STExsymbol{#2~#3}![#2~#3s]~instead!~(in~file~
3380     \stex_path_to_string:N \g_stex_currentfile_seq)
3381     \\\ \\\
3382   }
3383   \msg_warning:nn{stex}{warning/deprecated}
3384   \STExsymbol { #2~#3 }![ \comp{#2~#3s} ]
3385 } {
3386   \msg_set:nnn{stex}{warning/deprecated}{
3387     \\\
3388     \c_backslash_str defiii~is-deprecated! \\\
3389     Please~use~\c_backslash_str STExsymbol { \l_tmpa_str }[ #2~#3s ]~instead!~(in~file~
3390     \stex_path_to_string:N \g_stex_currentfile_seq)
3391     \\\ \\\
3392   }
3393   \msg_warning:nn{stex}{warning/deprecated}
3394   \exp_args:No \STExsymbol { \l_tmpa_str }![ \comp{#2~#3s} ]
3395 }
3396 }
3397
3398
3399 \cs_new_protected:Npn \defiii {
3400   \peek_charcode_remove:NTF * {
3401     \defiii_do:
3402   } {
3403     \defiii_do:
3404   }
3405 }
3406
3407 \NewDocumentCommand \defiii_do: { 0{} m m m } {
3408   \str_clear:N \l_tmpa_str
3409   \keys_set:nn { stex / deprec / defi } { #1 }
3410   \str_if_empty:NTF \l_tmpa_str {
3411     \msg_set:nnn{stex}{warning/deprecated}{
3412       \\\
3413       \c_backslash_str defiii~is-deprecated! \\\
3414       Please~use~\c_backslash_str STExsymbol{#2~#3~#4}![#2~#3~#4]~instead!~(in~file~
3415       \stex_path_to_string:N \g_stex_currentfile_seq)
3416       \\\ \\\
3417     }
3418     \msg_warning:nn{stex}{warning/deprecated}
3419     \STExsymbol { #2~#3~#4 }![ \comp{#2~#3~#4} ]
3420   } {
3421     \msg_set:nnn{stex}{warning/deprecated}{
3422       \\\
3423       \c_backslash_str defiii~is-deprecated! \\\
3424       Please~use~\c_backslash_str STExsymbol { \l_tmpa_str }[ #2~#3~#4 ]~instead!~(in~file~
3425       \stex_path_to_string:N \g_stex_currentfile_seq)
3426       \\\ \\\
3427     }
3428     \msg_warning:nn{stex}{warning/deprecated}
3429     \exp_args:No \STExsymbol { \l_tmpa_str }![ \comp{#2~#3~#4} ]
3430   }
3431 }
3432

```

```

3433 %\RequirePackage[hyperref]{ntheorem}
3434 %\theoremstyle{plain}
3435 %\RequirePackage{amsthm}
3436
3437 \NewDocumentEnvironment {definition} { 0{} } {
3438   \begin{STEXdefinition}{}
3439 }{
3440   \end{STEXdefinition}
3441 }
3442 \keys_define:nn { stex / omtex } {
3443   title .tl_set_x:n = \l_stex_omtext_title_str
3444 }
3445 \cs_new_protected:Nn \stex_omtext_args:n {
3446   \str_clear:N \l_stex_omtext_title_str
3447   \keys_set:nn { stex / omtex }{ #1 }
3448   \exp_args:NNo \str_set:Nn \l_stex_omtext_title_str
3449     \l_stex_omtext_title_str
3450 }
3451 \NewDocumentEnvironment {omtext} { 0{} } {
3452   \stex_omtext_args:n { #1 }
3453   \paragraph{\l_stex_omtext_title_str}
3454 }{
3455
3456 }
3457 \NewDocumentEnvironment {assertion} { 0{} } {
3458
3459 }{
3460
3461 }
3462
3463 \NewDocumentCommand \inlinedef { m } {
3464   \begin{group}
3465   \let\definiendum\__stex_deprec_definiendum:w
3466   \let\definame\__stex_deprec_definame:w
3467   #1
3468   \end{group}
3469 }
3470
3471 \NewDocumentCommand \inlineass { m } { #1 }
3472
3473 \NewDocumentCommand \trefi { 0{} m } {
3474   \str_set:Nn \l_tmpa_str { #1 }
3475   \str_if_empty:NTF \l_tmpa_str {
3476     \msg_set:nnn{stex}{warning/deprecated}{
3477       \\\
3478       \c_backslash_str trefi~is~deprecated! \\\
3479       Please~use~\c_backslash_str STEXsymbol{#2}![#2]~instead!~(in~file~
3480       \stex_path_to_string:N \g_stex_currentfile_seq)
3481       \\\
3482     }
3483     \msg_warning:nn{stex}{warning/deprecated}
3484     \STEXsymbol { #2 }![ \comp{#2} ]
3485   } {
3486     \msg_set:nnn{stex}{warning/deprecated}{

```

```

3487     \\\
3488     \c_backslash_str trefi-is-deprecated! \\\
3489     Please~use~\c_backslash_str STExsymbol { #1?#2 }[ #2 ]~instead!~(in~file~
3490     \stex_path_to_string:N \g_stex_currentfile_seq)
3491     \\\ \\\
3492   }
3493   \msg_warning:nn{stex}{warning/deprecated}
3494   \STExsymbol { #1 }![ \comp{#2} ]
3495 }
3496 }
3497
3498
3499 \NewDocumentCommand \Trefi { 0{} m } {
3500   \str_set:Nn \l_tmpa_str { #1 }
3501   \str_if_empty:NTF \l_tmpa_str {
3502     \msg_set:nnn{stex}{warning/deprecated}{
3503       \\\
3504       \c_backslash_str Trefi-is-deprecated! \\\
3505       Please~use~\c_backslash_str STExsymbol{#2}![\exp_after:wN \stex_capitalize:n #2]~inste
3506       \stex_path_to_string:N \g_stex_currentfile_seq)
3507       \\\ \\\
3508     }
3509     \msg_warning:nn{stex}{warning/deprecated}
3510     \STExsymbol { #2 }![ \comp{\exp_after:wN \stex_capitalize:n #2} ]
3511   } {
3512     \msg_set:nnn{stex}{warning/deprecated}{
3513       \\\
3514       \c_backslash_str Trefi-is-deprecated! \\\
3515       Please~use~\c_backslash_str STExsymbol { #1 }[ \exp_after:wN \stex_capitalize:n #2 ]~i
3516       \stex_path_to_string:N \g_stex_currentfile_seq)
3517       \\\ \\\
3518     }
3519     \msg_warning:nn{stex}{warning/deprecated}
3520     \STExsymbol { #1 }![ \comp{\exp_after:wN \stex_capitalize:n #2} ]
3521   }
3522 }
3523
3524 \NewDocumentCommand \trefis { 0{} m } {
3525   \str_set:Nn \l_tmpa_str { #1 }
3526   \str_if_empty:NTF \l_tmpa_str {
3527     \msg_set:nnn{stex}{warning/deprecated}{
3528       \\\
3529       \c_backslash_str trefi-is-deprecated! \\\
3530       Please~use~\c_backslash_str STExsymbol{#2}![#2s]~instead!~(in~file~
3531       \stex_path_to_string:N \g_stex_currentfile_seq)
3532       \\\ \\\
3533     }
3534     \msg_warning:nn{stex}{warning/deprecated}
3535     \STExsymbol { #2 }![ \comp{#2s} ]
3536   } {
3537     \msg_set:nnn{stex}{warning/deprecated}{
3538       \\\
3539       \c_backslash_str trefi-is-deprecated! \\\
3540       Please~use~\c_backslash_str STExsymbol { #1 }[ #2s ]~instead!~(in~file~

```

```

3541     \stex_path_to_string:N \g_stex_currentfile_seq)
3542     \\\ \\\
3543   }
3544   \msg_warning:nn{stex}{warning/deprecated}
3545   \STEXsymbol { #1 }![ \comp{#2s} ]
3546 }
3547 }
3548
3549
3550 \NewDocumentCommand \Trefis { O{} m } {
3551   \str_set:Nn \l_tmpa_str { #1 }
3552   \str_if_empty:NTF \l_tmpa_str {
3553     \msg_set:nnn{stex}{warning/deprecated}{
3554       \\\
3555       \c_backslash_str Trefis~is-deprecated! \\\
3556       Please~use~\c_backslash_str STEXsymbol{#2}![ \exp_after:wN \stex_capitalize:n #2s ]~inst
3557       \stex_path_to_string:N \g_stex_currentfile_seq)
3558       \\\ \\\
3559     }
3560     \msg_warning:nn{stex}{warning/deprecated}
3561     \STEXsymbol { #2 }![ \comp{ \exp_after:wN \stex_capitalize:n #2s } ]
3562   } {
3563     \msg_set:nnn{stex}{warning/deprecated}{
3564       \\\
3565       \c_backslash_str Trefis~is-deprecated! \\\
3566       Please~use~\c_backslash_str STEXsymbol { #1 }[ \exp_after:wN \stex_capitalize:n #2s ]~
3567       \stex_path_to_string:N \g_stex_currentfile_seq)
3568       \\\ \\\
3569     }
3570     \msg_warning:nn{stex}{warning/deprecated}
3571     \STEXsymbol { #1 }![ \comp{ \exp_after:wN \stex_capitalize:n #2s } ]
3572   }
3573 }
3574
3575 \NewDocumentCommand \trefii { O{} m m } {
3576   \str_set:Nn \l_tmpa_str { #1 }
3577   \str_if_empty:NTF \l_tmpa_str {
3578     \msg_set:nnn{stex}{warning/deprecated}{
3579       \\\
3580       \c_backslash_str trefii~is-deprecated! \\\
3581       Please~use~\c_backslash_str STEXsymbol{#2~#3}![#2~#3]~instead!~(in~file~
3582       \stex_path_to_string:N \g_stex_currentfile_seq)
3583       \\\ \\\
3584     }
3585     \msg_warning:nn{stex}{warning/deprecated}
3586     \STEXsymbol { #2~#3 }![ \comp{#2~#3} ]
3587   } {
3588     \msg_set:nnn{stex}{warning/deprecated}{
3589       \\\
3590       \c_backslash_str trefii~is-deprecated! \\\
3591       Please~use~\c_backslash_str STEXsymbol { #1 }[ #2~#3 ]~instead!~(in~file~
3592       \stex_path_to_string:N \g_stex_currentfile_seq)
3593       \\\ \\\
3594     }

```

```

3595     \msg_warning:nn{stex}{warning/deprecated}
3596     \STEXsymbol { #1 }![ \comp{#2~#3} ]
3597   }
3598 }
3599
3600 \NewDocumentCommand \trefiii { 0{ } m m m } {
3601   \str_set:Nn \l_tmpa_str { #1 }
3602   \str_if_empty:NTF \l_tmpa_str {
3603     \msg_set:nnn{stex}{warning/deprecated}{
3604       \\
3605       \c_backslash_str trefiii~is~deprecated! \\
3606       Please~use~\c_backslash_str STEXsymbol{#2~#3~#4}![#2~#3~#4]~instead!~(in~file~
3607       \stex_path_to_string:N \g_stex_currentfile_seq)
3608       \\ \\
3609     }
3610     \msg_warning:nn{stex}{warning/deprecated}
3611     \STEXsymbol { #2~#3~#4 }![ \comp{#2~#3~#4} ]
3612   } {
3613     \msg_set:nnn{stex}{warning/deprecated}{
3614       \\
3615       \c_backslash_str trefiii~is~deprecated! \\
3616       Please~use~\c_backslash_str STEXsymbol { #1 }[ #2~#3~#4 ]~instead!~(in~file~
3617       \stex_path_to_string:N \g_stex_currentfile_seq)
3618       \\ \\
3619     }
3620     \msg_warning:nn{stex}{warning/deprecated}
3621     \STEXsymbol { #1 }![ \comp{#2~#3~#4} ]
3622   }
3623 }
3624
3625
3626 \NewDocumentCommand \trefiis { 0{ } m m } {
3627   \str_set:Nn \l_tmpa_str { #1 }
3628   \str_if_empty:NTF \l_tmpa_str {
3629     \msg_set:nnn{stex}{warning/deprecated}{
3630       \\
3631       \c_backslash_str trefiis~is~deprecated! \\
3632       Please~use~\c_backslash_str STEXsymbol{#2~#3}![#2~#3s]~instead!~(in~file~
3633       \stex_path_to_string:N \g_stex_currentfile_seq)
3634       \\ \\
3635     }
3636     \msg_warning:nn{stex}{warning/deprecated}
3637     \STEXsymbol { #2~#3 }![ \comp{#2~#3s} ]
3638   } {
3639     \msg_set:nnn{stex}{warning/deprecated}{
3640       \\
3641       \c_backslash_str trefiis~is~deprecated! \\
3642       Please~use~\c_backslash_str STEXsymbol { #1 }[ #2~#3s ]~instead!~(in~file~
3643       \stex_path_to_string:N \g_stex_currentfile_seq)
3644       \\ \\
3645     }
3646     \msg_warning:nn{stex}{warning/deprecated}
3647     \STEXsymbol { #1 }![ \comp{#2~#3s} ]
3648   }

```

```

3649 }
3650
3651 \NewDocumentCommand \symvariant { O{} m O{0} m m } {
3652   \msg_set:nnn{stex}{warning/deprecated}{
3653     \\\
3654     \c_backslash_str symvariant~is~deprecated! \\\
3655     Please~use~\c_backslash_str notation[#4]{ #2 }~instead!~(in~file~
3656     \stex_path_to_string:N \g_stex_currentfile_seq)
3657     \\\ \\\
3658   }
3659   \msg_warning:nn{stex}{warning/deprecated}
3660
3661   \notation[variant=#4]{#2}{#5}
3662 }
3663
3664 \NewDocumentCommand \mixfixi { O{} m m m } {
3665   \msg_set:nnn{stex}{warning/deprecated}{
3666     \c_backslash_str mixfixi~is~fatally~deprecated!\\
3667     Symbol:~\l__stex_term_highlight_uri_str\\
3668     Current~file:~\stex_path_to_string:N \g_stex_currentfile_seq
3669   }
3670   \msg_error:nn{stex}{warning/deprecated}
3671 }
3672
3673
3674 \NewDocumentCommand \infix {} {
3675   \msg_set:nnn{stex}{warning/deprecated}{
3676     \c_backslash_str infix~is~fatally~deprecated!\\
3677     Symbol:~\l__stex_term_highlight_uri_str\\
3678     Current~file:~\stex_path_to_string:N \g_stex_currentfile_seq
3679   }
3680   \msg_error:nn{stex}{warning/deprecated}
3681 }
3682
3683 \let\iprec\infprec
3684
3685 \NewDocumentCommand \inlineex { m } {
3686   \msg_set:nnn{stex}{warning/deprecated}{
3687     \c_backslash_str inlineex~is~deprecated!\\
3688     No~replacement~exists~yet.\\
3689     Current~file:~\stex_path_to_string:N \g_stex_currentfile_seq
3690   }
3691   \msg_warning:nn{stex}{warning/deprecated}
3692   #1
3693 }
3694
3695
3696 \NewDocumentCommand \term { m } {
3697   \msg_set:nnn{stex}{warning/deprecated}{
3698     \c_backslash_str term~is~deprecated!\\
3699     No~replacement~exists~yet.\\
3700     Current~file:~\stex_path_to_string:N \g_stex_currentfile_seq
3701   }
3702   \msg_warning:nn{stex}{warning/deprecated}

```

```

3703   #1
3704 }
3705
3706
3707 \NewDocumentCommand \Definame { 0{} m } {
3708   \stex_get_symbol:n { #2 }
3709   \str_set:Nx \l_tmpa_str {
3710     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3711   }
3712   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3713   \scalatex_if:TF {
3714     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3715       \l_tmpa_str
3716     }
3717   } {
3718     \@defemph {
3719       \exp_after:wN \stex_capitalize:n \l_tmpa_str
3720     } { \l_stex_get_symbol_uri_str }
3721   }
3722 }
3723
3724 \NewDocumentCommand \Definiendum { 0{} m m } {
3725   \stex_get_symbol:n { #2 }
3726   \str_set:Nx \l_tmpa_str {
3727     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3728   }
3729   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3730   \scalatex_if:TF {
3731     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3732       \l_tmpa_str
3733     }
3734   } {
3735     \@defemph {
3736       \exp_after:wN \stex_capitalize:n \l_tmpa_str
3737     } { \l_stex_get_symbol_uri_str }
3738   }
3739 }
3740
3741 \NewDocumentCommand \Symname { 0{} m }{
3742   \stex_symname_args:n { #1 }
3743   \stex_get_symbol:n { #2 }
3744   \str_set:Nx \l_tmpa_str {
3745     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3746   }
3747   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3748   \exp_args:NNx \use:nn
3749   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }![
3750     \exp_after:wN \stex_capitalize:n \l_tmpa_str
3751     \l_stex_symname_post_str
3752   ] }
3753 }
3754
3755
3756 \seq_gput_right:Nx \g_stex_smsmode_allowedenvs_seq { \tl_to_str:n { mhmodnl } }

```



```

3757 \seq_gput_right:Nx \g_stex_smsmode_allowedenvs_seq { \tl_to_str:n { modsig } }
3758 \tl_gput_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\gimport\syms\symsii\symsiii\symsiv\
3759
3760 % omtex:
3761 \cs_new_protected:Npn \lec #1 {
3762   \strut\hfil\strut\null\hfill(#1)
3763 }
3764 \cs_new_protected:Npn \nlex #1 {
3765   \textcolor{green}{\sl #1}}
3766 }
3767
3768
3769 </compat>

```