

stex.sty: \TeX 2.0*

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

November 28, 2021

Abstract

TODO

1 Introduction

TODO

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

Contents

1	Introduction	1
2	Manual	3
2.1	Modules	3
2.2	Semantic Macros and Notations	3
2.3	Archives and Imports	7
3	Documentation	8
3.1	Utils	8
3.2	Files, Paths, URIs	9
3.3	MathHub Archives	10
3.4	The Module System	12
3.5	Symbols and Terms	20
3.6	Structural Features	25
4	Implementation	25
4.1	The \LaTeX document class	25
4.2	Preliminaries	26
4.3	Files, Paths and URIs	31
4.4	MathHub Repositories	34
4.5	Module System	39
4.6	Symbol Declarations	56
4.7	Notations	62
4.8	Terms	72
4.9	Notation Components	78
4.10	Structural Features	80
4.11	Put these somewhere	87
4.12	Metatheory	88
4.13	Auxiliary Packages	90

2 Manual

2.1 Modules

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

2.2 Semantic Macros and Notations

Semantic macros invoke a formally declared symbol.

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

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

Example 1

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

(ab)

.

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

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

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

Example 2

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

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

.

Not using an explicit option with a semantic macro yields the first declared notation, unless 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 ³

2.2.2 Precedences

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

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

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

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

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

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

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

arity > 0 have a default operator precedence of 0. If no argument precedences are explicitly provided, then by default they are equal to the operator precedence.

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

For example:

Example 10

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

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

2.3 Archives and Imports

2.3.1 Namespaces

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

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

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

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

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

2.3.2 Paths in Import-Statements

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

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

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

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

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

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

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

3 Documentation

3.1 Utils

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

<code>\stex_debug:n</code>	<code>\stex_debug:n {<message>}</code>
----------------------------	--

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

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

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

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

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

We have four macros for annotating generated HTML (via L^AT_EXML or S_CA_LT_EX) with attributes:

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

Annotates the HTML generated by `⟨content⟩` with

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

3.1.2 Languages

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

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

3.2 Files, Paths, URIs

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

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

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

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

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

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

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

Store the current working directory as path-sequence and string, respectively, and the (heuristically guessed) full path to the main file, based on the PWD and `\jobname`.

`\g_stex_currentfile_seq`

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

Test 1

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

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

3.3 MathHub Archives

`\mathhub`
`\c_stex_mathhub_seq`
`\c_stex_mathhub_str`

We determine the path to the local MathHub folder via one of three means, in order of precedence:

1. The `mathhub` package option, or
2. the `\mathhub`-macro, if it has been defined before the `\usepackage{stex}`-statement, or
3. the `MATHHUB` system variable.

In all three cases, `\c_stex_mathhub_seq` and `\c_stex_mathhub_str` are set accordingly.

`\l_stex_current_repository_prop`

Always points to the *current* MathHub repository (if we currently are in one). Has the fields **id**, **ns** (namespace), **narr** (narrative namespace; currently not in use) and **deps** (dependencies; currently not in use).

`\stex_set_current_repository:n`

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

`\stex_require_repository:n`

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

`\libinput`

`\libinput{<filename>}`

Inputs `<filename>.tex` from the `lib` folders in the current archive and the `meta-inf`-archive of the current archive group (if existent). Throws an error if no file by that name exists in either folder, includes both if both exist.

Test 2

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

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

3.4 The Module System

`\l_stex_current_module_prop`

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

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

Additionally, it stores:

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

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

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

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

`\stex_add_to_current_module:n`
`\STEXexport`

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

`\stex_add_constant_to_current_module:n`

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

`\stex_add_import_to_current_module:n`

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

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

Computes the namespace for file $\langle path \rangle$ in repository with namespace $\langle namespace \rangle$ as follows:

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

<code>\stex_modules_current_namespace:</code>

Computes the current namespace

Test 3

```

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

```

```

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

```

3.4.1 The module-environment

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

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

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

Test 4

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

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

Test 5

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

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

\l_stex_all_modules_seq

Stores full URIs for all modules currently in scope.

\STEXModule

\STEXModule {*<fragment>*}

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

`\stex_invoke_module:n`

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

Test 6

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

Module 3.2[STEXModuleTest1]

Module 3.3[STEXModuleTest2]

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

3.4.2 SMS Mode

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

`\g_stex_smsmode_allowedmacros_tl`

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

`\g_stex_smsmode_allowedmacros_escape_tl`

Macros that are executed with the category codes restored.

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

`\g_stex_smsmode_allowedenvs_seq`

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

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

Tests whether SMS mode is currently active.

`\stex_smsmode_set_codes:`

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

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

`\stex_in_smsmode:nn`

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

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

Test 7

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

3.4.3 Imports and Inheritance

`\importmodule`

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

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

Test 8

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

Module 3.5[Foo]

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

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

Module 3.6[Importtest]

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

Module 3.7[Importtest2]

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

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

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

Test 9

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

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

Module 3.8[UseTest1]

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

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

Test 10

Circular dependencies:

```

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

```

Circular dependencies:

Module 3.11[CircDep1]
»macro:->\stex_invoke_symbol:n {http://mathhub.info/tests/Foo/Bar/circular1?Circular1?fooA}<
»macro:->\stex_invoke_symbol:n {http://mathhub.info/tests/Bar/Foo/circular2?Circular2?fooB}<

<hr/> <hr/>	<hr/>
<code>\stex_import_module_uri:nn</code>	<code>\stex_import_module_uri:nn {<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.

<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 $\langle prec \rangle$.

<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) *sequence* argument (as comma-separated list of terms) of the current OMA or OMBIND, with (downwards) argument precedence $\langle prec \rangle$ and associative notation $\langle notation \rangle$.

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

<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 `\withbrackets`.

<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 `\left` and `\right` 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 $\mult{a,\plus{\frac{ab}{\frac{ac}}{}}}$
$\plus{a,\mult{b,c}}\text{ and }\mult{a,\plus{\frac{ab}{\frac{ac}}{}}}$
$\displaystyle \plus{a,\mult{b,c}}$ and
\withbrackets[]{$\displaystyle
\mult{a,\plus{\frac{ab}{\frac{ac}}{}}}$}
\end{module}
```

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

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

Test 16

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

<div> <div> Module 3.17[TextTest] </div> <div> 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 </div> </div>

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

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

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

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

<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 <i>{<symbols>}</i> (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,ltxcmds}
15 \ProvidesExplPackage{stex}{2021/08/01}{1.9}{bla}

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

```

\sTeX The sTeX logo:

```

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

```

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

Messages

```

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

```

\stex_debug:n Debug mode

```

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

```

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

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

(End definition for `\c__stex_sms_iow`.)

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

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

4.2.1 L^AT_EX_ML and S^CA_LT_EX

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

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

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

```
\if@latexml Conditionals for LATEXML:
\latexml_if_p:
\latexml_if:TF
66 \ifcsname if@latexml\endcsname\else
67   \expandafter\newif\csname if@latexml\endcsname\@latexmlfalse
68 \fi
69
70 \prg_new_conditional:Nnn \latexml_if: {p, T, F, TF} {
71   \if@latexml
72     \prg_return_true:
73   \else:
74     \prg_return_false:
75   \fi:
76 }
```

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

4.2.2 HTML Annotations

77 `<@=stex_annotate>`

`\l__stex_annotate_arg_tl`
`\c__stex_annotate_emptyarg_tl`

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

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

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

`__stex_annotate_checkempty:n`

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

(End definition for `__stex_annotate_checkempty:n`.)

`\stex_annotate:nnw`

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

```

```

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

```

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

4.2.3 Languages

```

173 <@@=stex_language>

```

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

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

```

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

```

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

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

```

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

```

4.3 Files, Paths and URIs

214 `<@@=stex_path>`

4.3.1 Generic Path Handling

We treat paths as L^AT_EX3-sequences (of the individual path segments, i.e. separated by a /-character) unix-style; i.e. a path is absolute if the sequence starts with an empty entry.

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

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

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

`\stex_path_to_string:NN`
`\stex_path_to_string:N`

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

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

`\c__stex_path_dot_str`
`\c__stex_path_up_str`

. and .., respectively.

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

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

`\stex_path_canonicalize:N`

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

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

```

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

```

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

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

```

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

```

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

4.3.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

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

```



```

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

```

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

We determine the PWD

`\c_stex_pwd_seq`
`\c_stex_pwd_str`

```

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

```

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

4.3.3 File Hooks and Tracking

```

305 <@@=stex_files>

```

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

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

`\g__stex_files_stack` keeps track of file changes

```

306 \seq_gclear_new:N\g__stex_files_stack

```

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`

```

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

```

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

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

```

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

```

```

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

```

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

4.4 MathHub Repositories

```

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

```

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

```

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

```

```

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

```

(End definition for __stex_mathhub_do_manifest:n.)

\l_stex_mathhub_manifest_file_seq

```

373 \str_new:N\l_stex_mathhub_manifest_file_seq

```

(End definition for \l_stex_mathhub_manifest_file_seq.)

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

```

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

```

(End definition for `_stex_mathhub_find_manifest:N`.)

`\c_stex_mathhub_manifest_ior` File variable used for MANIFEST-files

409 `\ior_new:N \c__stex_mathhub_manifest_ior`

(End definition for `\c_stex_mathhub_manifest_ior`.)

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

```

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

```

(End definition for `_stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

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

```

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

`\stex_require_repository:n`

```

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

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

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

```

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

`\inputref`

```

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

```

```

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

```

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

`\mhpath`

```

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

```

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

`\libinput`

```

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

```

```

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

```

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

4.5 Module System

```

557 <@@=stex_module>

```

`\l_stex_current_module_prop`

```

558 \prop_new:N \l_stex_current_module_prop

```

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

`stex_if_in_module_p:`

`stex_if_in_module:TF`

```

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

```

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

`stex_if_module_exists_p:n`

`stex_if_module_exists:nTF`

```

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

```

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

`\stex_add_to_current_module:n`

`\STEXexport`

```

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

```

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

`\stex_add_constant_to_current_module:n`

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

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

`\stex_add_import_to_current_module:n`

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

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

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

`\l_stex_modules_ns_str`

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


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

`\stex_modules_current_namespace:`

```

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

```

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

4.5.1 The module environment

`\l_stex_all_modules_seq` Stores all available modules

```

634 \seq_new:N \l_stex_all_modules_seq

```

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

`\STEXModule`

`\stex_invoke_module:n`

```

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

```

```

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

```

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

module module arguments:

```

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

```

```

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

```

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

```

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

```

```

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

```

```

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

```

```

868     }
869   }
870 }
871
872 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
873   \exp_args:Nx \STEXexport{
874     \stex_activate_module:n {\l_stex_module_meta_str}
875   }
876 }
877 % TODO: Inherit metatheory for nested modules?
878 }
879 \iffalse \end{stex_annotate_env} \fi % make syntax highlighting work again

```

(End definition for `_stex_module_begin_module:.`)

`_stex_module_end_module:` implements `\end{module}`

```

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

```

(End definition for `_stex_module_end_module:.`)

@module The core environment, with no header

```

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

```

`\stex_modules_heading:` Code for document headers

```

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

```

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

Finally:

```

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

```

4.5.2 SMS Mode

960 $\langle @@=\text{stex_smsmode} \rangle$

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

```

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

(End definition for $\backslash\text{g_stex_smsmode_allowedmacros_tl}$, $\backslash\text{g_stex_smsmode_allowedmacros_escape_tl}$, and $\backslash\text{g_stex_smsmode_allowedenvs_seq}$. These variables are documented on page 15.)

$\backslash\text{stex_if_smsmode_p}$:
 $\backslash\text{stex_if_smsmode}$: \underline{TF}

```

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

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

$\backslash\text{stex_smsmode_if_catcodes_p}$:
 $\backslash\text{stex_smsmode_if_catcodes}$: \underline{TF}

Checks whether the SMS mode category code scheme is active.

```

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

(End definition for $\backslash\text{stex_smsmode_if_catcodes}$: \underline{TF} .)

`\stex_smsmode_set_codes:`

```

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

```

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

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

```

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

```

(End definition for `__stex_smsmode_unset_codes:`.)

`\stex_in_smsmode:nn`

```

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

```

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

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

```

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

```

```

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

```

(End definition for `_stex_smsmode_cs:`.)

`_stex_smsmode_rescan_cs:` If the last token gobbled by `\stex_smsmode_cs:` happened to be a `\`, we need to rescan the cs name and reinsert it into the input stream:

```

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

```

(End definition for `_stex_smsmode_rescan_cs:`.)

`_stex_smsmode_checkbegin:n` called on `\begin`; checks whether the environment being opened is allowed in SMS mode.

```

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

```

(End definition for `_stex_smsmode_checkbegin:n`.)

`_stex_smsmode_checkend:n` called on `\end`; checks whether the environment being opened is allowed in SMS mode.

```

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

```

(End definition for `_stex_smsmode_checkend:n`.)

4.5.3 Inheritance

1126 <@@=stex_importmodule>

\stex_import_module_uri:nn

```

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

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

\l_stex_importmodule_name_str
 \l_stex_importmodule_archive_str
 \l_stex_importmodule_path_str
 \l_stex_importmodule_file_str

Store the return values of \stex_import_module_uri:nn.

```

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

(End definition for \l__stex_importmodule_name_str and others.)

\stex_import_require_module:nnnn

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

```

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

```

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

```

```

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

```

```

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

```

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

`\stex_activate_module:n`

```

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

```

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

`\importmodule`

```

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

```

```

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

```

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

`\usemodule`

```

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

```

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

`\g_stex_modules_in_file_seq` `\g_stex_module_files_prop`

```

1339 \seq_new:N \g_stex_modules_in_file_seq
1340 \prop_new:N \g_stex_module_files_prop

```

(End definition for `\g_stex_modules_in_file_seq` and `\g_stex_module_files_prop`. These variables are documented on page 19.)

4.6 Symbol Declarations

```

1341 <@@=stex_symdecl>

```

`\l_stex_all_symbols_seq` Stores all available symbols

```

1342 \seq_new:N \l_stex_all_symbols_seq

```

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

`\STEXsymbol`

```

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

```

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

`symdecl` arguments:

```

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

```



```

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

```

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

```

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

```

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

\stex_symdecl_do:n

```

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

```

```

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

```

```

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

```

```

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

```

```

1565 }
1566 }

```

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

`\stex_get_symbol:n`

```

1567 \str_new:N \l_stex_get_symbol_uri_str
1568
1569 \cs_new_protected:Nn \stex_get_symbol:n {
1570   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
1571     \__stex_symdecl_get_symbol_from_cs:n { #1 }
1572   }{
1573     % argument is a string
1574     % is it a command name?
1575     \cs_if_exist:cTF { #1 }{
1576       \cs_set_eq:Nc \l_tmpa_tl { #1 }
1577       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
1578       \str_if_empty:NNTF \l_tmpa_str {
1579         \exp_args:Nx \cs_if_eq:NNTF {
1580           \tl_head:N \l_tmpa_tl
1581         } \stex_invoke_symbol:n {
1582           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
1583         }{
1584           \__stex_symdecl_get_symbol_from_string:n { #1 }
1585         }
1586       } {
1587         \__stex_symdecl_get_symbol_from_string:n { #1 }
1588       }
1589     }{
1590       % argument is not a command name
1591       \__stex_symdecl_get_symbol_from_string:n { #1 }
1592       % \l_stex_all_symbols_seq
1593     }
1594   }
1595 }
1596
1597 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
1598   \bool_set_false:N \l_tmpa_bool
1599   \stex_if_in_module:T {
1600     \prop_get:NnN \l_stex_current_module_prop
1601     { constants } \l_tmpa_seq
1602     \exp_args:NNo \seq_if_in:NnTF \l_tmpa_seq { \l_tmpa_str } {
1603       \bool_set_true:N \l_tmpa_bool
1604       \str_set:Nx \l_stex_get_symbol_uri_str {
1605         \prop_item:Nn \l_stex_current_module_prop { ns } ?
1606         \prop_item:Nn \l_stex_current_module_prop { name } ? #1
1607       }
1608     }
1609   }
1610   \bool_if:NF \l_tmpa_bool {
1611     \tl_set:Nn \l_tmpa_tl {
1612       \msg_set:nnn{stex}{error/unknownsymbol}{
1613         No~symbol~#1~found!
1614       }
1615     }
1616   }

```

```

1615     \msg_error:nn{stex}{error/unknownsymbol}
1616   }
1617   \str_set:Nn \l_tmpa_str { #1 }
1618   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1619   \seq_map_inline:Nn \l_stex_all_symbols_seq {
1620     \str_set:Nn \l_tmpb_str { ##1 }
1621     \str_if_eq:eeT { \l_tmpa_str } {
1622       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1623     } {
1624       \seq_map_break:n {
1625         \tl_set:Nn \l_tmpa_tl {
1626           \str_set:Nn \l_stex_get_symbol_uri_str {
1627             ##1
1628           }
1629         }
1630       }
1631     }
1632   }
1633   \l_tmpa_tl
1634 }
1635 }
1636
1637 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
1638   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
1639   { \tl_tail:N \l_tmpa_tl }
1640   \tl_if_single:NTF \l_tmpa_tl {
1641     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
1642       \exp_after:wN \str_set:Nn \exp_after:wN
1643       \l_stex_get_symbol_uri_str \l_tmpa_tl
1644     }{
1645       % TODO
1646       % tail is not a single group
1647     }
1648   }{
1649     % TODO
1650     % tail is not a single group
1651   }
1652 }

```

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

4.7 Notations

```

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

```

```

1663 \cs_new_protected:Nn \__stex_notation_args:n {
1664   \str_clear:N \l__stex_notation_lang_str
1665   \str_clear:N \l__stex_notation_variant_str
1666   \str_clear:N \l__stex_notation_prec_str
1667   \tl_clear:N \l__stex_notation_op_tl
1668
1669   \keys_set:nn { stex / notation } { #1 }
1670
1671   \str_set:Nx \l__stex_notation_lang_str \l__stex_notation_lang_str
1672   \str_set:Nx \l__stex_notation_variant_str \l__stex_notation_variant_str
1673   \str_set:Nx \l__stex_notation_prec_str \l__stex_notation_prec_str
1674 }

```

\notation

```

1675 \NewDocumentCommand \notation { 0{} m } {
1676   \__stex_notation_args:n { #1 }
1677   \tl_clear:N \l_stex_symdecl_definiens_tl
1678   \stex_get_symbol:n { #2 }
1679   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
1680 }

```

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

\stex_notation_do:nn

```

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

```

```

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

```



```

1766 \cs_set:Npn \l_tmpa_str } { {
1767   \stex_term_math_omb:nnnn { #1 }
1768   { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1769   { \prop_item:Nn \l_tmpb_prop { opprec } }
1770   { \exp_not:n { #2 } }
1771 } }
1772 }{
1773 \str_if_in:NnTF \l_tmpb_str B {
1774   \exp_args:Nne \use:nn
1775   {
1776     \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1777     \cs_set:Npn \l_tmpa_str } { {
1778       \stex_term_math_omb:nnnn { #1 }
1779       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1780       { \prop_item:Nn \l_tmpb_prop { opprec } }
1781       { \exp_not:n { #2 } }
1782     } }
1783   }{
1784     \exp_args:Nne \use:nn
1785     {
1786       \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1787       \cs_set:Npn \l_tmpa_str } { {
1788         \stex_term_math_oma:nnnn { #1 }
1789         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1790         { \prop_item:Nn \l_tmpb_prop { opprec } }
1791         { \exp_not:n { #2 } }
1792       } }
1793     }
1794   }
1795
1796 \int_zero:N \l_tmpa_int
1797 \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
1798 \prop_get:NnN \l_tmpb_prop { argprecs } \l_tmpa_seq
1799 \__stex_notation_arguments:
1800 }
1801 }

```

(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

```

1802 \cs_new_protected:Nn \__stex_notation_arguments: {
1803   \int_incr:N \l_tmpa_int
1804   \str_if_empty:NnTF \l_tmpa_str {
1805     \__stex_notation_final:
1806   }{
1807     \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
1808     \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
1809     \str_if_eq:NnTF \l_tmpb_str a {
1810       \__stex_notation_argument_assoc:n
1811     }{
1812       \str_if_eq:NnTF \l_tmpb_str B {
1813         \__stex_notation_argument_assoc:n
1814       }{
1815         \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str

```

```

1816 \tl_put_right:Nx \l_tmpa_tl {
1817   { \stex_term_math_arg:nnn
1818     { \int_use:N \l_tmpa_int }
1819     { \l_tmpb_str }
1820     { ####\int_use:N \l_tmpa_int }
1821   }
1822 }
1823 \__stex_notation_arguments:
1824 }
1825 }
1826 }
1827 }

```

(End definition for __stex_notation_arguments:.)

__stex_notation_argument_assoc:n

```

1828 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
1829   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1830   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
1831   \tl_put_right:Nx \l_tmpa_tl {
1832     { \stex_term_math_assoc_arg:nnnn
1833       { \int_use:N \l_tmpa_int }
1834       { \l_tmpb_str }
1835       \exp_args:No \exp_not:n
1836       {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
1837       { ####\int_use:N \l_tmpa_int }
1838     }
1839   }
1840   \__stex_notation_arguments:
1841 }

```

(End definition for __stex_notation_argument_assoc:n.)

__stex_notation_final: Called after processing all notation arguments

```

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

```

```

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

```

```

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

```

```

1970     }
1971   }
1972   \stex_annotate_invisible:nnn { notationcomp }{}{
1973     $ \exp_args:Nno \use:nn { \use:c {
1974       stex_notation_ \prop_item:Nn \l_tmpb_prop { symbol }
1975       \c_hash_str \l__stex_notation_variant_str
1976       \c_hash_str \l__stex_notation_lang_str _cs
1977     } } { \l_tmpa_tl } $
1978   }
1979 }
1980 }
1981 }

```

(End definition for `__stex_notation_final:`)

\symdef

```

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

```

```

2020
2021 \NewDocumentCommand \symdef { 0{} m } {
2022   \_stex_notation_symdef_args:n { #1 }
2023   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2024   \stex_symdecl_do:n { #2 }
2025   \exp_args:Nx \stex_notation_do:nn {
2026     \prop_item:Nn \l_tmpa_prop { module } ?
2027     \prop_item:Nn \l_tmpa_prop { name }
2028   }
2029 }

```

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

`\stex_invoke_symbol:n` Invokes a semantic macro

```

2030 %\cs_new_protected:Nn \stex_invoke_symbol:n {
2031 % \peek_charcode_remove:NTF ! {
2032 %   \stex_term_custom:nn { #1 } { }
2033 % } {
2034 %   \if_mode_math:
2035 %     \exp_after:wN \_stex_notation_invoke_math:n
2036 %   \else:
2037 %     \exp_after:wN \_stex_notation_invoke_text:n
2038 %   \fi: { #1 }
2039 % }
2040 %}
2041
2042 \cs_new_protected:Nn \stex_invoke_symbol:n {
2043   \if_mode_math:
2044     \exp_after:wN \_stex_notation_invoke_math:n
2045   \else:
2046     \exp_after:wN \_stex_notation_invoke_text:n
2047   \fi: { #1 }
2048 }

```

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

`_stex_notation_invoke_math:n`

```

2049 \cs_new_protected:Nn \_stex_notation_invoke_math:n {
2050   \peek_charcode_remove:NTF ! {
2051     \peek_charcode:NTF [ {
2052       \_stex_notation_invoke_op:nw { #1 }
2053     }{
2054       \_stex_notation_invoke_op:nw { #1 } []
2055     }
2056   }{
2057     \peek_charcode_remove:NTF * {
2058       \_stex_notation_invoke_text:n { #1 }
2059     }{
2060       \peek_charcode:NTF [ {
2061         \_stex_notation_invoke_math:nw { #1 }
2062       }{
2063         \_stex_notation_invoke_math:nw { #1 } []
2064       }
2065     }
2066   }

```

```

2066 }
2067 }

```

(End definition for `_stex_notation_invoke_math:n`.)

`_stex_notation_invoke_op:nw`

```

2068 \cs_new_protected:Npn \_stex_notation_invoke_op:nw #1 [#2] {
2069   \_stex_notation_args:n { #2 }
2070   \cs_if_exist:cTF {
2071     stex_op_notation_ #1 \c_hash_str
2072     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str _cs
2073   }{
2074     \csname stex_op_notation_ #1 \c_hash_str
2075       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str _cs
2076     \endcsname
2077   }{
2078     % TODO throw error
2079   }
2080 }

```

(End definition for `_stex_notation_invoke_op:nw`.)

`_stex_notation_invoke_math:nw`

```

2081 \cs_new_protected:Npn \_stex_notation_invoke_math:nw #1 [#2] {
2082   \_stex_notation_args:n { #2 }
2083   \prop_set_eq:Nc \l_tmpa_prop {
2084     g_stex_symdecl_ #1 _prop
2085   }
2086   \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
2087   \seq_if_empty:NTF \l_tmpa_seq {
2088     \msg_set:nnn{stex}{error/nonotations}{
2089       Symbol~#1~used,~but~has~no~notations!
2090     }
2091     \msg_error:nn{stex}{error/nonotations}
2092   } {
2093     \seq_if_in:NxTF \l_tmpa_seq
2094     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2095       \use:c{
2096         stex_notation_ #1 \c_hash_str
2097         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2098         _cs
2099       }
2100     }{
2101       \str_if_empty:NTF \l__stex_notation_variant_str {
2102         \str_if_empty:NTF \l__stex_notation_lang_str {
2103           \seq_get_left:NN \l_tmpa_seq \l_tmpa_str
2104           \use:c{
2105             stex_notation_ #1 \c_hash_str \l_tmpa_str
2106             _cs
2107           }
2108         }{
2109           \msg_set:nnn{stex}{error/wrongnotation}{
2110             Symbol~#1~has~no~notation~
2111             \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2112           }

```

```

2113         \msg_error:nn{stex}{error/wrongnotation}
2114     }
2115     }{
2116         \msg_set:nnn{stex}{error/wrongnotation}{
2117             Symbol~#1~has~no~notation~
2118             \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2119         }
2120         \msg_error:nn{stex}{error/wrongnotation}
2121     }
2122 }
2123 }
2124 }

```

(End definition for `__stex_notation_invoke_math:nw`.)

`_stex_notation_invoke_text:n`

```

2125 \cs_new_protected:Nn \__stex_notation_invoke_text:n {
2126     \peek_charcode_remove:NTF ! {
2127         \stex_term_custom:nn { #1 } { }
2128     }{
2129         \prop_set_eq:Nc \l_tmpa_prop {
2130             g_stex_symdecl_ #1 _prop
2131         }
2132         \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2133         \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2134     }
2135 }

```

(End definition for `__stex_notation_invoke_text:n`.)

4.8 Terms

2136 `<@@=stex_term>`

Precedences:

```

\infprec
\neginfprec
\l__stex_term_downprec
2137 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2138 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2139 \int_new:N \l__stex_term_downprec
2140 \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
2141 \tl_set:Nn \l__stex_term_left_bracket_str (
2142 \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

```

2143 \cs_new_protected:Nn \_stex_term_maybe_brackets:nn {
2144     \int_compare:nNnTF { #1 } > \l__stex_term_downprec {
2145         \bool_if:NTF \l_stex_inarray_bool { #2 }{

```



```

2146     \dobrackets { #2 }
2147   }
2148   }{ #2 }
2149 }

```

(End definition for `_stex_term_maybe_brackets:nn`.)

`\dobrackets`

```

2150 %\RequirePackage{scalerel}
2151 \cs_new_protected:Npn \dobrackets #1 {
2152   %\ThisStyle{\if D\m@switch
2153   %   \exp_args:Nnx \use:nn
2154   %   { \exp_after:wN \left\l__stex_term_left_bracket_str #1 }
2155   %   { \exp_not:N\right\l__stex_term_right_bracket_str }
2156   % \else
2157   %   \exp_args:Nnx \use:nn
2158   %   { \l__stex_term_left_bracket_str #1 }
2159   %   { \l__stex_term_right_bracket_str }
2160   %\fi}
2161 }

```

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

`\withbrackets`

```

2162 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2163   \exp_args:Nnx \use:nn
2164   {
2165     \tl_set:Nx \l__stex_term_left_bracket_str { #1 }
2166     \tl_set:Nx \l__stex_term_right_bracket_str { #2 }
2167     #3
2168   }
2169   {
2170     \tl_set:Nn \exp_not:N \l__stex_term_left_bracket_str
2171     { \l__stex_term_left_bracket_str }
2172     \tl_set:Nn \exp_not:N \l__stex_term_right_bracket_str
2173     { \l__stex_term_right_bracket_str }
2174   }
2175 }

```

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

`\STEXinvisible`

```

2176 \cs_new_protected:Npn \STEXinvisible #1 {
2177   \stex_annotate_invisible:n { #1 }
2178 }

```

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

OMDOC terms:

`_stex_term_math_oms:nnnn`

```

2179 \cs_new_protected:Nn \_stex_term_oms:nnn {
2180   \stex_annotate:nnn{ OMID }{ #2 }{
2181     \stex_highlight_term:nn { #1 } { #3 }
2182   }
2183 }

```

```

2184
2185 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2186   \_stex_term_maybe_brackets:nn { #3 }{
2187     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2188   }
2189 }

```

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

`_stex_term_math_oma:nnnn`

```

2190 \cs_new_protected:Nn \_stex_term_oma:nnn {
2191   \stex_annotate:nnn{ OMA }{ #2 }{
2192     \stex_highlight_term:nn { #1 } { #3 }
2193   }
2194 }
2195
2196 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2197   \_stex_term_maybe_brackets:nn { #3 }{
2198     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2199   }
2200 }

```

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

`_stex_term_math_omb:nnnn`

```

2201 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2202   \stex_annotate:nnn{ OMBIND }{ #2 }{
2203     \stex_highlight_term:nn { #1 } { #3 }
2204   }
2205 }
2206
2207 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2208   \_stex_term_maybe_brackets:nn { #3 }{
2209     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2210   }
2211 }

```

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

`_stex_term_math_arg:nnn`

```

2212 \cs_new_protected:Nn \_stex_term_arg:nn {
2213   \stex_unhighlight_term:n {
2214     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2215   }
2216 }
2217 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
2218   \exp_args:Nnx \use:nn
2219     { \int_set:Nn \l__stex_term_downprec { #2 }
2220       \_stex_term_arg:nn { #1 }{ #3 }
2221     }
2222     { \int_set:Nn \exp_not:N \l__stex_term_downprec { \int_use:N \l__stex_term_downprec } }
2223 }

```

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

`_stex_term_math_assoc_arg:nnnn`

```

2224 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
2225   \seq_set_split:Nnn \l_tmpa_seq , { #4 }
2226   \int_compare:nNnTF { \seq_count:N \l_tmpa_seq } < 2 {
2227     \tl_set:Nn \l_tmpa_tl { #4 }
2228   }{
2229     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
2230     \seq_reverse:N \l_tmpa_seq
2231     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_tl
2232     \tl_set:No \l_tmpa_tl { \l_tmpb_tl }
2233
2234     \seq_map_inline:Nn \l_tmpa_seq {
2235       \exp_args:NNo \tl_set:No \l_tmpa_tl {
2236         \exp_args:Nno
2237         \l_tmpa_cs { ##1 } \l_tmpa_tl
2238       }
2239     }
2240
2241   }
2242   \exp_args:Nnno
2243   \_stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2244 }

```

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

`\stex_term_custom:nn`

```

2245 \cs_new_protected:Nn \stex_term_custom:nn {
2246   \str_set:Nn \l__stex_term_custom_uri { #1 }
2247   \str_set:Nn \l_tmpa_str { #2 }
2248   \tl_clear:N \l_tmpa_tl
2249   \int_zero:N \l_tmpa_int
2250   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2251   \__stex_term_custom_loop:
2252 }

```

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

`__stex_term_custom_loop:`

```

2253 \cs_new_protected:Nn \__stex_term_custom_loop: {
2254   \bool_set_false:N \l_tmpa_bool
2255   \bool_while_do:nn {
2256     \str_if_eq_p:ee X {
2257       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2258     }
2259   }{
2260     \int_incr:N \l_tmpa_int
2261   }
2262
2263   \peek_charcode:NTF [ {
2264     % notation/text component
2265     \__stex_term_custom_component:w
2266   } {
2267     \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2268       % all arguments read => finish

```

```

2269     \__stex_term_custom_final:
2270 } {
2271     % arguments missing
2272     \peek_charcode_remove:NTF * {
2273         % invisible, specific argument position or both
2274         \peek_charcode:NTF [ {
2275             % visible specific argument position
2276             \__stex_term_custom_arg:wn
2277         } {
2278             % invisible
2279             \peek_charcode_remove:NTF * {
2280                 % invisible specific argument position
2281                 \__stex_term_custom_arg_inv:wn
2282             } {
2283                 % invisible next argument
2284                 \__stex_term_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2285             }
2286         }
2287     } {
2288         % next normal argument
2289         \__stex_term_custom_arg:wn [ \l_tmpa_int + 1 ]
2290     }
2291 }
2292 }
2293 }

```

(End definition for __stex_term_custom_loop:.)

_stex_term_custom_arg_inv:wn

```

2294 \cs_new_protected:Npn \__stex_term_custom_arg_inv:wn [ #1 ] #2 {
2295     \bool_set_true:N \l_tmpa_bool
2296     \__stex_term_custom_arg:wn [ #1 ] { #2 }
2297 }

```

(End definition for _stex_term_custom_arg_inv:wn.)

__stex_term_custom_arg:wn

```

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

```

```

2316         \exp_not:n { { #2 } }
2317     }
2318 }
2319 } {
2320     \tl_put_right:Nx \l_tmpa_tl {
2321         \stex_term_arg:nn { \int_eval:n { #1 } }
2322         \exp_not:n { { #2 } }
2323     }
2324 }
2325
2326 \__stex_term_custom_loop:
2327 }

```

(End definition for __stex_term_custom_arg:wn.)

__stex_term_custom_set_X:n

```

2328 \cs_new_protected:Nn \__stex_term_custom_set_X:n {
2329     \str_set:Nx \l_tmpa_str {
2330         \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
2331         X
2332         \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
2333     }
2334 }

```

(End definition for __stex_term_custom_set_X:n.)

__stex_term_custom_component:

```

2335 \cs_new_protected:Npn \__stex_term_custom_component:w [ #1 ] {
2336     \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
2337     \__stex_term_custom_loop:
2338 }

```

(End definition for __stex_term_custom_component:.)

__stex_term_custom_final:

```

2339 \cs_new_protected:Nn \__stex_term_custom_final: {
2340     \int_compare:nNnTF \l_tmpb_int = 0 {
2341         \exp_args:Nnno \stex_term_oms:nnn
2342     }{
2343         \str_if_in:NnTF \l_tmpa_str {b} {
2344             \exp_args:Nnno \stex_term_ombind:nnn
2345         } {
2346             \exp_args:Nnno \stex_term_oma:nnn
2347         }
2348     }
2349     { \l__stex_term_custom_uri } { \l__stex_term_custom_uri } { \l_tmpa_tl }
2350 }

```

(End definition for __stex_term_custom_final:.)

\symref

\symname

```

2351 \NewDocumentCommand \symref { m m }{
2352     \STEXsymbol{#1}![#2]
2353 }
2354

```

```

2355 \keys_define:nn { stex / symname } {
2356   post      .tl_set_x:N    = \l_stex_symname_post_str
2357 }
2358
2359 \cs_new_protected:Nn \stex_symname_args:n {
2360   \str_clear:N \l_stex_symname_post_str
2361   \keys_set:nn { stex / symname } { #1 }
2362   \exp_args:NNNo \str_set:Nn \l_stex_symname_post_str
2363     \l_stex_symname_post_str
2364 }
2365
2366 \NewDocumentCommand \symname { 0{} m }{
2367   \stex_symname_args:n { #1 }
2368   \stex_get_symbol:n { #2 }
2369   \str_set:Nx \l_tmpa_str {
2370     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
2371   }
2372   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
2373   \exp_args:NNx \use:nn
2374   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }! [
2375     \l_tmpa_str \l_stex_symname_post_str
2376   ] }
2377 }

```

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

4.9 Notation Components

```

2378 <@@=stex_notationcomps>

```

`\stex_highlight_term:nn`

```

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

```

```

2403 %   } {
2404     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
2405 %   }
2406 % }
2407 }

```

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

```

\comp
\@comp
\@defemph
2408 \cs_new_protected:Npn \comp #1 {
2409   \str_if_empty:NF \l__stex_notationcomps_highlight_uri_str {
2410     \scalatex_if:TF {
2411       \stex_annotate:nnn { comp }{ \l__stex_notationcomps_highlight_uri_str }{ #1 }
2412     }{
2413       \exp_args:Nnx \@comp { #1 } { \l__stex_notationcomps_highlight_uri_str }
2414     }
2415   }
2416 }
2417
2418 \cs_new_protected:Npn \@comp #1 #2 {
2419   % \pdftooltip {
2420     \textcolor{blue}{#1}
2421   % } { #2 }
2422 }
2423
2424 \cs_new_protected:Npn \@defemph #1 #2 {
2425   % \pdftooltip {
2426     \textbf{\textcolor{magenta}{#1}}
2427   % } { #2 }
2428 }

```

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

\ellipses

```

2429 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix
\parrayline
\parraycell
2430 \bool_new:N \l_stex_inparray_bool
2431 \bool_set_false:N \l_stex_inparray_bool
2432 \NewDocumentCommand \parray { m m } {
2433   \begin{group}
2434     \bool_set_true:N \l_stex_inparray_bool
2435     \begin{array}{#1}
2436       #2
2437     \end{array}
2438   \end{group}
2439 }
2440
2441 \NewDocumentCommand \prmatrix { m } {
2442   \begin{group}
2443     \bool_set_true:N \l_stex_inparray_bool
2444     \begin{matrix}
2445       #1

```

```

2446 \end{matrix}
2447 \endgroup
2448 }
2449
2450 \def \parrayline #1 #2 {
2451   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
2452 }
2453
2454 \def \parraycell #1 {
2455   #1 \bool_if:NT \l_stex_inarray_bool {&}
2456 }

```

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

4.10 Structural Features

```

2457 <@=stex_features>

```

symboldoc

```

2458 \NewDocumentEnvironment{symboldoc}{m}{
2459   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
2460   \seq_clear:N \l_tmpb_seq
2461   \seq_map_inline:Nn \l_tmpa_seq {
2462     \stex_get_symbol:n { ##1 }
2463     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
2464       \l_stex_get_symbol_uri_str
2465     }
2466   }
2467   \par
2468   \exp_args:Nnnx
2469   \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
2470 }{
2471   \end{stex_annotate_env}
2472 }

```

STEXdefinition

```

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

```



```

2491     \l_tmpa_str
2492   }
2493 } {
2494   \@defemph {
2495     \l_tmpa_str
2496   } { \l_stex_get_symbol_uri_str }
2497 }
2498 }
2499
2500 \cs_new_protected:Nn \__stex_features_defi_begin:n {
2501   \let\definiendum\__stex_features_definiendum:w
2502   \let\definame\__stex_features_definame:w
2503   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
2504   \seq_clear:N \l_tmpb_seq
2505   \seq_map_inline:Nn \l_tmpa_seq {
2506     \stex_get_symbol:n { ##1 }
2507     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
2508       \l_stex_get_symbol_uri_str
2509     }
2510   }
2511   \exp_args:Nnnx
2512   \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
2513 }
2514
2515 \cs_new_protected:Nn \__stex_features_defi_end: {
2516   \end{stex_annotate_env}
2517 }
2518
2519 \NewDocumentEnvironment{STEXdefinition}{ m m m }{
2520   \__stex_features_defi_begin:n { #1 }
2521 }{
2522   \__stex_features_defi_end:
2523 }

```

\setSTEXdefinition

```

2524 \cs_new_protected:Npn \setSTEXdefinition #1 {
2525   \AddToHook{env/#1/before}[stex]{\__stex_features_defi_begin:n{}}
2526   \AddToHook{env/#1/after}[stex]{\__stex_features_defi_end:}
2527 }

```

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

structural@feature

```

2528
2529 \NewDocumentEnvironment{structural@feature}{ m m m }{
2530   \stex_if_in_module:F {
2531     \msg_set:nnn{stex}{error/nomodule}{
2532       Structural~Feature~has~to~occur~in~a~module:\\
2533       Feature~#2~of~type~#1\\
2534       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
2535     }
2536     \msg_error:nn{stex}{error/nomodule}
2537   }
2538 }

```

```

2539 \str_set:Nx \l_stex_module_name_str {
2540   \prop_item:Nn \l_stex_current_module_prop
2541     { name } / #2 - feature
2542 }
2543
2544
2545 \str_clear:N \l_tmpa_str
2546 \seq_clear:N \l_tmpa_seq
2547 \tl_clear:N \l_tmpa_tl
2548 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
2549   origname = #2,
2550   name      = \l_stex_module_name_str ,
2551   ns        = \l_stex_module_ns_str ,
2552   imports   = \exp_not:o { \l_tmpa_seq } ,
2553   constants = \exp_not:o { \l_tmpa_seq } ,
2554   content   = \exp_not:o { \l_tmpa_tl } ,
2555   file      = \exp_not:o { \g_stex_currentfile_seq } ,
2556   lang      = \l_stex_module_lang_str ,
2557   sig       = \l_tmpa_str ,
2558   meta      = \l_tmpa_str ,
2559   feature   = #1 ,
2560 }
2561
2562 \stex_if_smsmode:TF {
2563   \stex_smsmode_set_codes:
2564 } {
2565   \begin{stex_annotate_env}{ feature:#1 }{}
2566   \stex_annotate_invisible:nnn{header}{}{ #3 }
2567 }
2568 }{
2569   \str_set:Nx \l_tmpa_str {
2570     c_stex_feature_
2571     \prop_item:Nn \l_stex_current_module_prop { ns } ?
2572     \prop_item:Nn \l_stex_current_module_prop { name }
2573     _prop
2574   }
2575   \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
2576   \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
2577   \stex_if_smsmode:TF {
2578     \exp_args:Nx \stex_addtosms:n {
2579       \prop_gset_from_keyval:cn {
2580         c_stex_feature_
2581         \prop_item:Nn \l_stex_current_module_prop { ns } ?
2582         \prop_item:Nn \l_stex_current_module_prop { name }
2583         _prop
2584       } {
2585         origname = #2,
2586         name      = \prop_item:cn { \l_tmpa_str } { name } ,
2587         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
2588         imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
2589         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
2590         content   = \prop_item:cn { \l_tmpa_str } { content } ,
2591         file      = \prop_item:cn { \l_tmpa_str } { file } ,
2592         lang      = \prop_item:cn { \l_tmpa_str } { lang } ,

```

```

2593         sig      = \prop_item:cn { \l_tmpa_str } { sig } ,
2594         meta      = \prop_item:cn { \l_tmpa_str } { meta } ,
2595         feature    = \prop_item:cn { \l_tmpa_str } { feature }
2596     }
2597 }
2598 } {
2599     \end{stex_annotate_env}
2600 }
2601 }
2602

```

structure

```

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

```

```

2645 \exp_args:Nnx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
2646 }
2647 \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
2648 \exp_args:Nnx
2649 \AddToHookNext { env / structure / after }{
2650 \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}}{
2651 \_stex_term_math_oms:nnnn { \l_tmpa_str }{}{0}{}
2652 }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
2653 \STEXexport {
2654 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2655 {\prop_item:Nn \l_stex_current_module_prop { origname }}
2656 {\l_tmpa_str}
2657 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2658 {#2}{\l_tmpa_str}
2659 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
2660 % \prop_item:Nn \l_stex_current_module_prop { origname },
2661 % \l_tmpa_str
2662 % }
2663 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
2664 % #2,\l_tmpa_str
2665 % }
2666 % \tl_set:cx { #2 } {
2667 % \stex_invoke_structure:n { \l_tmpa_str }
2668 % }
2669 }
2670
2671 \end{structural@feature}
2672 % \g_stex_last_feature_prop
2673 }

```

\instantiate

```

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

```

```

2697         \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
2698         \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
2699         \l_tmpa_tl
2700         \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
2701             \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
2702             \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
2703         }{
2704             \tl_clear:N \l_tmpb_tl
2705         }
2706     }
2707 }{
2708     \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
2709     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
2710         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
2711         \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
2712         \tl_clear:N \l_tmpa_tl
2713     }{
2714         % TODO throw error
2715     }
2716 }
2717 % \l_tmpa_str: name
2718 % \l_tmpa_tl: definiens
2719 % \l_tmpb_tl: notation
2720 \tl_if_empty:NT \l__stex_features_structure_field_str {
2721     % TODO throw error
2722 }
2723 \str_clear:N \l_tmpb_str
2724
2725 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
2726 \seq_map_inline:Nn \l_tmpa_seq {
2727     \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
2728     \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
2729     \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
2730         \seq_map_break:n {
2731             \str_set:Nn \l_tmpb_str { ####1 }
2732         }
2733     }
2734 }
2735 \prop_get:cnN { g_stex_symdecl_ \l_tmpb_str _prop } {args}
2736 \l_tmpb_str
2737
2738 \tl_if_empty:NTF \l_tmpb_tl {
2739     \tl_if_empty:NF \l_tmpa_tl {
2740         \exp_args:Nx \use:n {
2741             \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fe
2742         }
2743     }
2744 }{
2745     \tl_if_empty:NTF \l_tmpa_tl {
2746         \exp_args:Nx \use:n {
2747             \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\
2748         }
2749     }
2750 }{

```

```

2751         \exp_args:Nx \use:n {
2752             \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
2753             \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
2754         }
2755     }
2756 }
2757 % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
2758 % \prop_item:Nn \l_stex_current_module_prop {name} ?
2759 % #3/\l__stex_features_structure_field_str
2760 % \par
2761 % \expandafter\present\csname
2762 %     g_stex_symdecl_
2763 % \prop_item:Nn \l_stex_current_module_prop {ns} ?
2764 % \prop_item:Nn \l_stex_current_module_prop {name} ?
2765 % #3/\l__stex_features_structure_field_str
2766 % _prop
2767 % \endcsname
2768 }
2769
2770 \tl_clear:N \l__stex_features_structure_def_tl
2771
2772 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
2773 \seq_map_inline:Nn \l_tmpa_seq {
2774     \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
2775     \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
2776     \exp_args:Nx \use:n {
2777         \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
2778
2779         }
2780     }
2781
2782     \prop_if_exist:cF {
2783         g_stex_symdecl_
2784         \prop_item:Nn \l_stex_current_module_prop {ns} ?
2785         \prop_item:Nn \l_stex_current_module_prop {name} ?
2786         #3/\l_tmpa_str
2787         _prop
2788     }{
2789         \prop_get:cnN { g_stex_symdecl_ ##1 _prop } {args}
2790         \l_tmpb_str
2791         \exp_args:Nx \use:n {
2792             \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
2793         }
2794     }
2795 }
2796
2797 \symdecl*[type={\STEXsymbol{module-type}}{
2798     \_stex_term_math_oms:nnnn {
2799         \prop_item:Nn \l__stex_features_structure_prop {ns} ?
2800         \prop_item:Nn \l__stex_features_structure_prop {name}
2801         }{}{0}{}
2802     }{}{#3}
2803
2804 % TODO: -> sms file

```

```

2805
2806 \tl_set:cx{ #3 }{
2807   \stex_invoke_structure:nnn {
2808     \prop_item:Nn \l_stex_current_module_prop {ns} ?
2809     \prop_item:Nn \l_stex_current_module_prop {name} ? #3
2810   } {
2811     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
2812     \prop_item:Nn \l__stex_features_structure_prop {name}
2813   }
2814 }
2815
2816 }

```

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

\stex_invoke_structure:nnn

```

2817 % #1: URI of the instance
2818 % #2: URI of the instantiated module
2819 \cs_new_protected:Nn \stex_invoke_structure:nnn {
2820   \tl_if_empty:nTF{ #3 }{
2821     \prop_set_eq:Nc \l__stex_features_structure_prop {
2822       c_stex_feature_ #2 _prop
2823     }
2824     \tl_clear:N \l_tmpa_tl
2825     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
2826     \seq_map_inline:Nn \l_tmpa_seq {
2827       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
2828       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
2829       \cs_if_exist:cT {
2830         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
2831       }{
2832         \tl_if_empty:NF \l_tmpa_tl {
2833           \tl_put_right:Nn \l_tmpa_tl {,}
2834         }
2835         \tl_put_right:Nx \l_tmpa_tl {
2836           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
2837         }
2838       }
2839     }
2840     \scalatexBREAK
2841     \exp_args:No \mathstrut \l_tmpa_tl
2842   }{
2843     \stex_invoke_symbol:n{#1/#3}
2844   }
2845 }

```

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

4.11 Put these somewhere

\MSC

```

2846 \NewDocumentCommand \MSC {m} {
2847   % TODO
2848 }

```



```

2889 %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
2890 %\notation[lambda]{mapto}{\comp\lambda #1 \comp.\; #2}{#1 \comp, #2}
2891 %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp.\; #2}{#1 \comp, #2}
2892
2893 % function/operator application
2894 \symdecl[args=ia]{apply}
2895 \notation[prec=0;0x\neginfprec,parens]{apply}{#1 \comp( #2 \comp)}{#1 \comp, #2}
2896 \notation[prec=0;0x\neginfprec,lambda]{apply}{#1 \; #2 }{#1 \; #2}
2897
2898 % ‘‘type’’ of all collections (sets, classes, types, kinds)
2899 \symdecl{collection}
2900 \notation[U]{collection}{\comp{\mathcal{U}}}
2901 \notation[set]{collection}{\comp{\textsf{Set}}}
2902
2903 % sequences
2904 \symdecl[args=1]{seqtype}
2905 \notation[kleene]{seqtype}{#1^{\comp\ast}}
2906
2907 \symdef[args=2,li]{sequence-index}{#1_{#2}}
2908 \notation[ui]{sequence-index}{#1^{#2}}
2909
2910 %\symdef[args=3,li]{sequence-from-to}{#1_{#2}\comp{\,\ellipses},#1_{#3}}
2911 %\notation[ui]{sequence-from-to}{#1^{#2}\comp{\,\ellipses},#1^{#3}}
2912 % ^ superceded by \aseqfromto and \livar/\uivar
2913
2914 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{\,\ellipses}}{#1\comp,#2}
2915 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{\,\ellipses\comp,}#2 }{#1\comp,#2}
2916
2917 % letin (‘‘let’’, local definitions, variable substitution)
2918 \symdecl[args=bii]{letin}
2919 \notation[let]{letin}{\comp{\rm let}}\;#1\comp{=}\#2\;\comp{\rm in}}\;#3}
2920 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
2921 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
2922
2923 % structures
2924 \symdecl*[args=1]{module-type}
2925 \notation{module-type}{\mathtt{MOD} #1}
2926 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
2927 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{#1 \comp, #2}
2928
2929 \STEXexport{
2930   \let\nappa\apply
2931   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
2932   \def\livar{\csname sequence-index\endcsname[li]}
2933   \def\uivar{\csname sequence-index\endcsname[ui]}
2934   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
2935   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
2936 }
2937
2938 \end{@module}
2939 \ExplSyntaxOff
2940 </metatheory>

```

4.13 Auxiliary Packages

4.13.1 tikzinput

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

```

2992 \RequirePackage{stex-tikzinput}
2993 }{}
2994 </tikzinput>
2995 <*stex-tikzinput>
2996 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
2997 \RequirePackage{stex}
2998 \RequirePackage{tikzinput}
2999
3000 % TODO
3001
3002 </stex-tikzinput>

```

4.13.2 sTeX1 Compatibility

```

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

```

```

3044 \str_if_eq:NNTF \l_tmpa_str \l_tmpb_str {
3045 \prop_set_eq:NN \l_modsig_old_module_prop \l_stex_current_module_prop
3046 \begin{@module}{modsig-#2}
3047 % \prop_set_eq:NN \l_stex_current_module_prop \l_modsig_old_module_prop
3048 } {
3049 \begin{@module}{#2}
3050 }
3051 } {
3052 \begin{@module}{#2}
3053 }
3054 }{
3055 \end{@module}
3056 \AddToHookNext { env / modsig / after }{
3057 \stex_if_in_module:T {
3058 \prop_get:NnN \l_stex_current_module_prop {name} \l_tmpa_str
3059 \str_set:Nn \l_tmpb_str { #2 }
3060 \str_if_eq:NNT \l_tmpa_str \l_tmpb_str {
3061 % \xdef \g_stex_module_after_group_tl {
3062 \stex_if_smsmode:TF {
3063 \exp_args:Nx
3064 \stex_add_to_current_module:n {
3065 \stex_debug:n{Activating~signature~of~#2}
3066 \exp_not:N \prop_item:cn { c_stex_module_
3067 \prop_item:Nn \l_stex_current_module_prop {ns} ?
3068 \prop_item:Nn \l_stex_current_module_prop {name}
3069 / modsig-#2_prop } { content }
3070 }
3071 }
3072 {
3073 \gdef \g_stex_modsig_after_group_tl {
3074 \stex_activate_module:n {
3075 \prop_item:Nn \l_stex_current_module_prop {ns} ?
3076 \prop_item:Nn \l_stex_current_module_prop {name}
3077 / modsig-#2
3078 }
3079
3080 \exp_args:Nx
3081 \stex_add_to_current_module:n {
3082 \stex_activate_module:n {
3083 \prop_item:Nn \l_stex_current_module_prop {ns} ?
3084 \prop_item:Nn \l_stex_current_module_prop {name}
3085 / modsig-#2
3086 }
3087 }
3088 }
3089 \aftergroup \g_stex_modsig_after_group_tl
3090 }
3091 }
3092 }
3093 }
3094 }
3095
3096 \cs_new_protected:Npn \gimport {
3097 \peek_charcode_remove:NTF * {

```

```

3098     \gimport_do:
3099   } {
3100     \gimport_do:
3101   }
3102 }
3103
3104 \NewDocumentCommand \gimport_do: { 0{} m } {
3105   \msg_set:nnn{stex}{warning/deprecated}{
3106     \\\
3107     \c_backslash_str gimport~is~deprecated! \\\
3108     Please~use~\c_backslash_str importmodule[#1]{#2}~instead!~(in~file~
3109     \stex_path_to_string:N \g_stex_currentfile_seq)
3110     \\\ \\\
3111   }
3112   \msg_warning:nn{stex}{warning/deprecated}
3113   \importmodule[#1]{#2}
3114 }
3115
3116 \cs_new_protected:Npn \guse {
3117   \peek_charcode_remove:NTF * {
3118     \guse_do:
3119   } {
3120     \guse_do:
3121   }
3122 }
3123
3124 \NewDocumentCommand \guse_do: { 0{} m } {
3125   \msg_set:nnn{stex}{warning/deprecated}{
3126     \\\
3127     \c_backslash_str guse~is~deprecated! \\\
3128     Please~use~\c_backslash_str usemodule[#1]{#2}~instead!~(in~file~
3129     \stex_path_to_string:N \g_stex_currentfile_seq)
3130     \\\ \\\
3131   }
3132   \msg_warning:nn{stex}{warning/deprecated}
3133   \usemodule[#1]{#2}
3134 }
3135
3136 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3137
3138 \cs_new_protected:Npn \symi {
3139   \peek_charcode_remove:NTF * {
3140     \symi_do:
3141   } {
3142     \symi_do:
3143   }
3144 }
3145
3146 \NewDocumentCommand \symi_do: { 0{} m } {
3147   \msg_set:nnn{stex}{warning/deprecated}{
3148     \\\
3149     \c_backslash_str symi~is~deprecated! \\\
3150     Please~use~\c_backslash_str symdecl[#1]{#2}~instead!~(in~file~
3151     \stex_path_to_string:N \g_stex_currentfile_seq)

```

```

3152     \\\ \\\
3153   }
3154   \msg_warning:nn{stex}{warning/deprecated}
3155   \symdecl*{#1}{#2}
3156 }
3157
3158 \cs_new_protected:Npn \symii {
3159   \peek_charcode_remove:NTF * {
3160     \symii_do:
3161   } {
3162     \symii_do:
3163   }
3164 }
3165
3166 \NewDocumentCommand \symii_do: { 0{} m m } {
3167   \msg_set:nnn{stex}{warning/deprecated}{
3168     \\\
3169     \c_backslash_str symii~is~deprecated! \\\
3170     Please~use~\c_backslash_str symdecl{#1}{#2-#3}~instead!~(in~file~
3171     \stex_path_to_string:N \g_stex_currentfile_seq)
3172     \\\ \\\
3173   }
3174   \msg_warning:nn{stex}{warning/deprecated}
3175   \symdecl*{#1}{#2-#3}
3176 }
3177
3178 \cs_new_protected:Npn \symiii {
3179   \peek_charcode_remove:NTF * {
3180     \symiii_do:
3181   } {
3182     \symiii_do:
3183   }
3184 }
3185
3186 \NewDocumentCommand \symiii_do: { 0{} m m m } {
3187   \msg_set:nnn{stex}{warning/deprecated}{
3188     \\\
3189     \c_backslash_str symiii~is~deprecated! \\\
3190     Please~use~\c_backslash_str symdecl{#1}{#2-#3-#4}~instead!~(in~file~
3191     \stex_path_to_string:N \g_stex_currentfile_seq)
3192     \\\ \\\
3193   }
3194   \msg_warning:nn{stex}{warning/deprecated}
3195   \symdecl*{#1}{#2-#3-#4}
3196 }
3197
3198 \keys_define:nn { stex / deprec / defi } {
3199   name .tl_set_x:N = \l_tmpa_str
3200 }
3201
3202 \cs_new_protected:Npn \defi {
3203   \peek_charcode_remove:NTF * {
3204     \defi_do:
3205   } {

```

```

3206     \defi_do:
3207   }
3208 }
3209
3210 \NewDocumentCommand \defi_do: { 0{} m } {
3211   \str_clear:N \l_tmpa_str
3212   \keys_set:nn { stex / deprec / defi } { #1 }
3213
3214   \str_if_empty:NTF \l_tmpa_str {
3215     \msg_set:nnn{stex}{warning/deprecated}{
3216       \\\
3217       \c_backslash_str defi-is~deprecated! \\\
3218       Please~use~\c_backslash_str STEXsymbol{#2}![#2]~instead!~(in~file~
3219       \stex_path_to_string:N \g_stex_currentfile_seq)
3220       \\\ \\\
3221     }
3222     \msg_warning:nn{stex}{warning/deprecated}
3223     \STEXsymbol { #2 }![ \comp{#2} ]
3224   } {
3225     \msg_set:nnn{stex}{warning/deprecated}{
3226       \\\
3227       \c_backslash_str defi-is~deprecated! \\\
3228       Please~use~\c_backslash_str STEXsymbol { \l_tmpa_str }[ #2 ]~instead!~(in~file~
3229       \stex_path_to_string:N \g_stex_currentfile_seq)
3230       \\\ \\\
3231     }
3232     \msg_warning:nn{stex}{warning/deprecated}
3233     \exp_args:No \STEXsymbol { \l_tmpa_str }![ \comp{#2} ]
3234   }
3235 }
3236
3237
3238 \cs_new_protected:Npn \Defi {
3239   \peek_charcode_remove:NTF * {
3240     \Defi_do:
3241   } {
3242     \Defi_do:
3243   }
3244 }
3245
3246 \NewDocumentCommand \Defi_do: { 0{} m } {
3247   \str_clear:N \l_tmpa_str
3248   \keys_set:nn { stex / deprec / defi } { #1 }
3249
3250   \str_if_empty:NTF \l_tmpa_str {
3251     \msg_set:nnn{stex}{warning/deprecated}{
3252       \\\
3253       \c_backslash_str Defi-is~deprecated! \\\
3254       Please~use~\c_backslash_str STEXsymbol{#2}![\exp_after:wN \stex_capitalize:n #2]~inste
3255       \stex_path_to_string:N \g_stex_currentfile_seq)
3256       \\\ \\\
3257     }
3258     \msg_warning:nn{stex}{warning/deprecated}
3259     \STEXsymbol { #2 }![ \comp{\exp_after:wN \stex_capitalize:n #2} ]

```

```

3260 } {
3261   \msg_set:nnn{stex}{warning/deprecated}{
3262     \\\
3263     \c_backslash_str Defi~is~deprecated! \\\
3264     Please~use~\c_backslash_str STExsymbol { \l_tmpa_str }[ \exp_after:wN \stex_capitalize
3265     \stex_path_to_string:N \g_stex_currentfile_seq)
3266     \\\ \\\
3267   }
3268   \msg_warning:nn{stex}{warning/deprecated}
3269   \exp_args:No \STExsymbol { \l_tmpa_str }![ \comp{\exp_after:wN \stex_capitalize:n #2} ]
3270 }
3271 }
3272
3273 \cs_new_protected:Npn \adefi {
3274   \peek_charcode_remove:NTF * {
3275     \adefi_do:
3276   } {
3277     \adefi_do:
3278   }
3279 }
3280
3281 \NewDocumentCommand \adefi_do: { 0{} m m } {
3282   \str_clear:N \l_tmpa_str
3283   \keys_set:nn { stex / deprec / defi } { #1 }
3284
3285   \str_if_empty:NTF \l_tmpa_str {
3286     \msg_set:nnn{stex}{warning/deprecated}{
3287       \\\
3288       \c_backslash_str adefi~is~deprecated! \\\
3289       Please~use~\c_backslash_str STExsymbol{#3}![#2]~instead!~(in~file~
3290       \stex_path_to_string:N \g_stex_currentfile_seq)
3291       \\\ \\\
3292     }
3293     \msg_warning:nn{stex}{warning/deprecated}
3294     \STExsymbol { #3 }![ \comp{#2} ]
3295   } {
3296     \msg_set:nnn{stex}{warning/deprecated}{
3297       \\\
3298       \c_backslash_str adefi~is~deprecated! \\\
3299       Please~use~\c_backslash_str STExsymbol { \l_tmpa_str }[ #2 ]~instead!~(in~file~
3300       \stex_path_to_string:N \g_stex_currentfile_seq)
3301       \\\ \\\
3302     }
3303     \msg_warning:nn{stex}{warning/deprecated}
3304     \exp_args:No \STExsymbol { \l_tmpa_str }![ \comp{#2} ]
3305   }
3306 }
3307
3308 \cs_new_protected:Npn \defis {
3309   \peek_charcode_remove:NTF * {
3310     \defis_do:
3311   } {
3312     \defis_do:
3313   }

```



```

3314 }
3315
3316 \NewDocumentCommand \defis_do: { 0{} m } {
3317   \str_clear:N \l_tmpa_str
3318   \keys_set:nn { stex / deprec / defi } { #1 }
3319
3320   \str_if_empty:NTF \l_tmpa_str {
3321     \msg_set:nnn{stex}{warning/deprecated}{
3322       \\\
3323       \c_backslash_str defis-is-deprecated! \\\
3324       Please~use~\c_backslash_str STEXsymbol{#2}![#2s]~instead!~(in~file~
3325       \stex_path_to_string:N \g_stex_currentfile_seq)
3326       \\\
3327     }
3328     \msg_warning:nn{stex}{warning/deprecated}
3329     \STEXsymbol { #2 }![ \comp{#2s} ]
3330   } {
3331     \msg_set:nnn{stex}{warning/deprecated}{
3332       \\\
3333       \c_backslash_str defis-is-deprecated! \\\
3334       Please~use~\c_backslash_str STEXsymbol { \l_tmpa_str }[ #2s ]~instead!~(in~file~
3335       \stex_path_to_string:N \g_stex_currentfile_seq)
3336       \\\
3337     }
3338     \msg_warning:nn{stex}{warning/deprecated}
3339     \exp_args:No \STEXsymbol { \l_tmpa_str }![ \comp{#2s} ]
3340   }
3341 }
3342
3343 \cs_new_protected:Npn \defii {
3344   \peek_charcode_remove:NTF * {
3345     \defii_do:
3346   } {
3347     \defii_do:
3348   }
3349 }
3350
3351 \NewDocumentCommand \defii_do: { 0{} m m } {
3352   \str_clear:N \l_tmpa_str
3353   \keys_set:nn { stex / deprec / defi } { #1 }
3354   \str_if_empty:NTF \l_tmpa_str {
3355     \msg_set:nnn{stex}{warning/deprecated}{
3356       \\\
3357       \c_backslash_str defii-is-deprecated! \\\
3358       Please~use~\c_backslash_str STEXsymbol{#2-#3}![#2~#3]~instead!~(in~file~
3359       \stex_path_to_string:N \g_stex_currentfile_seq)
3360       \\\
3361     }
3362     \msg_warning:nn{stex}{warning/deprecated}
3363     \STEXsymbol { #2-#3 }![ \comp{#2~#3} ]
3364   } {
3365     \msg_set:nnn{stex}{warning/deprecated}{
3366       \\\
3367       \c_backslash_str defii-is-deprecated! \\\

```

```

3368     Please~use~\c_backslash_str STExsymbol { \l_tmpa_str }[ #2~#3 ]~instead!~(in~file~
3369     \stex_path_to_string:N \g_stex_currentfile_seq)
3370     \\\ \\\
3371   }
3372   \msg_warning:nn{stex}{warning/deprecated}
3373   \exp_args:No \STExsymbol { \l_tmpa_str }![ \comp{#2~#3} ]
3374 }
3375 }
3376
3377
3378 \cs_new_protected:Npn \defiis {
3379   \peek_charcode_remove:NTF * {
3380     \defiis_do:
3381   } {
3382     \defiis_do:
3383   }
3384 }
3385
3386 \NewDocumentCommand \defiis_do: { O{} m m } {
3387   \str_clear:N \l_tmpa_str
3388   \keys_set:nn { stex / deprec / defi } { #1 }
3389   \str_if_empty:NTF \l_tmpa_str {
3390     \msg_set:nnn{stex}{warning/deprecated}{
3391       \\\
3392       \c_backslash_str defiis~is~deprecated! \\\
3393       Please~use~\c_backslash_str STExsymbol{#2~#3}![#2~#3s]~instead!~(in~file~
3394       \stex_path_to_string:N \g_stex_currentfile_seq)
3395       \\\ \\\
3396     }
3397     \msg_warning:nn{stex}{warning/deprecated}
3398     \STExsymbol { #2~#3 }![ \comp{#2~#3s} ]
3399   } {
3400     \msg_set:nnn{stex}{warning/deprecated}{
3401       \\\
3402       \c_backslash_str defiis~is~deprecated! \\\
3403       Please~use~\c_backslash_str STExsymbol { \l_tmpa_str }[ #2~#3s ]~instead!~(in~file~
3404       \stex_path_to_string:N \g_stex_currentfile_seq)
3405       \\\ \\\
3406     }
3407     \msg_warning:nn{stex}{warning/deprecated}
3408     \exp_args:No \STExsymbol { \l_tmpa_str }![ \comp{#2~#3s} ]
3409   }
3410 }
3411
3412
3413 \cs_new_protected:Npn \defiii {
3414   \peek_charcode_remove:NTF * {
3415     \defiii_do:
3416   } {
3417     \defiii_do:
3418   }
3419 }
3420
3421 \NewDocumentCommand \defiii_do: { O{} m m m } {

```

```

3422 \str_clear:N \l_tmpa_str
3423 \keys_set:nn { stex / deprec / defi } { #1 }
3424 \str_if_empty:NTF \l_tmpa_str {
3425   \msg_set:nnn{stex}{warning/deprecated}{
3426     \\\
3427     \c_backslash_str defiii~is-deprecated! \\\
3428     Please~use~\c_backslash_str STExsymbol{#2~#3~#4}![#2~#3~#4]~instead!~(in~file~
3429     \stex_path_to_string:N \g_stex_currentfile_seq)
3430     \\\ \\\
3431   }
3432   \msg_warning:nn{stex}{warning/deprecated}
3433   \STExsymbol { #2~#3~#4 }![ \comp{#2~#3~#4} ]
3434 } {
3435   \msg_set:nnn{stex}{warning/deprecated}{
3436     \\\
3437     \c_backslash_str defiii~is-deprecated! \\\
3438     Please~use~\c_backslash_str STExsymbol { \l_tmpa_str }[ #2~#3~#4 ]~instead!~(in~file~
3439     \stex_path_to_string:N \g_stex_currentfile_seq)
3440     \\\ \\\
3441   }
3442   \msg_warning:nn{stex}{warning/deprecated}
3443   \exp_args:No \STExsymbol { \l_tmpa_str }![ \comp{#2~#3~#4} ]
3444 }
3445 }
3446
3447 %\RequirePackage[hyperref]{ntheorem}
3448 %\theoremstyle{plain}
3449 %\RequirePackage{amsthm}
3450
3451 \NewDocumentEnvironment {definition} { 0{} } {
3452   \begin{STExdefinition}{}
3453 }{
3454   \end{STExdefinition}
3455 }
3456 \keys_define:nn { stex / omtex } {
3457   id .tl_set_x:N = \l_stex_omtext_id_str ,
3458   title .tl_set_x:N = \l_stex_omtext_title_str ,
3459   type .tl_set_x:N = \l_stex_omtext_type_tl ,
3460   for .tl_set_x:N = \l_stex_omtext_for_tl ,
3461   from .tl_set_x:N = \l_stex_omtext_from_tl ,
3462   start .tl_set_x:N = \l_stex_omtext_start_str ,
3463 }
3464 \cs_new_protected:Nn \stex_omtext_args:n {
3465   \str_clear:N \l_stex_omtext_title_str
3466   \str_clear:N \l_stex_omtext_start_str
3467   \keys_set:nn { stex / omtex } { #1 }
3468   \exp_args:NNo \str_set:Nn \l_stex_omtext_title_str
3469     \l_stex_omtext_title_str
3470   \exp_args:NNo \str_set:Nn \l_stex_omtext_start_str
3471     \l_stex_omtext_start_str
3472 }
3473 \NewDocumentEnvironment {omtext} { 0{} } {
3474   \stex_omtext_args:n { #1 }
3475   \textbf{\str_if_empty:NTF \l_stex_omtext_start_str {

```

```

3476     \l_stex_omtext_title_str
3477   }{
3478     \l_stex_omtext_start_str :
3479   }}
3480 }{
3481
3482 }
3483 \NewDocumentEnvironment {assertion} { 0{ } } {
3484
3485 }{
3486
3487 }
3488
3489 \NewDocumentCommand \inlinedef { m } {
3490   \beginngroup
3491   \let\definiendum\_stex_deprec_definiendum:w
3492   \let\definame\_stex_deprec_definame:w
3493   #1
3494   \endgroup
3495 }
3496
3497 \NewDocumentCommand \inlineass { m } { #1 }
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}![#2]~instead!~(in~file~
3506       \stex_path_to_string:N \g_stex_currentfile_seq)
3507       \\\ \\\
3508     }
3509     \msg_warning:nn{stex}{warning/deprecated}
3510     \STExsymbol { #2 }![ \comp{#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?#2 }[ #2 ]~instead!~(in~file~
3516       \stex_path_to_string:N \g_stex_currentfile_seq)
3517       \\\ \\\
3518     }
3519     \msg_warning:nn{stex}{warning/deprecated}
3520     \STExsymbol { #1 }![ \comp{#2} ]
3521   }
3522 }
3523
3524
3525 \NewDocumentCommand \Trefi { 0{ } m } {
3526   \str_set:Nn \l_tmpa_str { #1 }
3527   \str_if_empty:NTF \l_tmpa_str {
3528     \msg_set:nnn{stex}{warning/deprecated}{
3529       \\\

```

```

3530     \c_backslash_str Trefi~is~deprecated! \\
3531     Please~use~\c_backslash_str STExsymbol{#2}![\exp_after:wN \stex_capitalize:n #2]~inste
3532     \stex_path_to_string:N \g_stex_currentfile_seq)
3533     \\ \\
3534   }
3535   \msg_warning:nn{stex}{warning/deprecated}
3536   \STExsymbol { #2 }![ \comp{\exp_after:wN \stex_capitalize:n #2} ]
3537 } {
3538   \msg_set:nnn{stex}{warning/deprecated}{
3539     \\
3540     \c_backslash_str Trefi~is~deprecated! \\
3541     Please~use~\c_backslash_str STExsymbol { #1 }[ \exp_after:wN \stex_capitalize:n #2 ]~i
3542     \stex_path_to_string:N \g_stex_currentfile_seq)
3543     \\ \\
3544   }
3545   \msg_warning:nn{stex}{warning/deprecated}
3546   \STExsymbol { #1 }![ \comp{\exp_after:wN \stex_capitalize:n #2} ]
3547 }
3548 }
3549
3550 \NewDocumentCommand \trefis { 0{} 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 trefi~is~deprecated! \\
3556       Please~use~\c_backslash_str STExsymbol{#2}![#2s]~instead!~(in~file~
3557       \stex_path_to_string:N \g_stex_currentfile_seq)
3558       \\ \\
3559     }
3560     \msg_warning:nn{stex}{warning/deprecated}
3561     \STExsymbol { #2 }![ \comp{#2s} ]
3562   } {
3563     \msg_set:nnn{stex}{warning/deprecated}{
3564       \\
3565       \c_backslash_str trefi~is~deprecated! \\
3566       Please~use~\c_backslash_str STExsymbol { #1 }[ #2s ]~instead!~(in~file~
3567       \stex_path_to_string:N \g_stex_currentfile_seq)
3568       \\ \\
3569     }
3570     \msg_warning:nn{stex}{warning/deprecated}
3571     \STExsymbol { #1 }![ \comp{#2s} ]
3572   }
3573 }
3574
3575
3576 \NewDocumentCommand \Trefis { 0{} m } {
3577   \str_set:Nn \l_tmpa_str { #1 }
3578   \str_if_empty:NTF \l_tmpa_str {
3579     \msg_set:nnn{stex}{warning/deprecated}{
3580       \\
3581       \c_backslash_str Trefis~is~deprecated! \\
3582       Please~use~\c_backslash_str STExsymbol{#2}![\exp_after:wN \stex_capitalize:n #2s]~inst
3583       \stex_path_to_string:N \g_stex_currentfile_seq)

```

```

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

```

```

3638 } {
3639   \msg_set:nnn{stex}{warning/deprecated}{
3640     \\\
3641     \c_backslash_str trefiii~is-deprecated! \\\
3642     Please~use~\c_backslash_str STExsymbol { #1 }[ #2~#3~#4 ]~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~#3~#4} ]
3648 }
3649 }
3650
3651
3652 \NewDocumentCommand \trefiis { 0{} m m } {
3653   \str_set:Nn \l_tmpa_str { #1 }
3654   \str_if_empty:NTF \l_tmpa_str {
3655     \msg_set:nnn{stex}{warning/deprecated}{
3656       \\\
3657       \c_backslash_str trefiis~is-deprecated! \\\
3658       Please~use~\c_backslash_str STExsymbol{#2~#3}![#2~#3s]~instead!~(in~file~
3659       \stex_path_to_string:N \g_stex_currentfile_seq)
3660       \\\
3661     }
3662     \msg_warning:nn{stex}{warning/deprecated}
3663     \STExsymbol { #2~#3 }![ \comp{#2~#3s} ]
3664   } {
3665     \msg_set:nnn{stex}{warning/deprecated}{
3666       \\\
3667       \c_backslash_str trefiis~is-deprecated! \\\
3668       Please~use~\c_backslash_str STExsymbol { #1 }[ #2~#3s ]~instead!~(in~file~
3669       \stex_path_to_string:N \g_stex_currentfile_seq)
3670       \\\
3671     }
3672     \msg_warning:nn{stex}{warning/deprecated}
3673     \STExsymbol { #1 }![ \comp{#2~#3s} ]
3674   }
3675 }
3676
3677 \NewDocumentCommand \symvariant { 0{} m 0{0} m m } {
3678   \msg_set:nnn{stex}{warning/deprecated}{
3679     \\\
3680     \c_backslash_str symvariant~is-deprecated! \\\
3681     Please~use~\c_backslash_str notation[#4]{ #2 }~instead!~(in~file~
3682     \stex_path_to_string:N \g_stex_currentfile_seq)
3683     \\\
3684   }
3685   \msg_warning:nn{stex}{warning/deprecated}
3686
3687   \notation[variant=#4]{#2}{#5}
3688 }
3689
3690 \NewDocumentCommand \mixfixi { 0{} m m m } {
3691   \msg_set:nnn{stex}{warning/deprecated}{

```

```

3692     \c_backslash_str mixfixi~is~fatally~deprecated!\\
3693     Symbol:~\l__stex_term_highlight_uri_str\\
3694     Current~file:~\stex_path_to_string:N \g_stex_currentfile_seq
3695   }
3696   \msg_error:nn{stex}{warning/deprecated}
3697 }
3698
3699
3700 \NewDocumentCommand \infix {} {
3701   \msg_set:nnn{stex}{warning/deprecated}{
3702     \c_backslash_str infix~is~fatally~deprecated!\\
3703     Symbol:~\l__stex_term_highlight_uri_str\\
3704     Current~file:~\stex_path_to_string:N \g_stex_currentfile_seq
3705   }
3706   \msg_error:nn{stex}{warning/deprecated}
3707 }
3708
3709 \let\iprec\infpres
3710
3711 \NewDocumentCommand \inlineex { m } {
3712   \msg_set:nnn{stex}{warning/deprecated}{
3713     \c_backslash_str inlineex~is~deprecated!\\
3714     No~replacement~exists~yet.\\
3715     Current~file:~\stex_path_to_string:N \g_stex_currentfile_seq
3716   }
3717   \msg_warning:nn{stex}{warning/deprecated}
3718   #1
3719 }
3720
3721
3722 \NewDocumentCommand \term { m } {
3723   \msg_set:nnn{stex}{warning/deprecated}{
3724     \c_backslash_str term~is~deprecated!\\
3725     No~replacement~exists~yet.\\
3726     Current~file:~\stex_path_to_string:N \g_stex_currentfile_seq
3727   }
3728   \msg_warning:nn{stex}{warning/deprecated}
3729   #1
3730 }
3731
3732
3733 \NewDocumentCommand \Definame { 0{} m } {
3734   \stex_get_symbol:n { #2 }
3735   \str_set:Nx \l_tmpa_str {
3736     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3737   }
3738   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3739   \scalatex_if:TF {
3740     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3741       \l_tmpa_str
3742     }
3743   } {
3744     \@defemph {
3745       \exp_after:wN \stex_capitalize:n \l_tmpa_str

```



```

3746     } { \l_stex_get_symbol_uri_str }
3747   }
3748 }
3749
3750 \NewDocumentCommand \Definiendum { 0{} m m } {
3751   \stex_get_symbol:n { #2 }
3752   \str_set:Nx \l_tmpa_str {
3753     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3754   }
3755   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3756   \scalatex_if:TF {
3757     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3758       \l_tmpa_str
3759     }
3760   } {
3761     \@defemph {
3762       \exp_after:wN \stex_capitalize:n \l_tmpa_str
3763     } { \l_stex_get_symbol_uri_str }
3764   }
3765 }
3766
3767 \NewDocumentCommand \Symname { 0{} m }{
3768   \stex_symname_args:n { #1 }
3769   \stex_get_symbol:n { #2 }
3770   \str_set:Nx \l_tmpa_str {
3771     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3772   }
3773   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3774   \exp_args:NNx \use:nn
3775   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }![
3776     \exp_after:wN \stex_capitalize:n \l_tmpa_str
3777     \l_stex_symname_post_str
3778   ] }
3779 }
3780
3781
3782 \seq_gput_right:Nx \g_stex_smsmode_allowedenvs_seq { \tl_to_str:n { mhmodnl } }
3783 \seq_gput_right:Nx \g_stex_smsmode_allowedenvs_seq { \tl_to_str:n { modsig } }
3784 \tl_gput_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\gimport\syml\symii\symiii\symiv\
3785
3786 % omtex:
3787 \cs_new_protected:Npn \lec #1 {
3788   \strut\hfil\strut\hfill(#1)
3789 }
3790 \cs_new_protected:Npn \nlex #1 {
3791   \textcolor{green}{\sl #1}
3792 }
3793
3794
3795 </compat>

```