

stex.sty: \TeX 2.0*

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

2021-11-18

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	38
4.6	Symbol Declarations	55
4.7	Notations	61
4.8	Terms	71
4.9	Notation Components	77
4.10	Structural Features	79
4.11	Put these somewhere	87
4.12	Metatheory	87
4.13	Auxiliary Packages	89

2 Manual

2.1 Modules

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

2.2 Semantic Macros and Notations

Semantic macros invoke a formally declared symbol.

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

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

Example 1

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

(ab)

.

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

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

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

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

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

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

For a macro with arity > 0 , we can refer to the operator *itself* semantically by suffixing the semantic macro with an exclamation point `!` in either text or math mode. For that reason `\notation` (and thus `\symdef`) take an additional optional argument `op=`, which allows to assign a notation for the operator itself. e.g.

Example 6

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

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

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

Example 7

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

```
Multiplication (denoted by  $\cdot$ ) is defined by...
```

The macro `\comp` as used everywhere above is responsible for highlighting, linking, and tooltips, and should be wrapped around the notation (or text) components that should be treated accordingly. While it is attractive to just wrap a whole notation, this would also wrap around e.g. the arguments themselves, so instead, the user is tasked with marking the notation components themselves.

The precise behaviour of `\comp` is governed by the macro `\@comp`, which takes two arguments: The tex code of the text (unexpanded) to highlight, and the URI of the current symbol. `\@comp` can be safely redefined to customize the behaviour.

The starred variant `\symdecl*{foo}` does not introduce a semantic macro, but still declares a corresponding symbol. `foo` (like any other symbol, for that matter) can then be accessed via `\STEXsymbol{foo}` or (if `foo` was declared in a module `Foo`) via `\STEXModule{Foo}?{foo}`.

both `\STEXsymbol` and `\STEXModule` take any arbitrary ending segment of a full URI to determine which symbol or module is meant. e.g. `\STEXsymbol{Foo?foo}` is also valid, as are e.g. `\STEXModule{path?Foo}?{foo}` or `\STEXsymbol{path?Foo?foo}`

There's also a convient shortcut `\symref{?foo}{some text}` for `\STEXsymbol{?foo}![some text]`

2.2.1 Other Argument Types

So far, we have stated the arity of a semantic macro directly. This works if we only have “normal” (or more precisely: *i*-type) arguments. To make use of other argument types, instead of providing the arity numerically, we can provide it as a sequence of characters representing the argument types – e.g. instead of writing `args=2`, we can equivalently write `args=ii`, indicating that the macro takes two *i*-type arguments.

Besides *i*-type arguments, `STEX` has two other types, which we will discuss now.

The first are *binding* (*b*-type) arguments, representing variables that are *bound* by the operator. This is the case for example in the above `\forevery`-macro: The first argument is not actually an argument that the `forevery` “function” is “applied” to; rather, the first argument is a new variable (e.g. *x*) that is *bound* in the subsequent argument. More accurately, the macro should therefore have been implemented thusly:

```
\symdef[args=bi]{forevery}{\forall #1.\; #2}
```

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

Example 8

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

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

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

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

Example 9

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

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

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

² 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:

```
\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>	LaTeX2e and LaTeX3 conditionals for LaTeXML.
<code>\latexml_if_p:</code>	
<code>\latexml_if:T</code>	
<code>\latexml_if:F</code>	
<code>\latexml_if:TF</code>	

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

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

Annotates the HTML generated by `⟨content⟩` with

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

3.1.2 Languages

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

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

3.2 Files, Paths, URIs

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

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

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

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

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

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

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

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

`\g_stex_currentfile_seq`

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

Test 1

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

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

3.3 MathHub Archives

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

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

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

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

`\l_stex_current_repository_prop`

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

`\stex_set_current_repository:n`

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

`\stex_require_repository:n`

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

`\libinput`

`\libinput{<filename>}`

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

Test 2

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

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

3.4 The Module System

`\l_stex_current_module_prop`

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

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

Additionally, it stores:

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

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

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

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

`\stex_add_to_current_module:n`
`\STEXexport`

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

`\stex_add_constant_to_current_module:n`

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

`\stex_add_import_to_current_module:n`

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

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

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

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

<code>\stex_modules_current_namespace:</code>

Computes the current namespace

Test 3

```

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

```

```

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

```

3.4.1 The module-environment

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

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

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

Test 4

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

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

Test 5

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

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

\l_stex_all_modules_seq

Stores full URIs for all modules currently in scope.

\STEXModule

\STEXModule {*<fragment>*}

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

`\stex_invoke_module:n`

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

Test 6

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

Module 3.2[STEXModuleTest1]

Module 3.3[STEXModuleTest2]

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

3.4.2 SMS Mode

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

`\g_stex_smsmode_allowedmacros_tl`

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

`\g_stex_smsmode_allowedmacros_escape_tl`

Macros that are executed with the category codes restored.

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

`\g_stex_smsmode_allowedenvs_seq`

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

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

Tests whether SMS mode is currently active.

`\stex_smsmode_set_codes:`

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

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

`\stex_in_smsmode:nn`

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

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

Test 7

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

3.4.3 Imports and Inheritance

`\importmodule`

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

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

Test 8

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

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

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

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

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

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

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

Test 9

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

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

Module 3.8[UseTest1]

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

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

Test 10

Circular dependencies:

```

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

```

Circular dependencies:

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

<hr/> <hr/> <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).
	<ol style="list-style-type: none"> 1. If <code><archive-ID></code> is empty: <ol style="list-style-type: none"> (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: <ol style="list-style-type: none"> (a) If <code><path></code> is empty, then <code><name></code> must have been declared earlier in the same file and retrievable from <code>\g_stex_modules_in_file_seq</code>, or a file with name <code><name>.<lang>.tex</code> must exist in the top <code>source</code> folder of the archive, containing a module <code><name></code>. That module should lie directly in the namespace of the archive. (b) If <code><path></code> is not empty, it must point to the path of the containing file as well as the namespace, relative to the namespace of the archive. If a module by that namespace exists, it is returned. Otherwise, we call <code>\stex_require_module:nn</code> on the <code>source</code> directory of the archive to find the file.

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

Checks whether a module with URI `<ns>?<name>` already exists. If not, it looks for a plausible file that declares a module with that URI.

Finally, activates that module by executing its `content`-field.

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

A property list mapping file paths to the lists of all modules declared therein. `\g_stex_modules_in_file_seq` always points to the current file(-stream - `\inputs` are considered the same file).

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

3.5 Symbols and Terms

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

Declares a new symbol with semantic macro `\macroname`. Optional arguments are:

- **name**: An (OMDOC) name. By default equal to `⟨macroname⟩`.
- **type**: An (ideally semantic) term. Not used by $\S\mathrm{TEX}$, but passed on to MMT for semantic services.
- **local**: A boolean (by default false). If set, this declaration will not be added to the module content, i.e. importing the current module will not make this declaration available.
- **args**: Specifies the “signature” of the semantic macro. Can be either an integer $0 \leq n \leq 9$, or a (more precise) sequence of the following characters:
 - i a “normal” argument, e.g. `\symdecl[args=ii]{plus}` allows for `\plus{2}{2}`.
 - a an *associative* argument; i.e. a sequence of arbitrarily many arguments provided as a comma-separated list, e.g. `\symdecl[args=a]{plus}` allows for `\plus{2,2,2}`.
 - b a *variable* argument. Is treated by $\S\mathrm{TEX}$ like an i-argument, but an application is turned into an `OMBind` in OMDOC, binding the provided variable in the subsequent arguments of the operator; e.g. `\symdecl[args=bi]{forall}` allows for `\forall{x\in\mathrm{Nat}}{x\geq 0}`.

`\stex_symdecl_do:n`

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

Ultimately stores the symbol `⟨URI⟩` in the property list `\g_stex_symdecl_⟨URI⟩_prop` with fields:

- **name** (string),
- **module** (string),
- **notations** (sequence of strings; initially empty),
- **local** (boolean),
- **type** (token list),
- **args** (string of is, as and bs),
- **arity** (integer string),
- **assocs** (integer string; number of associative arguments),

Test 11

```
\begin{module}{SymdeclTest}
\symdecl[name=foo, args=3]{bar}
\symdecl[name=foobar, args=iab]{bari}
\symdecl[def=\bar* abc]{bardef}
\ExplSyntaxOn
Meaning:-\present\bar\
\stex_get_symbol:n { bar }
Result:-\l_stex_get_symbol_uri_str\
Meaning:-\present\bardef\
\ExplSyntaxOff
\end{module}
```

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

\l_stex_all_symbols_seq

Stores full URIs for all modules currently in scope.

\stex_get_symbol:n

Computes the full URI of a symbol from a macro argument, e.g. the macro name, the macro itself, the full URI...

\STEXsymbol

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

\symref

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

\stex_invoke_symbol:n

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

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

\notation

`\notation[⟨args⟩]{⟨symbol⟩}{⟨notations+⟩}`
Introduces a new notation for `⟨symbol⟩`, see `\stex_notation_do:nn`

`\stex_notation_do:nn`

`\stex_notation_do:nn{<URI>}{<notations+>}`

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

Ultimately stores the notation in the property list `\g_stex_notation_<URI>#<variant>#<lang>_prop` with fields:

- symbol (URI string),
- language (string),
- variant (string),
- opprec (integer string),
- argprecs (sequence of integer strings)

Test 12

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

Module 3.13[NotationTest]

`\symdef`

`\symdef[<args>]{<symbol>}{<notations+>}`

Combines `\symdecl` and `\notation` by introducing a new symbol and assigning a new notation for it.

Test 13

```
\begin{module}{SymdefTest}
\symdef[ args=a, prec=50]{plus}{ #1 }{#1 \comp+ #2}
$\plus{ a,b,c }$
\end{module}
```

Module 3.14[SymdefTest]
(a+b+c)

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

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

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


```

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

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

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

```

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

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

Test 16

```

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

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

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

 $\bar{!}[\mathtt{bar}]\$$ 

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

\bar{!bar}

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

\end{module}

```

```

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

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

```

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

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

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

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

`\@defemph`

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

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

<code>\STEXinvisible</code>	Exports its argument as OMDOC (invisible), but does not produce PDF output. Useful e.g. for semantic macros that take arguments that are not part of the symbolic notation.
-----------------------------	---

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

3.6 Structural Features

<code>symboldoc</code>	$\begin{array}{l} \backslash\text{begin}\{\langle\text{symboldoc}\rangle\}\{\langle\text{symbols}\rangle\} \langle\text{text}\rangle \backslash\text{end}\{\langle\text{symboldoc}\rangle\} \\ \text{Declares } \langle\text{text}\rangle \text{ to be a (natural language, encyclopaedic) description of } \{\langle\text{symbols}\rangle\} \\ \text{(a comma separated list of symbol identifiers).} \end{array}$
------------------------	---

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}
 $\$ \text{isa}\{\backslash\text{op } ab\} \backslash\text{universe}\$$ 
\end{structure}

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

\present\magma

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

Test:  $\$ \text{mM}\{op\}ab\$$ 

Test2:  $\$ \text{mM}\{\}\$$ 
\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 \TeX document class

```

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

```

```

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

```

4.2 Preliminaries

```

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

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

```

\sTeX The sTeX logo:

```

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

```

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

Messages

```

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

```

\stex_debug:n Debug mode

```

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

```

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

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

(End definition for `\c__stex_sms_iow`.)

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

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

4.2.1 L^AT_EX_ML and S^CA_LT_EX

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

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

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

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

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

4.2.2 HTML Annotations

77 `<@=stex_annotate>`

`\l__stex_annotate_arg_tl`
`\c__stex_annotate_emptyarg_tl`

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

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

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

`__stex_annotate_checkempty:n`

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

(End definition for `__stex_annotate_checkempty:n`.)

`\stex_annotate:nnw`

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

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

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

```

116   } {
117     \tl_use:N \l__stex_annotate_arg_tl
118   }
119 }
120 \NewDocumentEnvironment{stex_annotate_env} { m m } {
121   \par
122   \scalatex_annotate_HTML_begin:n {
123     property="stex:#1" ~
124     resource="#2"
125   }
126 }{
127   \scalatex_annotate_HTML_end:
128 }
129 }{
130   \latexml_if:TF {
131     \cs_new_protected:Nn \stex_annotate:nnn {
132       \__stex_annotate_checkempty:n { #3 }
133       \mode_if_math:TF {
134         \cs:w latexml@annotate@math\cs_end:{#1}{#2}{
135           \tl_use:N \l__stex_annotate_arg_tl
136         }
137       }{
138         \cs:w latexml@annotate@text\cs_end:{#1}{#2}{
139           \tl_use:N \l__stex_annotate_arg_tl
140         }
141       }
142     }
143     \cs_new_protected:Nn \stex_annotate_invisible:n {
144       \__stex_annotate_checkempty:n { #1 }
145       \mode_if_math:TF {
146         \cs:w latexml@invisible@math\cs_end:{
147           \tl_use:N \l__stex_annotate_arg_tl
148         }
149       } {
150         \cs:w latexml@invisible@text\cs_end:{
151           \tl_use:N \l__stex_annotate_arg_tl
152         }
153       }
154     }
155     \cs_new_protected:Nn \stex_annotate_invisible:nnn {
156       \__stex_annotate_checkempty:n { #3 }
157       \cs:w latexml@annotate@invisible\cs_end:{#1}{#2}{
158         \tl_use:N \l__stex_annotate_arg_tl
159       }
160     }
161     \NewDocumentEnvironment{stex_annotate_env} { m m } {
162       \par\begin{latexml@annotateenv}{#1}{#2}
163     }{
164       \end{latexml@annotateenv}
165     }
166   }{
167     \cs_new_protected:Nn \stex_annotate:nnn {#3}
168     \cs_new_protected:Nn \stex_annotate_invisible:n {}
169     \cs_new_protected:Nn \stex_annotate_invisible:nnn {}

```

```

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

```

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

4.2.3 Languages

```

173 <@@=stex_language>

```

`\c_stex_languages_prop`

`\c_stex_language_abbrevs_prop`

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

```

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

```

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

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

```

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

```

4.3 Files, Paths and URIs

214 `<@@=stex_path>`

4.3.1 Generic Path Handling

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

```

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

```

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

```

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

```

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

```

\c__stex_path_dot_str . and .., respectively.
\c__stex_path_up_str
241 \str_const:Nn \c__stex_path_dot_str {.}
242 \str_const:Nn \c__stex_path_up_str {...}

```

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

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

```

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

```

```

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

```

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

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

```

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

```

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

4.3.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

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

```



```

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

```

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

We determine the PWD

`\c_stex_pwd_seq`
`\c_stex_pwd_str`

```

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

```

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

4.3.3 File Hooks and Tracking

```

305 <@=stex_files>

```

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

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

`\g__stex_files_stack` keeps track of file changes

```

306 \seq_gclear_new:N\g__stex_files_stack

```

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`

```

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

```

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

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

```

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

```

```

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

```

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

4.4 MathHub Repositories

```

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

```

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

```

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

```

```

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

```

(End definition for __stex_mathhub_do_manifest:n.)

\l_stex_mathhub_manifest_file_seq

```

373 \str_new:N\l__stex_mathhub_manifest_file_seq

```

(End definition for \l_stex_mathhub_manifest_file_seq.)

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

```

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

```

(End definition for `_stex_mathhub_find_manifest:N`.)

`\c_stex_mathhub_manifest_ior` File variable used for MANIFEST-files

409 `\ior_new:N \c__stex_mathhub_manifest_ior`

(End definition for `\c_stex_mathhub_manifest_ior`.)

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

```

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

```

(End definition for `_stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

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

```

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

`\stex_require_repository:n`

```

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

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

`\l_stex_current_repository_prop` Current MathHub repository

```

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

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

`\libinput`

```

483 \cs_new_protected:Npn \libinput #1 {
484   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
485     \msg_set:nnn{stex}{error/norepository}{
486       \c_backslash_str libinput~needs~to~be~called~in~an~archive
487     }
488     \msg_error:nn{stex}{error/norepository}
489   }
490   \bool_set_false:N \l_tmpa_bool
491   \tl_clear:N \l_tmpa_tl
492   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
493   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
494   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str
495   \seq_pop_left:NNT \l_tmpb_seq \l_tmpb_str {
```

```

496 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
497 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
498 / meta-inf / lib / #1.tex}{
499 \bool_set_true:N \l_tmpa_bool
500 \tl_put_right:Nx \l_tmpa_tl {
501 \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
502 / meta-inf / lib / #1.tex}
503 }
504 }{}
505 }
506 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
507 / \l_tmpa_str / lib / #1.tex
508 }{
509 \bool_set_true:N \l_tmpa_bool
510 \tl_put_right:Nx \l_tmpa_tl {
511 \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
512 / \l_tmpa_str / lib / #1.tex}
513 }
514 }{}
515 \bool_if:NF \l_tmpa_bool {
516 \msg_set:nnn{stex}{error/nofile}{
517 \c_backslash_str libinput~no~file~#1.tex~found!
518 }
519 \msg_error:nn{stex}{error/nofile}
520 }
521 \l_tmpa_tl
522 }

```

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

4.5 Module System

```

523 <@@=stex_module>

```

`\l_stex_current_module_prop`

```

524 \prop_new:N \l_stex_current_module_prop

```

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

`stex_if_in_module_p:`

`stex_if_in_module:TF`

```

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

```

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

`stex_if_module_exists_p:n`

`stex_if_module_exists:nTF`

```

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

```

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

`\stex_add_to_current_module:n`
`\STEXexport`

```

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

```

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

`\stex_add_constant_to_current_module:n`

```

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

```

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

`\stex_add_import_to_current_module:n`

```

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

```

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

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

`\l_stex_modules_ns_str`

```

555 \str_new:N \l_stex_modules_ns_str
556 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
557   \str_set:Nx \l_tmpa_str { #1 }
558   \seq_set_eq:NN \l_tmpa_seq #2
559   % split off file extension
560   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
561   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
562   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
563   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
564
565   \bool_set_true:N \l_tmpa_bool
566   \bool_while_do:Nn \l_tmpa_bool {
567     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
568     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
569       {source} { \bool_set_false:N \l_tmpa_bool }

```

```

570   }{}{
571     \seq_if_empty:NT \l_tmpa_seq {
572       \bool_set_false:N \l_tmpa_bool
573     }
574   }
575 }
576
577 \seq_if_empty:NTF \l_tmpa_seq {
578   \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
579 }{
580   \str_set:Nx \l_stex_modules_ns_str {
581     \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
582   }
583 }
584 }

```

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

`\stex_modules_current_namespace:`

```

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

```

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

4.5.1 The module environment

`\l_stex_all_modules_seq` Stores all available modules

```

600 \seq_new:N \l_stex_all_modules_seq

```

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

`\STEXModule`

`\stex_invoke_module:n`

```

601 \NewDocumentCommand \STEXModule { m } {
602   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
603   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
604   \tl_set:Nn \l_tmpa_tl {
605     \msg_set:nnn{stex}{error/unknownmodule}{
606       No~module~#1~found!
607     }
608     \msg_error:nn{stex}{error/unknownmodule}
609   }

```



```

610 \seq_map_inline:Nn \l_stex_all_modules_seq {
611   \str_set:Nn \l_tmpb_str { ##1 }
612   \str_if_eq:eeT { \l_tmpa_str } {
613     \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
614   } {
615     \seq_map_break:n {
616       \tl_set:Nn \l_tmpa_tl {
617         \stex_invoke_module:n { ##1 }
618       }
619     }
620   }
621 }
622 \l_tmpa_tl
623 }
624
625 \cs_new_protected:Nn \stex_invoke_module:n {
626   \stex_debug:n{Invoking~module~#1}
627   \peek_charcode_remove:NTF ! {
628     \__stex_module_invoke_uri:nN { #1 }
629   } {
630     \peek_charcode_remove:NTF ? {
631       \__stex_module_invoke_symbol:nn { #1 }
632     } {
633       \msg_set:nnn{stex}{error/syntax}{
634         Syntax~error:~?~or~!~expected~after~
635         \c_backslash_str STEXModule{#1}
636       }
637       \msg_error:nn{stex}{error/syntax}
638     }
639   }
640 }
641
642 \cs_new_protected:Nn \__stex_module_invoke_uri:nN {
643   \str_set:Nn #2 { #1 }
644 }
645
646 \cs_new_protected:Nn \__stex_module_invoke_symbol:nn {
647   \stex_invoke_symbol:n{#1?#2}
648 }

```

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

module module arguments:

```

649 \keys_define:nn { stex / module } {
650   title      .tl_set_x:N = \l_stex_module_title_str ,
651   ns         .tl_set_x:N = \l_stex_module_ns_str ,
652   lang       .tl_set_x:N = \l_stex_module_lang_str ,
653   sig        .tl_set_x:N = \l_stex_module_sig_str ,
654   creators   .tl_set_x:N = \l_stex_module_creators_str ,
655   contributors .tl_set_x:N = \l_stex_module_contributors_str ,
656   meta       .tl_set_x:N = \l_stex_module_meta_str
657 }
658

```

```

659 % module parameters here? In the body?
660
661 \cs_new_protected:Nn \__stex_module_args:n {
662   \str_clear:N \l_stex_module_title_str
663   \str_clear:N \l_stex_module_ns_str
664   \str_clear:N \l_stex_module_lang_str
665   \str_clear:N \l_stex_module_sig_str
666   \str_clear:N \l_stex_module_creators_str
667   \str_clear:N \l_stex_module_contributors_str
668   \str_clear:N \l_stex_module_meta_str
669   \keys_set:nn { stex / module } { #1 }
670   \exp_args:NNo \str_set:Nn \l_stex_module_title_str
671     \l_stex_module_title_str
672   \exp_args:NNo \str_set:Nn \l_stex_module_ns_str
673     \l_stex_module_ns_str
674   \exp_args:NNo \str_set:Nn \l_stex_module_lang_str
675     \l_stex_module_lang_str
676   \exp_args:NNo \str_set:Nn \l_stex_module_sig_str
677     \l_stex_module_sig_str
678   \exp_args:NNo \str_set:Nn \l_stex_module_meta_str
679     \l_stex_module_meta_str
680   \exp_args:NNo \str_set:Nn \l_stex_module_creators_str
681     \l_stex_module_creators_str
682   \exp_args:NNo \str_set:Nn \l_stex_module_contributors_str
683     \l_stex_module_contributors_str
684 }

```

__stex_module_begin_module: implements \begin{module}

```

685 \cs_new_protected:Nn \__stex_module_begin_module: {
686   % Nested module?
687   \stex_if_in_module:TF {
688     % Nested module
689     \prop_get:NnN \l_stex_current_module_prop
690       { ns } \l_stex_module_ns_str
691     \str_set:Nx \l_stex_module_name_str {
692       \prop_item:Nn \l_stex_current_module_prop
693         { name } / \l_stex_module_name_str
694     }
695   }{
696     % not nested:
697     \str_if_empty:NT \l_stex_module_ns_str {
698       \stex_modules_current_namespace:
699       \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
700       \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
701         / {\l_stex_module_ns_str}
702       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
703       \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
704         \str_set:Nx \l_stex_module_ns_str {
705           \stex_path_to_string:N \l_tmpa_seq
706         }
707       }
708     }
709   }
710 }

```

```

711 % language
712 \str_if_empty:NT \l_stex_module_lang_str {
713   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
714   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
715   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
716   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
717   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
718     \stex_debug:n {Language~\l_stex_module_lang_str~
719       inferred~from~file~name}
720     \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
721   }
722 }
723
724 \str_if_empty:NF \l_stex_module_lang_str {
725   \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
726   \l_tmpa_str {
727     \ltx@ifpackageloaded{babel}{
728       \exp_args:Nx \selectlanguage { \l_tmpa_str }
729     }{}
730   } {
731     \msg_set:nnn{stex}{error/unknownlanguage}{
732       Unknown~language~\l_tmpa_str
733     }
734     \msg_error:nn{stex}{error/unknownlanguage}
735   }
736 }
737
738 % signature
739 \str_if_empty:NTF \l_stex_module_sig_str {
740   \str_clear:N \l_tmpa_str
741   \seq_clear:N \l_tmpa_seq
742   \tl_clear:N \l_tmpa_tl
743   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
744     name      = \l_stex_module_name_str ,
745     ns        = \l_stex_module_ns_str ,
746     imports   = \exp_not:o { \l_tmpa_seq } ,
747     constants = \exp_not:o { \l_tmpa_seq } ,
748     content   = \exp_not:o { \l_tmpa_tl } ,
749     file      = \exp_not:o { \g_stex_currentfile_seq } ,
750     lang      = \l_stex_module_lang_str ,
751     sig       = \l_stex_module_sig_str ,
752     meta      = \l_stex_module_meta_str
753   }
754 }{
755   \str_if_empty:NT \l_stex_module_lang_str {
756     \msg_set:nnn{stex}{error/siglanguage}{
757       Module~\l_stex_module_ns_str?\l_stex_module_name_str~
758       declares~signature~\l_stex_module_sig_str,~but~does~not~
759       declare~its~language
760     }
761     \msg_error:nn{stex}{error/siglanguage}
762   }
763
764   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq

```

```

765 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
766 \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
767 \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
768 \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
769 \str_set:Nx \l_tmpa_str {
770   \stex_path_to_string:N \l_tmpa_seq /
771   \l_tmpa_str . \l_stex_module_sig_str .tex
772 }
773 \IfFileExists \l_tmpa_str {
774   \exp_args:No \stex_in_smsmode:nn { \l_tmpa_str } {
775     \seq_clear:N \l_stex_all_modules_seq
776     \prop_clear:N \l_stex_current_module_prop
777     \stex_debug:n{Loading~signature~\l_tmpa_str}
778     \input { \l_tmpa_str }
779   }
780 }{
781   \msg_set:nnn{stex}{error/modulemissing}{
782     No~file~for~signature~module~\l_tmpa_str~found
783   }
784   \msg_error:nn{stex}{error/modulemissing}
785 }
786 \stex_activate_module:n {
787   \l_stex_module_ns_str ? \l_stex_module_name_str
788 }
789 \prop_set_eq:Nc \l_stex_current_module_prop {
790   c_stex_module_
791   \l_stex_module_ns_str ?
792   \l_stex_module_name_str
793   _prop
794 }
795 }
796
797 % metatheory
798 \str_if_empty:NT \l_stex_module_meta_str {
799   \str_set:Nx \l_stex_module_meta_str {
800     \c_stex_metatheory_ns_str ? Metatheory
801   }
802 }
803
804
805 \stex_debug:n{
806   New~module:\\
807   Namespace:~\l_stex_module_ns_str\\
808   Name:~\l_stex_module_name_str\\
809   Language:~\l_stex_module_lang_str\\
810   Signature:~\l_stex_module_sig_str\\
811   Metatheory:~\l_stex_module_meta_str\\
812   File:~\stex_path_to_string:N \g_stex_currentfile_seq
813 }
814
815 \seq_put_right:Nx \l_stex_all_modules_seq {
816   \l_stex_module_ns_str ? \l_stex_module_name_str
817 }
818

```

```

819 \seq_gput_right:Nx \g_stex_modules_in_file_seq
820   { \l_stex_module_ns_str ? \l_stex_module_name_str }
821
822 \stex_if_smsmode:TF {
823   \stex_smsmode_set_codes:
824 } {
825   \begin{stex_annotate_env} {theory} {
826     \l_stex_module_ns_str ? \l_stex_module_name_str
827   }
828
829   \stex_annotate_invisible:nnn{header}{} {
830     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
831     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
832     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
833       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
834     }
835   }
836 }
837
838 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
839   \exp_args:Nx \STEXexport{
840     \stex_activate_module:n { \l_stex_module_meta_str }
841   }
842 }
843 % TODO: Inherit metatheory for nested modules?
844 }
845 \iffalse \end{stex_annotate_env} \fi % make syntax highlighting work again

```

(End definition for _stex_module_begin_module:.)

_stex_module_end_module: implements \end{module}

```

846 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
847 \cs_new_protected:Nn \_stex_module_end_module: {
848   \str_set:Nx \l_tmpa_str {
849     c_stex_module_
850     \prop_item:Nn \l_stex_current_module_prop { ns } ?
851     \prop_item:Nn \l_stex_current_module_prop { name }
852     _prop
853   }
854   %^^A \prop_new:c { \l_tmpa_str }
855   \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
856   \stex_debug:n{Closing-module~\prop_item:Nn \l_stex_current_module_prop { name }}
857   \stex_if_smsmode:TF {
858     \exp_args:Nx \stex_addtosms:n {
859       \prop_gset_from_keyval:cn {
860         c_stex_module_
861         \prop_item:Nn \l_stex_current_module_prop { ns } ?
862         \prop_item:Nn \l_stex_current_module_prop { name }
863         _prop
864       } {
865         name      = \prop_item:cn { \l_tmpa_str } { name } ,
866         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
867         imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
868         constants = \prop_item:cn { \l_tmpa_str } { constants } ,

```

```

869         content = \prop_item:cn { \l_tmpa_str } { content } ,
870         file     = \prop_item:cn { \l_tmpa_str } { file } ,
871         lang     = \prop_item:cn { \l_tmpa_str } { lang } ,
872         sig      = \prop_item:cn { \l_tmpa_str } { sig } ,
873         meta     = \prop_item:cn { \l_tmpa_str } { meta }
874     }
875 }
876 }{
877     \end{stex_annotate_env}
878 }
879 }

```

(End definition for `_stex_module_end_module:.`)

@module The core environment, with no header

```

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

```

\stex_modules_heading: Code for document headers

```

888 \cs_if_exist:NTF \thesection {
889     \newcounter{module}[section]
890 }{
891     \newcounter{module}
892 }
893
894 \bool_if:NT \c_stex_showmods_bool {
895     \latexml_if:F { \RequirePackage{mdframed} }
896 }
897
898 \cs_new_protected:Nn \stex_modules_heading: {
899     \stepcounter{module}
900     \par
901     \bool_if:NT \c_stex_showmods_bool {
902         \noindent{\textbf{Module} ~
903             \cs_if_exist:NT \thesection {\thesection.}
904             \themodule ~ [\l_stex_module_name_str]
905         }
906         % TODO references
907         % \sref@label@id{Module \thesection.\themodule [\module@name]]%
908         \str_if_empty:NTF \l_stex_module_title_str {
909             }{
910                 \quad(\l_stex_module_title_str)\hfill
911             }\par
912         }
913     }

```

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

Finally:

```

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

```

4.5.2 SMS Mode

```

926 <@=stex_smsmode>

```

```

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

```

```

927 \tl_new:N \g_stex_smsmode_allowedmacros_tl
928 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
929 \seq_new:N \g_stex_smsmode_allowedenvs_seq
930
931 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
932   \makeatletter
933   \makeatother
934   \ExplSyntaxOn
935   \ExplSyntaxOff
936 }
937
938 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
939   \symdef
940   \importmodule
941   \notation
942   \symdecl
943   \STEXexport
944 }
945
946 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
947   \tl_to_str:n {
948     module,
949     @module
950   }
951 }

```

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

```

\stex_if_smsmode_p:
\stex_if_smsmode:TF

```

```

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

```

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

```

\stex_smsmode_if_catcodes_p: Checks whether the SMS mode category code scheme is active.
__stex_smsmode_if_catcodes:TF
957 \bool_new:N \g__stex_smsmode_catcode_bool
958 \bool_set_false:N \g__stex_smsmode_catcode_bool
959 \prg_new_conditional:Nnn \__stex_smsmode_if_catcodes: { p, T, F, TF } {
960   \bool_if:NTF \g__stex_smsmode_catcode_bool
961   \prg_return_true: \prg_return_false:
962 }

```

(End definition for __stex_smsmode_if_catcodes:TF.)

\stex_smsmode_set_codes:

```

963 \cs_new_protected:Nn \stex_smsmode_set_codes: {
964   \stex_if_smsmode:T {
965     \__stex_smsmode_if_catcodes:F {
966       \bool_gset_true:N \g__stex_smsmode_catcode_bool
967       \exp_after:wN \char_gset_active_eq:NN
968       \c_backslash_str \__stex_smsmode_cs:
969       \tex_global:D \char_set_catcode_active:N \
970       \tex_global:D \char_set_catcode_other:N $
971       \tex_global:D \char_set_catcode_other:N ^
972       \tex_global:D \char_set_catcode_other:N _
973       \tex_global:D \char_set_catcode_other:N &
974       \tex_global:D \char_set_catcode_other:N ##
975     }
976   }
977 } \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.

```

978 \cs_new_protected:Nn \__stex_smsmode_unset_codes: {
979   \__stex_smsmode_if_catcodes:T {
980     \bool_gset_false:N \g__stex_smsmode_catcode_bool
981     \exp_after:wN \tex_global:D \exp_after:wN
982     \char_set_catcode_escape:N \c_backslash_str
983     \tex_global:D \char_set_catcode_math_toggle:N $
984     \tex_global:D \char_set_catcode_math_superscript:N ^
985     \tex_global:D \char_set_catcode_math_subscript:N _
986     \tex_global:D \char_set_catcode_alignment:N &
987     \tex_global:D \char_set_catcode_parameter:N ##
988   }
989 } \iffalse $ \fi % to make syntax highlighting work again

```

(End definition for __stex_smsmode_unset_codes:.)

\stex_in_smsmode:nn

```

990 \cs_new_protected:Nn \stex_in_smsmode:nn {
991   \vbox_set:Nn \l_tmpa_box {
992     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
993     \bool_gset_true:N \g__stex_smsmode_bool
994     \stex_smsmode_set_codes:
995     #2
996     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
997     \stex_if_smsmode:F {

```



```

998     \__stex_smsmode_unset_codes:
999   }
1000 }
1001 \box_clear:N \l_tmpa_box
1002 }

```

(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:

```

1003 \cs_new_protected:Nn \__stex_smsmode_cs: {
1004   \str_clear:N \l_tmpa_str
1005   \peek_analysis_map_inline:n {
1006     % #1: token (one expansion)
1007     % #2: charcode
1008     % #3 catcode
1009     \token_if_eq_charcode:NNTF ##3 B {
1010       % token is a letter
1011       \exp_args:NNNo \str_put_right:Nn \l_tmpa_str { ##1 }
1012     } {
1013       \str_if_empty:NNTF \l_tmpa_str {
1014         % we don't allow (or need) single non-letter CSs
1015         % for now
1016         \peek_analysis_map_break:
1017       } {
1018         \str_if_eq:ontf \l_tmpa_str { begin } {
1019           \peek_analysis_map_break:n {
1020             \exp_after:wN \__stex_smsmode_checkbegin:n ##1
1021           }
1022         } {
1023           \str_if_eq:ontf \l_tmpa_str { end } {
1024             \peek_analysis_map_break:n {
1025               \exp_after:wN \__stex_smsmode_checkend:n ##1
1026             }
1027           } {
1028             \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
1029             \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1030               \g_stex_smsmode_allowedmacros_tl
1031               { \use:c{\l_tmpa_str} } {
1032               \stex_debug:n{Executing~1:~\l_tmpa_str}
1033               \peek_analysis_map_break:n {
1034                 \exp_after:wN \l_tmpa_tl ##1
1035               }
1036             } {
1037               \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1038               \g_stex_smsmode_allowedmacros_escape_tl
1039               { \use:c{\l_tmpa_str} } {
1040               \stex_debug:n{Executing~2:~\l_tmpa_str}
1041               % TODO \__stex_smsmode_rescan_cs:
1042               \exp_after:wN \exp_after:wN \exp_after:wN
1043               \token_if_eq_charcode:NNTF \exp_after:wN \c_backslash_str ##1 {
1044               \peek_analysis_map_break:n {
1045                 \__stex_smsmode_unset_codes:
1046                 \__stex_smsmode_rescan_cs:

```

```

1047 %           }
1048 %           } {
1049           \peek_analysis_map_break:n {
1050           \__stex_smsmode_unset_codes:
1051           \exp_after:wN \l_tmpa_tl ##1
1052           }
1053 %           }
1054           } {
1055           \peek_analysis_map_break:n { ##1 }
1056           }
1057         }
1058       }
1059     }
1060   }
1061 }
1062 }
1063 }

```

(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:

```

1064 \cs_new_protected:Nn \__stex_smsmode_rescan_cs: {
1065   \str_clear:N \l_tmpb_str
1066   \peek_analysis_map_inline:n {
1067     \token_if_eq_charcode:NNTF ##3 B {
1068       % token is a letter
1069       \exp_args:NNo \str_put_right:Nn \l_tmpb_str { ##1 }
1070     } {
1071       \peek_analysis_map_break:n {
1072         \exp_after:wN \use:c \exp_after:wN {
1073           \exp_after:wN \l_tmpa_str\exp_after:wN
1074         } \use:c { \l_tmpb_str \exp_after:wN } ##1
1075       }
1076     }
1077   }
1078 }

```

(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.

```

1079 \cs_new_protected:Nn \__stex_smsmode_checkbegin:n {
1080   \str_set:Nn \l_tmpa_str { #1 }
1081   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1082     \__stex_smsmode_unset_codes:
1083     \begin{#1}
1084   }
1085 }

```

(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.

```

1086 \cs_new_protected:Nn \__stex_smsmode_checkend:n {
1087   \str_set:Nn \l_tmpa_str { #1 }
1088   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {

```

```

1089     \end{#1}
1090   }
1091 }

```

(End definition for `_stex_smsmode_checkend:n`.)

4.5.3 Inheritance

```

1092 <@@=stex_importmodule>

```

`\stex_import_module_uri:nn`

```

1093 \cs_new_protected:Nn \stex_import_module_uri:nn {
1094   \str_set:Nx \l__stex_importmodule_archive_str { #1 }
1095   \str_set:Nx \l__stex_importmodule_path_str { #2 }
1096   \str_if_empty:NT \l__stex_importmodule_archive_str {
1097     \prop_if_empty:NF \l_stex_current_repository_prop {
1098       \prop_get:NnN \l_stex_current_repository_prop { id } \l__stex_importmodule_archive_str
1099     }
1100   }
1101
1102   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l__stex_importmodule_path_str }
1103   \seq_pop_right:NN \l_tmpb_seq \l__stex_importmodule_name_str
1104   \str_set:Nx \l__stex_importmodule_path_str { \seq_use:Nn \l_tmpb_seq ? }
1105
1106   \str_if_empty:NTF \l__stex_importmodule_archive_str {
1107     \stex_modules_current_namespace:
1108     \str_if_empty:NF \l__stex_importmodule_path_str {
1109       \str_set:Nx \l_stex_module_ns_str {
1110         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1111       }
1112     }
1113   }{
1114     \stex_require_repository:n \l__stex_importmodule_archive_str
1115     \prop_get:cnN { c_stex_mathhub_\l__stex_importmodule_archive_str _manifest_prop } { ns }
1116     \l_stex_module_ns_str
1117     \str_if_empty:NF \l__stex_importmodule_path_str {
1118       \str_set:Nx \l_stex_module_ns_str {
1119         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1120       }
1121     }
1122   }
1123 }

```

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

<code>\l_stex_importmodule_name_str</code>	Store the return values of <code>\stex_import_module_uri:nn</code> .
<code>\l_stex_importmodule_archive_str</code>	1124 <code>\str_new:N \l__stex_importmodule_name_str</code>
<code>\l_stex_importmodule_path_str</code>	1125 <code>\str_new:N \l__stex_importmodule_archive_str</code>
<code>\l_stex_importmodule_file_str</code>	1126 <code>\str_new:N \l__stex_importmodule_path_str</code>
	1127 <code>\str_new:N \g__stex_importmodule_file_str</code>

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

```

\stex_import_require_module:nnnn      {\<ns>} {\<archive-ID>} {\<path>} {\<name>}
1128 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1129   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1130     % \stex_debug:n{Arguments: #1, #2, #3, #4}
1131
1132     % archive
1133     \str_set:Nx \l_tmpa_str { #2 }
1134     \str_if_empty:NTF \l_tmpa_str {
1135       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1136     } {
1137       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1138       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1139       \seq_put_right:Nn \l_tmpa_seq { source }
1140     }
1141
1142     % path
1143     \str_set:Nx \l_tmpb_str { #3 }
1144     \str_if_empty:NTF \l_tmpb_str {
1145       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1146
1147       \ltx@ifpackageloaded{babel} {
1148         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1149           { \language } \l_tmpb_str {
1150           \msg_set:nnn{stex}{error/unknownlanguage}{
1151             Unknown~language~\language
1152           }
1153           \msg_error:nn{stex}{error/unknownlanguage}
1154         }
1155       } {
1156         \str_clear:N \l_tmpb_str
1157       }
1158
1159       \stex_debug:n{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1160       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1161         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1162       }{
1163         \stex_debug:n{Checking~\l_tmpa_str.tex}
1164         \IfFileExists{ \l_tmpa_str.tex }{
1165           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1166         }{
1167           % try english as default
1168           \stex_debug:n{Checking~\l_tmpa_str.en.tex}
1169           \IfFileExists{ \l_tmpa_str.en.tex }{
1170             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1171           }{
1172             \msg_set:nnn{stex}{error/modulemissing}{
1173               No~file~for~module~#1?#4~found
1174             }
1175             \msg_error:nn{stex}{error/modulemissing}
1176           }
1177         }
1178       }
1179     } {
1180

```

```

1181 \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1182 \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1183
1184 \ltx@ifpackageloaded{babel} {
1185   \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1186     { \language } \l_tmpb_str {
1187       \msg_set:nnn{stex}{error/unknownlanguage}{
1188         Unknown~language~\language
1189       }
1190       \msg_error:nn{stex}{error/unknownlanguage}
1191     }
1192 } {
1193   \str_clear:N \l_tmpb_str
1194 }
1195
1196 \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1197
1198 \stex_debug:n{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1199 \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1200   \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1201 }{
1202   \stex_debug:n{Checking~\l_tmpa_str/#4.tex}
1203   \IfFileExists{ \l_tmpa_str/#4.tex }{
1204     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1205   }{
1206     % try english as default
1207     \stex_debug:n{Checking~\l_tmpa_str/#4.en.tex}
1208     \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1209       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1210     }{
1211       \stex_debug:n{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1212       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1213         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1214       }{
1215         \stex_debug:n{Checking~\l_tmpa_str.tex}
1216         \IfFileExists{ \l_tmpa_str.tex }{
1217           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1218         }{
1219           % try english as default
1220           \stex_debug:n{Checking~\l_tmpa_str.en.tex}
1221           \IfFileExists{ \l_tmpa_str.en.tex }{
1222             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1223           }{
1224             \msg_set:nnn{stex}{error/modulemissing}{
1225               No~file~for~module~#1?#4~found
1226             }
1227             \msg_error:nn{stex}{error/modulemissing}
1228           }
1229         }
1230       }
1231     }
1232   }
1233 }
1234 }

```

```

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

```

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

`\stex_activate_module:n`

```

1265 \cs_new_protected:Nn \stex_activate_module:n {
1266 \stex_debug:n{Activating~module~#1}
1267 \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1268 \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1269 \prop_item:cn { c_stex_module_#1_prop } { content }
1270 }
1271 }

```

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

`\importmodule`

```

1272 \NewDocumentCommand \importmodule { 0{} m } {
1273 \stex_import_module_uri:nn { #1 } { #2 }
1274 \stex_debug:n{Importing~module:~
1275 \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1276 }
1277 \stex_if_smsmode:F {
1278 \stex_import_require_module:nnnn
1279 { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1280 { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1281 \stex_annotate_invisible:nnn

```

```

1282     {import} {\l_stex_module_ns_str ? \l__stex_importmodule_name_str} {}
1283   }
1284   \exp_args:Nx \stex_add_to_current_module:n {
1285     \stex_import_require_module:nnnn
1286     { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1287     { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1288   }
1289   \exp_args:Nx \stex_add_import_to_current_module:n {
1290     \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1291   }
1292   \stex_smsmode_set_codes:
1293 }

```

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

`\usemodule`

```

1294 \NewDocumentCommand \usemodule { 0{} m } {
1295   \stex_if_smsmode:F {
1296     \stex_import_module_uri:nn { #1 } { #2 }
1297     \stex_import_require_module:nnnn
1298     { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1299     { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1300     \stex_annotate_invisible:nnn
1301     {usemodule} {\l_stex_module_ns_str ? \l__stex_importmodule_name_str} {}
1302   }
1303   \stex_smsmode_set_codes:
1304 }

```

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

`\g_stex_modules_in_file_seq` `\g_stex_module_files_prop`

```

1305 \seq_new:N \g_stex_modules_in_file_seq
1306 \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

```

1307 <@@=stex_symdecl>

```

`\l_stex_all_symbols_seq` Stores all available symbols

```

1308 \seq_new:N \l_stex_all_symbols_seq

```

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

`\STEXsymbol`

```

1309 \NewDocumentCommand \STEXsymbol { m } {
1310   \stex_get_symbol:n { #1 }
1311   \exp_args:No
1312   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1313 }

```

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

`symdecl` arguments:

```

1314 \keys_define:nn { stex / symdecl } {
1315   name      .tl_set:x:N = \l_stex_symdecl_name_str ,
1316   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1317   args      .tl_set:x:N = \l_stex_symdecl_args_str ,
1318   type      .tl_set:N   = \l_stex_symdecl_type_tl ,
1319   align     .tl_set:N   = \l_stex_symdecl_align_str , % TODO(?)
1320   gfc       .tl_set:N   = \l_stex_symdecl_gfc_str , % TODO(?)
1321   specializes .tl_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
1322   def       .tl_set:N   = \l_stex_symdecl_definiens_tl
1323 }
1324
1325 \bool_new:N \l_stex_symdecl_make_macro_bool
1326
1327 \cs_new_protected:Nn \__stex_symdecl_args:n {
1328   \str_clear:N \l_stex_symdecl_name_str
1329   \str_clear:N \l_stex_symdecl_args_str
1330   \bool_set_false:N \l_stex_symdecl_local_bool
1331   \tl_clear:N \l_stex_symdecl_type_tl
1332   \tl_clear:N \l_stex_symdecl_definiens_tl
1333 }
1334
1335 \keys_set:nn { stex / symdecl } { #1 }
1336
1337 \exp_args:NNo \str_set:Nn \l_stex_symdecl_name_str
1338   \l_stex_symdecl_name_str
1339 \exp_args:NNo \str_set:Nn \l_stex_symdecl_args_str
1340   \l_stex_symdecl_args_str
1341 }

```

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

```

1341
1342 \NewDocumentCommand \symdecl { s O{} m } {
1343   \__stex_symdecl_args:n { #2 }
1344   \IfBooleanTF #1 {
1345     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1346   } {
1347     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1348   }
1349   \stex_symdecl_do:n { #3 }
1350   \stex_smsmode_set_codes:
1351 }

```

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

`\stex_symdecl_do:n`

```

1352 \cs_new_protected:Nn \stex_symdecl_do:n {
1353   \stex_if_in_module:F {
1354     % TODO throw error? some default namespace?
1355   }
1356
1357   \str_if_empty:NT \l_stex_symdecl_name_str {

```



```

1358   \str_set:Nx \l_stex_symdecl_name_str { #1 }
1359 }
1360
1361 \prop_if_exist:cT { g_stex_symdecl_
1362   \prop_item:Nn \l_stex_current_module_prop {ns} ?
1363   \prop_item:Nn \l_stex_current_module_prop {name} ?
1364   \l_stex_symdecl_name_str
1365   _prop
1366 }{
1367   % TODO throw error (beware of circular dependencies)
1368 }
1369
1370 \prop_clear:N \l_tmpa_prop
1371 \prop_put:Nnx \l_tmpa_prop { module } {
1372   \prop_item:Nn \l_stex_current_module_prop {ns} ?
1373   \prop_item:Nn \l_stex_current_module_prop {name}
1374 }
1375 \seq_clear:N \l_tmpa_seq
1376 \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
1377 \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1378 \prop_put:Nno \l_tmpa_prop { local } \l_stex_symdecl_local_bool
1379 \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1380
1381 \exp_args:No \stex_add_constant_to_current_module:n {
1382   \l_stex_symdecl_name_str
1383 }
1384
1385 % arity/args
1386 \int_zero:N \l_tmpb_int
1387
1388 \bool_set_true:N \l_tmpa_bool
1389 \str_map_inline:Nn \l_stex_symdecl_args_str {
1390   \token_case_meaning:NnF ##1 {
1391     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1392     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1393     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1394     {\tl_to_str:n a} {
1395       \bool_set_false:N \l_tmpa_bool
1396       \int_incr:N \l_tmpb_int
1397     }
1398     {\tl_to_str:n B} {
1399       \bool_set_false:N \l_tmpa_bool
1400       \int_incr:N \l_tmpb_int
1401     }
1402   }{
1403     \msg_set:nnn{stex}{error/wrongargs}{
1404       args~value~in~symbol~declaration~for~
1405       \prop_item:Nn \l_stex_current_module_prop {ns} ?
1406       \prop_item:Nn \l_stex_current_module_prop {name} ?
1407       \l_stex_symdecl_name_str ~
1408       needs~to~be~
1409       i,~a,~b~or~B,~but~##1~given
1410     }
1411     \msg_error:nn{stex}{error/wrongargs}

```

```

1412     }
1413 }
1414 \bool_if:NTF \l_tmpa_bool {
1415   % possibly numeric
1416   \str_if_empty:NTF \l_stex_symdecl_args_str {
1417     \prop_put:Nnn \l_tmpa_prop { args } {}
1418     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1419   }{
1420     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1421     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1422     \str_clear:N \l_tmpa_str
1423     \int_step_inline:nn \l_tmpa_int {
1424       \str_put_right:Nn \l_tmpa_str i
1425     }
1426     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1427   }
1428 } {
1429   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1430   \prop_put:Nnx \l_tmpa_prop { arity }
1431   { \str_count:N \l_stex_symdecl_args_str }
1432 }
1433 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
1434
1435
1436 % semantic macro
1437
1438 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1439   \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1440     \prop_item:Nn \l_tmpa_prop { module } ?
1441     \prop_item:Nn \l_tmpa_prop { name }
1442   } }
1443
1444   \bool_if:NF \l_stex_symdecl_local_bool {
1445     \exp_args:Nx \stex_add_to_current_module:n {
1446       \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1447         \prop_item:Nn \l_tmpa_prop { module } ?
1448         \prop_item:Nn \l_tmpa_prop { name }
1449       } }
1450     }
1451   }
1452 }
1453
1454 % add to all symbols
1455
1456 \bool_if:NF \l_stex_symdecl_local_bool {
1457   \exp_args:Nx \stex_add_to_current_module:n {
1458     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1459       \prop_item:Nn \l_tmpa_prop { module } ?
1460       \prop_item:Nn \l_tmpa_prop { name }
1461     }
1462   }
1463 }
1464
1465 \stex_debug:n{New~symbol:~

```

```

1466 \prop_item:Nn \l_tmpa_prop { module } ?
1467 \prop_item:Nn \l_tmpa_prop { name }^^J
1468 Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
1469 Args:~\prop_item:Nn \l_tmpa_prop { args }
1470 }
1471
1472 % circular dependencies require this:
1473
1474 \prop_if_exist:cF {
1475   g_stex_symdecl_
1476   \prop_item:Nn \l_tmpa_prop { module } ?
1477   \prop_item:Nn \l_tmpa_prop { name }
1478   _prop
1479 } {
1480   \prop_gset_eq:cN {
1481     g_stex_symdecl_
1482     \prop_item:Nn \l_tmpa_prop { module } ?
1483     \prop_item:Nn \l_tmpa_prop { name }
1484     _prop
1485   } \l_tmpa_prop
1486 }
1487
1488 \stex_if_smsmode:TF {
1489   \bool_if:NF \l_stex_symdecl_local_bool {
1490     \exp_args:Nx \stex_addtosms:n {
1491       \prop_gset_from_keyval:cn {
1492         g_stex_symdecl_
1493         \prop_item:Nn \l_tmpa_prop { module } ?
1494         \prop_item:Nn \l_tmpa_prop { name }
1495         _prop
1496       } {
1497         name      = \prop_item:Nn \l_tmpa_prop { name }      ,
1498         module    = \prop_item:Nn \l_tmpa_prop { module }    ,
1499         notations = \prop_item:Nn \l_tmpa_prop { notations } ,
1500         local     = \prop_item:Nn \l_tmpa_prop { local }     ,
1501         type      = \prop_item:Nn \l_tmpa_prop { type }      ,
1502         args      = \prop_item:Nn \l_tmpa_prop { args }      ,
1503         arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
1504         assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
1505       }
1506       \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1507         \prop_item:Nn \l_tmpa_prop { module } ?
1508         \prop_item:Nn \l_tmpa_prop { name }
1509       }
1510     }
1511   }
1512 }{
1513   \exp_args:NNx \seq_put_right:Nn \l_stex_all_symbols_seq {
1514     \prop_item:Nn \l_tmpa_prop { module } ?
1515     \prop_item:Nn \l_tmpa_prop { name }
1516   }
1517   \stex_annotate_invisible:nnn {symdecl} {
1518     \prop_item:Nn \l_tmpa_prop { module } ?
1519     \prop_item:Nn \l_tmpa_prop { name }

```

```

1520 } {
1521   \stex_annotate_invisible:nnn{type}{-}{\l_stex_symdecl_type_tl$}
1522   \stex_annotate_invisible:nnn{args}{-}{
1523     \prop_item:Nn \l_tmpa_prop { args }
1524   }
1525   \stex_annotate_invisible:nnn{macroname}{-}{#1}
1526   \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
1527     \stex_annotate_invisible:nnn{definiens}{-}{
1528       {\l_stex_symdecl_definiens_tl$}
1529     }
1530   }
1531 }
1532 }

```

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

`\stex_get_symbol:n`

```

1533 \str_new:N \l_stex_get_symbol_uri_str
1534
1535 \cs_new_protected:Nn \stex_get_symbol:n {
1536   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
1537     \__stex_symdecl_get_symbol_from_cs:n { #1 }
1538   }{
1539     % argument is a string
1540     % is it a command name?
1541     \cs_if_exist:cTF { #1 }{
1542       \cs_set_eq:Nc \l_tmpa_tl { #1 }
1543       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
1544       \str_if_empty:NNTF \l_tmpa_str {
1545         \exp_args:Nx \cs_if_eq:NNTF {
1546           \tl_head:N \l_tmpa_tl
1547         } \stex_invoke_symbol:n {
1548           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
1549         }{
1550           \__stex_symdecl_get_symbol_from_string:n { #1 }
1551         }
1552       } {
1553         \__stex_symdecl_get_symbol_from_string:n { #1 }
1554       }
1555     }{
1556       % argument is not a command name
1557       \__stex_symdecl_get_symbol_from_string:n { #1 }
1558       % \l_stex_all_symbols_seq
1559     }
1560   }
1561 }
1562
1563 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
1564   \prop_get:NnN \l_stex_current_module_prop
1565   { constants } \l_tmpa_seq
1566   \seq_if_in:NnTF \l_tmpa_seq { #1 } {
1567     \str_set:Nx \l_stex_get_symbol_uri_str {
1568       \prop_item:Nn \l_stex_current_module_prop { ns } ?
1569       \prop_item:Nn \l_stex_current_module_prop { name } ? #1

```

```

1570 }
1571 } {
1572   \tl_set:Nn \l_tmpa_tl {
1573     \msg_set:nnn{stex}{error/unknownsymbol}{
1574       No~symbol~#1~found!
1575     }
1576     \msg_error:nn{stex}{error/unknownsymbol}
1577   }
1578   \str_set:Nn \l_tmpa_str { #1 }
1579   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1580   \seq_map_inline:Nn \l_stex_all_symbols_seq {
1581     \str_set:Nn \l_tmpb_str { ##1 }
1582     \str_if_eq:eeT { \l_tmpa_str } {
1583       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1584     } {
1585       \seq_map_break:n {
1586         \tl_set:Nn \l_tmpa_tl {
1587           \str_set:Nn \l_stex_get_symbol_uri_str {
1588             ##1
1589           }
1590         }
1591       }
1592     }
1593   }
1594   \l_tmpa_tl
1595 }
1596 }
1597
1598 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
1599   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
1600     { \tl_tail:N \l_tmpa_tl }
1601   \tl_if_single:NTF \l_tmpa_tl {
1602     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
1603       \exp_after:wN \str_set:Nn \exp_after:wN
1604         \l_stex_get_symbol_uri_str \l_tmpa_tl
1605     }{
1606       % TODO
1607       % tail is not a single group
1608     }
1609   }{
1610     % TODO
1611     % tail is not a single group
1612   }
1613 }

```

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

4.7 Notations

```

1614 <@@=stex_notation>
      notation arguments:
1615 \keys_define:nn { stex / notation } {
1616   lang      .tl_set_x:N = \l__stex_notation_lang_str ,
1617   variant   .tl_set_x:N = \l__stex_notation_variant_str ,

```

```

1618 prec      .tl_set_x:N = \l__stex_notation_prec_str ,
1619 op        .tl_set:N   = \l__stex_notation_op_tl ,
1620 unknown   .code:n      = \str_set:Nx
1621           \l__stex_notation_variant_str \l_keys_key_str
1622 }
1623
1624 \cs_new_protected:Nn \__stex_notation_args:n {
1625   \str_clear:N \l__stex_notation_lang_str
1626   \str_clear:N \l__stex_notation_variant_str
1627   \str_clear:N \l__stex_notation_prec_str
1628   \tl_clear:N \l__stex_notation_op_tl
1629
1630   \keys_set:nn { stex / notation } { #1 }
1631
1632   \str_set:Nx \l__stex_notation_lang_str \l__stex_notation_lang_str
1633   \str_set:Nx \l__stex_notation_variant_str \l__stex_notation_variant_str
1634   \str_set:Nx \l__stex_notation_prec_str \l__stex_notation_prec_str
1635 }

```

\notation

```

1636 \NewDocumentCommand \notation { 0{} m } {
1637   \__stex_notation_args:n { #1 }
1638   \tl_clear:N \l_stex_symdecl_definiens_tl
1639   \stex_get_symbol:n { #2 }
1640   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
1641 }

```

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

\stex_notation_do:nn

```

1642 \cs_new_protected:Nn \stex_notation_do:nn {
1643   \prop_set_eq:Nc \l_tmpa_prop {
1644     g_stex_symdecl_ #1 _prop
1645   }
1646
1647   \prop_clear:N \l_tmpb_prop
1648   \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
1649   \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
1650   \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
1651
1652   % precedences
1653   \seq_clear:N \l_tmpb_seq
1654   \exp_args:NNno
1655   \str_if_empty:NTF \l__stex_notation_prec_str {
1656     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1657     \int_compare:nNnTF \l_tmpa_str = 0 {
1658       \exp_args:NNnx
1659       \prop_put:Nno \l_tmpb_prop { opprec }
1660       { \infprec }
1661     }{
1662       \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
1663     }
1664   } {
1665     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
1666       \exp_args:NNnx

```

```

1667 \prop_put:Nno \l_tmpb_prop { opprec }
1668 { \infprec }
1669 \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1670 \int_step_inline:nn { \l_tmpa_str } {
1671   \exp_args:NNx
1672   \seq_put_right:Nn \l_tmpb_seq { \neginfprec }
1673 }
1674 }{
1675   \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
1676   \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
1677     \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
1678     \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
1679       \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
1680       \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
1681       \seq_map_inline:Nn \l_tmpa_seq {
1682         \seq_put_right:Nn \l_tmpb_seq { ##1 }
1683       }
1684     }
1685     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1686   }{
1687     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1688     \int_compare:nNnTF \l_tmpa_str = 0 {
1689       \exp_args:NNnx
1690       \prop_put:Nno \l_tmpb_prop { opprec }
1691       { \infprec }
1692     }{
1693       \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
1694     }
1695   }
1696 }
1697 }
1698
1699 \seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
1700 \int_step_inline:nn { \l_tmpa_str } {
1701   \seq_pop_left:NNTF \l_tmpa_seq \l_tmpb_str {
1702     \exp_args:NNx
1703     \seq_put_right:Nn \l_tmpb_seq {
1704       \prop_item:Nn \l_tmpb_prop { opprec }
1705     }
1706   }
1707 }
1708
1709 \prop_put:Nno \l_tmpb_prop { argprec } \l_tmpb_seq
1710 \tl_clear:N \l_tmpa_tl
1711
1712 \int_compare:nNnTF \l_tmpa_str = 0 {
1713   \exp_args:NNe
1714   \cs_set:Npn \l__stex_notation_macrocode_cs {
1715     \_stex_term_math_oms:nnnn { #1 }
1716     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1717     { \prop_item:Nn \l_tmpb_prop { opprec } }
1718     { \exp_not:n { #2 } }
1719   }
1720   \__stex_notation_final:

```

```

1721 }{
1722   \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
1723   \str_if_in:NnTF \l_tmpb_str b {
1724     \exp_args:Nne \use:nn
1725     {
1726       \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1727       \cs_set:Npn \l_tmpa_str } { {
1728         \stex_term_math_omb:nnnn { #1 }
1729         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1730         { \prop_item:Nn \l_tmpb_prop { opprec } }
1731         { \exp_not:n { #2 } }
1732       } }
1733   }{
1734     \str_if_in:NnTF \l_tmpb_str B {
1735       \exp_args:Nne \use:nn
1736       {
1737         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1738         \cs_set:Npn \l_tmpa_str } { {
1739           \stex_term_math_omb:nnnn { #1 }
1740           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1741           { \prop_item:Nn \l_tmpb_prop { opprec } }
1742           { \exp_not:n { #2 } }
1743         } }
1744     }{
1745       \exp_args:Nne \use:nn
1746       {
1747         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1748         \cs_set:Npn \l_tmpa_str } { {
1749           \stex_term_math_oma:nnnn { #1 }
1750           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1751           { \prop_item:Nn \l_tmpb_prop { opprec } }
1752           { \exp_not:n { #2 } }
1753         } }
1754     }
1755   }
1756
1757   \int_zero:N \l_tmpa_int
1758   \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
1759   \prop_get:NnN \l_tmpb_prop { argprecs } \l_tmpa_seq
1760   \__stex_notation_arguments:
1761 }
1762 }

```

(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

```

1763 \cs_new_protected:Nn \__stex_notation_arguments: {
1764   \int_incr:N \l_tmpa_int
1765   \str_if_empty:NnTF \l_tmpa_str {
1766     \__stex_notation_final:
1767   }{
1768     \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
1769     \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
1770     \str_if_eq:VnTF \l_tmpb_str a {

```



```

1771     \__stex_notation_argument_assoc:n
1772   }{
1773     \str_if_eq:VnTF \l_tmpb_str B {
1774       \__stex_notation_argument_assoc:n
1775     }{
1776       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1777       \tl_put_right:Nx \l_tmpa_tl {
1778         { \stex_term_math_arg:nnn
1779           { \int_use:N \l_tmpa_int }
1780           { \l_tmpb_str }
1781           { ####\int_use:N \l_tmpa_int }
1782         }
1783       }
1784       \__stex_notation_arguments:
1785     }
1786   }
1787 }
1788 }

```

(End definition for __stex_notation_arguments:.)

__stex_notation_argument_assoc:n

```

1789 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
1790   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1791   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
1792   \tl_put_right:Nx \l_tmpa_tl {
1793     { \stex_term_math_assoc_arg:nnnn
1794       { \int_use:N \l_tmpa_int }
1795       { \l_tmpb_str }
1796       \exp_args:No \exp_not:n
1797       {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
1798       { ####\int_use:N \l_tmpa_int }
1799     }
1800   }
1801   \__stex_notation_arguments:
1802 }

```

(End definition for __stex_notation_argument_assoc:n.)

__stex_notation_final: Called after processing all notation arguments

```

1803 \cs_new_protected:Nn \__stex_notation_final: {
1804   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
1805   \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
1806   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
1807   \exp_args:Nne \use:nn
1808   {
1809     \cs_generate_from_arg_count:cNnn {
1810       stex_notation_ \l_tmpa_str \c_hash_str
1811       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1812       _cs
1813     }
1814     \cs_gset:Npn \l_tmpb_str { { {
1815       \exp_after:wN \exp_after:wN \exp_after:wN
1816       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN

```

```

1817     { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
1818 } }
1819
1820 \tl_if_empty:NF \l__stex_notation_op_tl {
1821   \cs_gset:cpx {
1822     stex_op_notation_ \l_tmpa_str \c_hash_str
1823     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1824     _cs
1825   } {
1826     \stex_term_oms:nnn {
1827       \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
1828       \l__stex_notation_lang_str
1829     }{
1830       \l_tmpa_str
1831     }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } } }
1832   }
1833 }
1834
1835
1836
1837 \stex_debug:n{
1838   Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1839   ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
1840   Operator~precedence:~
1841   \prop_item:Nn \l_tmpb_prop { opprec }^^J
1842   Argument~precedences:~
1843   \seq_use:Nn \l_tmpa_seq {,~}^^J
1844   Notation: \cs_meaning:c {
1845     stex_notation_ \l_tmpa_str \c_hash_str
1846     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1847     _cs
1848   }
1849 }
1850
1851 \prop_gset_eq:cN {
1852   g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
1853   \c_hash_str \l__stex_notation_lang_str _prop
1854 } \l_tmpb_prop
1855
1856 \exp_args:Nx
1857 \stex_add_to_current_module:n {
1858   \prop_get:cnN {
1859     g_stex_symdecl_
1860     \prop_item:Nn \l_tmpb_prop { symbol }
1861     _prop
1862   } { notations } \exp_not:N \l_tmpa_seq
1863   \seq_put_right:Nn \exp_not:N \l_tmpa_seq {
1864     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1865   }
1866   \prop_put:cno {
1867     g_stex_symdecl_
1868     \prop_item:Nn \l_tmpb_prop { symbol }
1869     _prop
1870   } { notations } \exp_not:N \l_tmpa_seq

```

```

1871 }
1872
1873 \stex_if_smsmode:TF {
1874   \stex_smsmode_set_codes:
1875   \exp_args:Nx \stex_addtosms:n {
1876     \prop_gset_from_keyval:cn {
1877       g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
1878       \c_hash_str \l__stex_notation_lang_str _prop
1879     } {
1880       symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
1881       language    = \prop_item:Nn \l_tmpb_prop { language }    ,
1882       variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,
1883       opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
1884       argprec     = \prop_item:Nn \l_tmpb_prop { argprec }     ,
1885     }
1886   }
1887 }{
1888   \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
1889   \seq_put_right:Nx \l_tmpa_seq {
1890     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1891   }
1892   \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
1893   \prop_set_eq:cN {
1894     g_stex_symdecl_ \l_tmpa_str _prop
1895   } \l_tmpa_prop
1896
1897   % HTML annotations
1898   \stex_annotate_invisible:nnn { notation }
1899   { \prop_item:Nn \l_tmpb_prop { symbol } } {
1900     \stex_annotate_invisible:nnn { notationfragment }
1901     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
1902     \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
1903     \stex_annotate_invisible:nnn { precedence }
1904     { \prop_item:Nn \l_tmpb_prop { opprec } ;
1905       \seq_use:Nn \l_tmpa_seq { x }
1906     }{}
1907
1908     \int_zero:N \l_tmpa_int
1909     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
1910     \tl_clear:N \l_tmpa_tl
1911     \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{}{
1912       \int_incr:N \l_tmpa_int
1913       \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
1914       \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
1915       \str_if_eq:VnTF \l_tmpb_str a {
1916         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
1917           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
1918           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
1919         } }
1920       }{
1921         \str_if_eq:VnTF \l_tmpb_str B {
1922           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
1923             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
1924             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b

```

```

1925         } }
1926     }{
1927         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
1928             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
1929         } }
1930     }
1931 }
1932 }
1933 \stex_annotate_invisible:nnn { notationcomp }{}{
1934     $ \exp_args:Nno \use:nn { \use:c {
1935         stex_notation_ \prop_item:Nn \l_tmpb_prop { symbol }
1936         \c_hash_str \l__stex_notation_variant_str
1937         \c_hash_str \l__stex_notation_lang_str_cs
1938     } } { \l_tmpa_tl } $
1939 }
1940 }
1941 }
1942 }

```

(End definition for `_stex_notation_final:`)

\symdef

```

1943 \keys_define:nn { stex / symdef } {
1944     name .tl_set:N = \l_stex_symdecl_name_str ,
1945     local .bool_set:N = \l_stex_symdecl_local_bool ,
1946     args .tl_set:N = \l_stex_symdecl_args_str ,
1947     type .tl_set:N = \l_stex_symdecl_type_tl ,
1948     def .tl_set:N = \l_stex_symdecl_definiens_tl ,
1949     op .tl_set:N = \l__stex_notation_op_tl ,
1950     lang .tl_set:N = \l__stex_notation_lang_str ,
1951     variant .tl_set:N = \l__stex_notation_variant_str ,
1952     prec .tl_set:N = \l__stex_notation_prec_str ,
1953     unknown .code:n = \str_set:Nx
1954         \l__stex_notation_variant_str \l_keys_key_str
1955 }
1956
1957 \cs_new_protected:Nn \_stex_notation_symdef_args:n {
1958     \str_clear:N \l_stex_symdecl_name_str
1959     \str_clear:N \l_stex_symdecl_args_str
1960     \bool_set_false:N \l_stex_symdecl_local_bool
1961     \tl_clear:N \l_stex_symdecl_type_tl
1962     \tl_clear:N \l_stex_symdecl_definiens_tl
1963     \str_clear:N \l__stex_notation_lang_str
1964     \str_clear:N \l__stex_notation_variant_str
1965     \str_clear:N \l__stex_notation_prec_str
1966     \tl_clear:N \l__stex_notation_op_tl
1967
1968     \keys_set:nn { stex / symdef } { #1 }
1969
1970     \exp_args:Nno \str_set:Nn \l_stex_symdecl_name_str
1971         \l_stex_symdecl_name_str
1972     \exp_args:Nno \str_set:Nn \l_stex_symdecl_args_str
1973         \l_stex_symdecl_args_str
1974     \exp_args:Nno \str_set:Nn \l__stex_notation_lang_str

```

```

1975     \l__stex_notation_lang_str
1976 \exp_args:NNo \str_set:Nn \l__stex_notation_variant_str
1977     \l__stex_notation_variant_str
1978 \exp_args:NNo \str_set:Nn \l__stex_notation_prec_str
1979     \l__stex_notation_prec_str
1980 }
1981
1982 \NewDocumentCommand \symdef { 0{} m } {
1983   \__stex_notation_symdef_args:n { #1 }
1984   \bool_set_true:N \l_stex_symdecl_make_macro_bool
1985   \stex_symdecl_do:n { #2 }
1986   \exp_args:Nx \stex_notation_do:nn {
1987     \prop_item:Nn \l_tmpa_prop { module } ?
1988     \prop_item:Nn \l_tmpa_prop { name }
1989   }
1990 }

```

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

`\stex_invoke_symbol:n` Invokes a semantic macro

```

1991 %\cs_new_protected:Nn \stex_invoke_symbol:n {
1992 %   \peek_charcode_remove:NTF ! {
1993 %     \stex_term_custom:nn { #1 } { }
1994 %   } {
1995 %     \if_mode_math:
1996 %       \exp_after:wN \__stex_notation_invoke_math:n
1997 %     \else:
1998 %       \exp_after:wN \__stex_notation_invoke_text:n
1999 %     \fi: { #1 }
2000 %   }
2001 %}
2002
2003 \cs_new_protected:Nn \stex_invoke_symbol:n {
2004   \if_mode_math:
2005     \exp_after:wN \__stex_notation_invoke_math:n
2006   \else:
2007     \exp_after:wN \__stex_notation_invoke_text:n
2008   \fi: { #1 }
2009 }

```

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

`__stex_notation_invoke_math:n`

```

2010 \cs_new_protected:Nn \__stex_notation_invoke_math:n {
2011   \peek_charcode_remove:NTF ! {
2012     \peek_charcode:NTF [ {
2013       \__stex_notation_invoke_op:nw { #1 }
2014     }{
2015       \__stex_notation_invoke_op:nw { #1 } []
2016     }
2017   }{
2018     \peek_charcode_remove:NTF * {
2019       \__stex_notation_invoke_text:n { #1 }
2020     }{

```

```

2021     \peek_charcode:NTF [ {
2022       \__stex_notation_invoke_math:nw { #1 }
2023     }{
2024       \__stex_notation_invoke_math:nw { #1 } []
2025     }
2026   }
2027 }
2028 }

```

(End definition for __stex_notation_invoke_math:n.)

__stex_notation_invoke_op:nw

```

2029 \cs_new_protected:Npn \__stex_notation_invoke_op:nw #1 [#2] {
2030   \__stex_notation_args:n { #2 }
2031   \cs_if_exist:cTF {
2032     stex_op_notation_ #1 \c_hash_str
2033     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str _cs
2034   }{
2035     \csname stex_op_notation_ #1 \c_hash_str
2036       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str _cs
2037     \endcsname
2038   }{
2039     % TODO throw error
2040   }
2041 }

```

(End definition for __stex_notation_invoke_op:nw.)

__stex_notation_invoke_math:nw

```

2042 \cs_new_protected:Npn \__stex_notation_invoke_math:nw #1 [#2] {
2043   \__stex_notation_args:n { #2 }
2044   \prop_set_eq:Nc \l_tmpa_prop {
2045     g_stex_symdecl_ #1 _prop
2046   }
2047   \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
2048   \seq_if_empty:NTF \l_tmpa_seq {
2049     \msg_set:nnn{stex}{error/nonotations}{
2050       Symbol~#1~used,~but~has~no~notations!
2051     }
2052     \msg_error:nn{stex}{error/nonotations}
2053   } {
2054     \seq_if_in:NxTF \l_tmpa_seq
2055     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2056       \use:c{
2057         stex_notation_ #1 \c_hash_str
2058         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2059         _cs
2060       }
2061     }{
2062       \str_if_empty:NTF \l__stex_notation_variant_str {
2063         \str_if_empty:NTF \l__stex_notation_lang_str {
2064           \seq_get_left:NN \l_tmpa_seq \l_tmpa_str
2065           \use:c{
2066             stex_notation_ #1 \c_hash_str \l_tmpa_str
2067             _cs

```

```

2068     }
2069   }{
2070     \msg_set:nnn{stex}{error/wrongnotation}{
2071       Symbol~#1~has~no~notation~
2072       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2073     }
2074     \msg_error:nn{stex}{error/wrongnotation}
2075   }
2076 }{
2077   \msg_set:nnn{stex}{error/wrongnotation}{
2078     Symbol~#1~has~no~notation~
2079     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2080   }
2081   \msg_error:nn{stex}{error/wrongnotation}
2082 }
2083 }
2084 }
2085 }

```

(End definition for `__stex_notation_invoke_math:nw`.)

`__stex_notation_invoke_text:n`

```

2086 \cs_new_protected:Nn \__stex_notation_invoke_text:n {
2087   \peek_charcode_remove:NTF ! {
2088     \stex_term_custom:nn { #1 } { }
2089   }{
2090     \prop_set_eq:Nc \l_tmpa_prop {
2091       g_stex_symdecl_ #1 _prop
2092     }
2093     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2094     \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2095   }
2096 }

```

(End definition for `__stex_notation_invoke_text:n`.)

4.8 Terms

2097 `<@@=stex_term`)

Precedences:

```

\infprec
\neginfprec
\l__stex_term_downprec
2098 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2099 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2100 \int_new:N \l__stex_term_downprec
2101 \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
2102 \tl_set:Nn \l__stex_term_left_bracket_str (
2103 \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

```
2104 \cs_new_protected:Nn \_stex_term_maybe_brackets:nn {
2105   \int_compare:nNnTF { #1 } > \l__stex_term_downprec {
2106     \bool_if:NTF \l_stex_inpararray_bool { #2 }{
2107       \dobrackets { #2 }
2108     }
2109   }{ #2 }
2110 }
```

(End definition for `_stex_term_maybe_brackets:nn`.)

`\dobrackets`

```
2111 %\RequirePackage{scalerel}
2112 \cs_new_protected:Npn \dobrackets #1 {
2113   %\ThisStyle{\if D\m@switch
2114   %   \exp_args:Nnx \use:nn
2115   %   { \exp_after:wN \left\l__stex_term_left_bracket_str #1 }
2116   %   { \exp_not:N\right\l__stex_term_right_bracket_str }
2117   %   \else
2118   %   \exp_args:Nnx \use:nn
2119   %   { \l__stex_term_left_bracket_str #1 }
2120   %   { \l__stex_term_right_bracket_str }
2121   %\fi}
2122 }
```

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

`\withbrackets`

```
2123 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2124   \exp_args:Nnx \use:nn
2125   {
2126     \tl_set:Nx \l__stex_term_left_bracket_str { #1 }
2127     \tl_set:Nx \l__stex_term_right_bracket_str { #2 }
2128     #3
2129   }
2130   {
2131     \tl_set:Nn \exp_not:N \l__stex_term_left_bracket_str
2132     { \l__stex_term_left_bracket_str }
2133     \tl_set:Nn \exp_not:N \l__stex_term_right_bracket_str
2134     { \l__stex_term_right_bracket_str }
2135   }
2136 }
```

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

`\STEXinvisible`

```
2137 \cs_new_protected:Npn \STEXinvisible #1 {
2138   \stex_annotate_invisible:n { #1 }
2139 }
```

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

OMDOC terms:

`_stex_term_math_oms:nnnn`

```

2140 \cs_new_protected:Nn \_stex_term_oms:nnn {
2141   \stex_annotate:nnn{ OMID }{ #2 }{
2142     \stex_highlight_term:nn { #1 } { #3 }
2143   }
2144 }
2145
2146 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2147   \_stex_term_maybe_brackets:nn { #3 }{
2148     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2149   }
2150 }

```

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

`_stex_term_math_oma:nnnn`

```

2151 \cs_new_protected:Nn \_stex_term_oma:nnn {
2152   \stex_annotate:nnn{ OMA }{ #2 }{
2153     \stex_highlight_term:nn { #1 } { #3 }
2154   }
2155 }
2156
2157 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2158   \_stex_term_maybe_brackets:nn { #3 }{
2159     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2160   }
2161 }

```

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

`_stex_term_math_omb:nnnn`

```

2162 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2163   \stex_annotate:nnn{ OMBIND }{ #2 }{
2164     \stex_highlight_term:nn { #1 } { #3 }
2165   }
2166 }
2167
2168 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2169   \_stex_term_maybe_brackets:nn { #3 }{
2170     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2171   }
2172 }

```

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

`_stex_term_math_arg:nnn`

```

2173 \cs_new_protected:Nn \_stex_term_arg:nn {
2174   \stex_unhighlight_term:n {
2175     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2176   }
2177 }
2178 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
2179   \exp_args:Nnx \use:nn
2180   { \int_set:Nn \l__stex_term_downprec { #2 }

```

```

2181     \stex_term_arg:nn { #1 }{ #3 }
2182   }
2183   { \int_set:Nn \exp_not:N \l__stex_term_downprec { \int_use:N \l__stex_term_downprec } }
2184 }

```

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

`\stex_term_math_assoc_arg:nnnn`

```

2185 \cs_new_protected:Nn \stex_term_math_assoc_arg:nnnn {
2186   \seq_set_split:Nnn \l_tmpa_seq , { #4 }
2187   \int_compare:nNnTF { \seq_count:N \l_tmpa_seq } < 2 {
2188     \tl_set:Nn \l_tmpa_tl { #4 }
2189   }{
2190     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
2191     \seq_reverse:N \l_tmpa_seq
2192     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_tl
2193     \tl_set:No \l_tmpa_tl { \l_tmpb_tl }
2194
2195     \seq_map_inline:Nn \l_tmpa_seq {
2196       \exp_args:NNo \tl_set:No \l_tmpa_tl {
2197         \exp_args:Nno
2198         \l_tmpa_cs { ##1 } \l_tmpa_tl
2199       }
2200     }
2201
2202   }
2203   \exp_args:Nnno
2204   \stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2205 }

```

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

`\stex_term_custom:nn`

```

2206 \cs_new_protected:Nn \stex_term_custom:nn {
2207   \str_set:Nn \l__stex_term_custom_uri { #1 }
2208   \str_set:Nn \l_tmpa_str { #2 }
2209   \tl_clear:N \l_tmpa_tl
2210   \int_zero:N \l_tmpa_int
2211   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2212   \__stex_term_custom_loop:
2213 }

```

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

`__stex_term_custom_loop:`

```

2214 \cs_new_protected:Nn \__stex_term_custom_loop: {
2215   \bool_set_false:N \l_tmpa_bool
2216   \bool_while_do:nn {
2217     \str_if_eq_p:ee X {
2218       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2219     }
2220   }{
2221     \int_incr:N \l_tmpa_int
2222   }
2223 }

```

```

2224 \peek_charcode:NTF [ {
2225   % notation/text component
2226   \__stex_term_custom_component:w
2227 } {
2228   \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2229     % all arguments read => finish
2230     \__stex_term_custom_final:
2231   } {
2232     % arguments missing
2233     \peek_charcode_remove:NTF * {
2234       % invisible, specific argument position or both
2235       \peek_charcode:NTF [ {
2236         % visible specific argument position
2237         \__stex_term_custom_arg:wn
2238       } {
2239         % invisible
2240         \peek_charcode_remove:NTF * {
2241           % invisible specific argument position
2242           \__stex_term_custom_arg_inv:wn
2243         } {
2244           % invisible next argument
2245           \__stex_term_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2246         }
2247       }
2248     } {
2249       % next normal argument
2250       \__stex_term_custom_arg:wn [ \l_tmpa_int + 1 ]
2251     }
2252   }
2253 }
2254 }

```

(End definition for __stex_term_custom_loop:.)

__stex_term_custom_arg_inv:wn

```

2255 \cs_new_protected:Npn \__stex_term_custom_arg_inv:wn [ #1 ] #2 {
2256   \bool_set_true:N \l_tmpa_bool
2257   \__stex_term_custom_arg:wn [ #1 ] { #2 }
2258 }

```

(End definition for __stex_term_custom_arg_inv:wn.)

__stex_term_custom_arg:wn

```

2259 \cs_new_protected:Npn \__stex_term_custom_arg:wn [ #1 ] #2 {
2260   \str_set:Nx \l_tmpb_str {
2261     \str_item:Nn \l_tmpa_str { #1 }
2262   }
2263   \str_case:VnTF \l_tmpb_str {
2264     { X } { } % TODO throw error ?
2265     { i } { \__stex_term_custom_set_X:n { #1 } }
2266     { b } { \__stex_term_custom_set_X:n { #1 } }
2267     { a } { \__stex_term_custom_set_X:n { #1 } } % TODO ?
2268     { B } { \__stex_term_custom_set_X:n { #1 } } % TODO ?
2269   }{}{
2270     % TODO throw error

```

```

2271 }
2272
2273 \bool_if:nTF \l_tmpa_bool {
2274   \tl_put_right:Nx \l_tmpa_tl {
2275     \stex_annotate_invisible:n {
2276       \stex_term_arg:nn { \int_eval:n { #1 } }
2277       \exp_not:n { { #2 } }
2278     }
2279   }
2280 } {
2281   \tl_put_right:Nx \l_tmpa_tl {
2282     \stex_term_arg:nn { \int_eval:n { #1 } }
2283     \exp_not:n { { #2 } }
2284   }
2285 }
2286
2287 \__stex_term_custom_loop:
2288 }

```

(End definition for __stex_term_custom_arg:wn.)

__stex_term_custom_set_X:n

```

2289 \cs_new_protected:Nn \__stex_term_custom_set_X:n {
2290   \str_set:Nx \l_tmpa_str {
2291     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
2292     X
2293     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
2294   }
2295 }

```

(End definition for __stex_term_custom_set_X:n.)

__stex_term_custom_component:

```

2296 \cs_new_protected:Npn \__stex_term_custom_component:w [ #1 ] {
2297   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
2298   \__stex_term_custom_loop:
2299 }

```

(End definition for __stex_term_custom_component:.)

__stex_term_custom_final:

```

2300 \cs_new_protected:Nn \__stex_term_custom_final: {
2301   \int_compare:nNnTF \l_tmpb_int = 0 {
2302     \exp_args:Nnno \stex_term_oms:nnn
2303   } {
2304     \str_if_in:NnTF \l_tmpa_str {b} {
2305       \exp_args:Nnno \stex_term_ombind:nnn
2306     } {
2307       \exp_args:Nnno \stex_term_oma:nnn
2308     }
2309   }
2310   { \l__stex_term_custom_uri } { \l__stex_term_custom_uri } { \l_tmpa_tl }
2311 }

```

(End definition for __stex_term_custom_final:.)

```

\symref
\symname
2312 \NewDocumentCommand \symref { m m }{
2313   \STEXsymbol{#1}! [#2]
2314 }
2315
2316 \keys_define:nn { stex / symname } {
2317   post      .tl_set_x:N    = \l_stex_symname_post_str
2318 }
2319
2320 \cs_new_protected:Nn \stex_symname_args:n {
2321   \str_clear:N \l_stex_symname_post_str
2322   \keys_set:nn { stex / symname } { #1 }
2323   \exp_args:NNo \str_set:Nn \l_stex_symname_post_str
2324     \l_stex_symname_post_str
2325 }
2326
2327 \NewDocumentCommand \symname { 0{} m }{
2328   \stex_symname_args:n { #1 }
2329   \stex_get_symbol:n { #2 }
2330   \str_set:Nx \l_tmpa_str {
2331     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
2332   }
2333   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
2334   \exp_args:NNx \use:nn
2335   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }! [
2336     \l_tmpa_str \l_stex_symname_post_str
2337   ] }
2338 }

```

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

4.9 Notation Components

```

2339 <@@=stex_notationcomps>
\stex_highlight_term:nn
2340 \latexml_if:F {
2341   \scalatex_if:F{
2342     \RequirePackage{pdfcomment}
2343   }
2344 }
2345
2346 \str_new:N \l__stex_notationcomps_highlight_uri_str
2347 \cs_new_protected:Nn \stex_highlight_term:nn {
2348   \exp_args:Nnx
2349   \use:nn {
2350     \str_set:Nx \l__stex_notationcomps_highlight_uri_str { #1 }
2351     #2
2352   } {
2353     \str_set:Nx \exp_not:N \l__stex_notationcomps_highlight_uri_str
2354       { \l__stex_notationcomps_highlight_uri_str }
2355   }
2356 }
2357

```

```

2358 \cs_new_protected:Nn \stex_unhighlight_term:n {
2359 % \latexml_if:TF {
2360 %   #1
2361 % } {
2362 %   \scalatex_if:TF {
2363 %     #1
2364 %   } {
2365     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
2366 %   }
2367 % }
2368 }

```

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

```

\comp
\@comp
\@defemph
2369 \cs_new_protected:Npn \comp #1 {
2370   \str_if_empty:NF \l__stex_notationcomps_highlight_uri_str {
2371     \scalatex_if:TF {
2372       \stex_annotate:nnn { comp }{ \l__stex_notationcomps_highlight_uri_str }{ #1 }
2373     }{
2374       \exp_args:Nnx \@comp { #1 } { \l__stex_notationcomps_highlight_uri_str }
2375     }
2376   }
2377 }
2378
2379 \cs_new_protected:Npn \@comp #1 #2 {
2380   \pdftooltip {
2381     \textcolor{blue}{#1}
2382   } { #2 }
2383 }
2384
2385 \cs_new_protected:Npn \@defemph #1 #2 {
2386   \pdftooltip {
2387     \textbf{\textcolor{magenta}{#1}}
2388   } { #2 }
2389 }

```

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

`\ellipses`

```

2390 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix
\parrayline
\parraycell
2391 \bool_new:N \l_stex_inparray_bool
2392 \bool_set_false:N \l_stex_inparray_bool
2393 \NewDocumentCommand \parray { m m } {
2394   \begingroup
2395   \bool_set_true:N \l_stex_inparray_bool
2396   \begin{array}{#1}
2397     #2
2398   \end{array}
2399   \endgroup
2400 }

```

```

2401
2402 \NewDocumentCommand \prmatrix { m } {
2403   \begingroup
2404   \bool_set_true:N \l_stex_inarray_bool
2405   \begin{matrix}
2406     #1
2407   \end{matrix}
2408   \endgroup
2409 }
2410
2411 \def \parrayline #1 #2 {
2412   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
2413 }
2414
2415 \def \parraycell #1 {
2416   #1 \bool_if:NT \l_stex_inarray_bool {&}
2417 }

```

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

4.10 Structural Features

```

2418 <@@=stex_features>

```

symboldoc

```

2419 \NewDocumentEnvironment{symboldoc}{ m }{
2420   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
2421   \seq_clear:N \l_tmpb_seq
2422   \seq_map_inline:Nn \l_tmpa_seq {
2423     \stex_get_symbol:n { ##1 }
2424     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
2425       \l_stex_get_symbol_uri_str
2426     }
2427   }
2428   \par
2429   \exp_args:Nnnx
2430   \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
2431 }{
2432   \end{stex_annotate_env}
2433 }

```

STEXdefinition

```

2434
2435 \NewDocumentCommand \__stex_features_definiendum:w { 0{} m m } {
2436   \stex_get_symbol:n { ##2 }
2437   \scalatex_if:TF {
2438     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { ##3 }
2439   } {
2440     \exp_args:Nnx \@defemph { ##3 } { \l_stex_get_symbol_uri_str }
2441   }
2442 }
2443 \NewDocumentCommand \__stex_features_definame:w { 0{} m } {
2444   % TODO: root
2445   \stex_get_symbol:n { ##2 }

```

```

2446 \str_set:Nx \l_tmpa_str {
2447   \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
2448 }
2449 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
2450 \scalatex_if:TF {
2451   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
2452     \l_tmpa_str
2453   }
2454 } {
2455   \@defemph {
2456     \l_tmpa_str
2457   } { \l_stex_get_symbol_uri_str }
2458 }
2459 }
2460
2461 \cs_new_protected:Nn \__stex_features_defi_begin:n {
2462   \let\definiendum\__stex_features_definiendum:w
2463   \let\definame\__stex_features_definame:w
2464   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
2465   \seq_clear:N \l_tmpb_seq
2466   \seq_map_inline:Nn \l_tmpa_seq {
2467     \stex_get_symbol:n { ##1 }
2468     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
2469       \l_stex_get_symbol_uri_str
2470     }
2471   }
2472   \exp_args:Nnnx
2473   \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
2474 }
2475
2476 \cs_new_protected:Nn \__stex_features_defi_end: {
2477   \end{stex_annotate_env}
2478 }
2479
2480 \NewDocumentEnvironment{STEXdefinition}{ m }{
2481   \__stex_features_defi_begin:n { #1 }
2482 }{
2483   \__stex_features_defi_end:
2484 }

```

`\setSTEXdefinition`

```

2485 \cs_new_protected:Npn \setSTEXdefinition #1 {
2486   \AddToHook{env/#1/before}[stex]{\__stex_features_defi_begin:n{}}
2487   \AddToHook{env/#1/after}[stex]{\__stex_features_defi_end:}
2488 }

```

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

`structural@feature`

```

2489
2490 \NewDocumentEnvironment{structural@feature}{ m m m }{
2491   \stex_if_in_module:F {
2492     \msg_set:nnn{stex}{error/nomodule}{
2493       Structural~Feature~has~to~occur~in~a~module:\\

```



```

2494     Feature~#2~of~type~#1\\
2495     In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
2496   }
2497   \msg_error:nn{stex}{error/nomodule}
2498 }
2499
2500 \str_set:Nx \l_stex_module_name_str {
2501   \prop_item:Nn \l_stex_current_module_prop
2502     { name } / #2 - feature
2503 }
2504
2505
2506 \str_clear:N \l_tmpa_str
2507 \seq_clear:N \l_tmpa_seq
2508 \tl_clear:N \l_tmpa_tl
2509 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
2510   origname = #2,
2511   name      = \l_stex_module_name_str ,
2512   ns        = \l_stex_module_ns_str ,
2513   imports   = \exp_not:o { \l_tmpa_seq } ,
2514   constants = \exp_not:o { \l_tmpa_seq } ,
2515   content   = \exp_not:o { \l_tmpa_tl } ,
2516   file      = \exp_not:o { \g_stex_currentfile_seq } ,
2517   lang      = \l_stex_module_lang_str ,
2518   sig       = \l_tmpa_str ,
2519   meta      = \l_tmpa_str ,
2520   feature   = #1 ,
2521 }
2522
2523 \stex_if_smsmode:TF {
2524   \stex_smsmode_set_codes:
2525 } {
2526   \begin{stex_annotate_env}{ feature:#1 }{}
2527   \stex_annotate_invisible:nnn{header}{}{ #3 }
2528 }
2529 }{
2530   \str_set:Nx \l_tmpa_str {
2531     c_stex_feature_
2532     \prop_item:Nn \l_stex_current_module_prop { ns } ?
2533     \prop_item:Nn \l_stex_current_module_prop { name }
2534     _prop
2535   }
2536   \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
2537   \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
2538   \stex_if_smsmode:TF {
2539     \exp_args:Nx \stex_addtosms:n {
2540       \prop_gset_from_keyval:cn {
2541         c_stex_feature_
2542         \prop_item:Nn \l_stex_current_module_prop { ns } ?
2543         \prop_item:Nn \l_stex_current_module_prop { name }
2544         _prop
2545       } {
2546         origname = #2,
2547         name      = \prop_item:cn { \l_tmpa_str } { name } ,

```

```

2548         ns      = \prop_item:cn { \l_tmpa_str } { ns } ,
2549         imports  = \prop_item:cn { \l_tmpa_str } { imports } ,
2550         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
2551         content  = \prop_item:cn { \l_tmpa_str } { content } ,
2552         file     = \prop_item:cn { \l_tmpa_str } { file } ,
2553         lang     = \prop_item:cn { \l_tmpa_str } { lang } ,
2554         sig      = \prop_item:cn { \l_tmpa_str } { sig } ,
2555         meta     = \prop_item:cn { \l_tmpa_str } { meta } ,
2556         feature  = \prop_item:cn { \l_tmpa_str } { feature }
2557     }
2558 }
2559 } {
2560     \end{stex_annotate_env}
2561 }
2562 }
2563

```

structure

```

2564
2565 \prop_new:N \l_stex_all_structures_prop
2566
2567 \keys_define:nn { stex / features / structure } {
2568     name          .tl_set_x:N = \l__stex_features_structure_name_str ,
2569 }
2570
2571 \cs_new_protected:Nn \__stex_features_structure_args:n {
2572     \str_clear:N \l__stex_features_structure_name_str
2573     \keys_set:nn { stex / features / structure } { #1 }
2574     \exp_args:NNo \str_set:Nn \l__stex_features_structure_name_str
2575         \l__stex_features_structure_name_str
2576 }
2577
2578 %\stex_new_feature:nnnn { structure } { 0{ } m } {
2579 % \__stex_features_structure_args:n { ##1 }
2580 % \str_if_empty:NT \l__stex_features_structure_name_str {
2581 %     \str_set:Nx \l__stex_features_structure_name_str { ##2 }
2582 % }
2583 %} {
2584 %
2585 %}
2586
2587 \NewDocumentEnvironment{structure}{ 0{ } m }{
2588     \__stex_features_structure_args:n { #1 }
2589     \str_if_empty:NT \l__stex_features_structure_name_str {
2590         \str_set:Nx \l__stex_features_structure_name_str { #2 }
2591     }
2592     \exp_args:Nnnx
2593     \begin{structural@feature}{ structure }
2594         { \l__stex_features_structure_name_str }{}
2595         \seq_clear:N \l_tmpa_seq
2596         \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
2597     }{
2598     }{
2599         \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq

```

```

2600 \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
2601 \str_set:Nx \l_tmpa_str {
2602   \prop_item:Nn \l_stex_current_module_prop { ns } ?
2603   \prop_item:Nn \l_stex_current_module_prop { name }
2604 }
2605 \seq_map_inline:Nn \l_tmpa_seq {
2606   \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
2607 }
2608 \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
2609 \exp_args:Nnx
2610 \AddToHookNext { env / structure / after }{
2611   \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}}{
2612     \stex_term_math_oms:nnnn { \l_tmpa_str }{}{0}{}
2613   }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
2614 \STEXexport {
2615   \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2616     {\prop_item:Nn \l_stex_current_module_prop { origname }}
2617     {\l_tmpa_str}
2618   \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2619     {#2}{\l_tmpa_str}
2620   % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
2621   %   \prop_item:Nn \l_stex_current_module_prop { origname },
2622   %   \l_tmpa_str
2623   % }
2624   % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
2625   %   #2,\l_tmpa_str
2626   % }
2627   % \tl_set:cx { #2 } {
2628   %   \stex_invoke_structure:n { \l_tmpa_str }
2629   % }
2630 }
2631
2632 \end{structural@feature}
2633 % \g_stex_last_feature_prop
2634 }

```

\instantiate

```

2635 \seq_new:N \l__stex_features_structure_field_seq
2636 \str_new:N \l__stex_features_structure_field_str
2637 \str_new:N \l__stex_features_structure_def_tl
2638 \prop_new:N \l__stex_features_structure_prop
2639 \NewDocumentCommand \instantiate { m O{} m }{
2640   \stex_smsmode_set_codes:
2641   \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
2642   \prop_set_eq:Nc \l__stex_features_structure_prop {
2643     c_stex_feature_\l_tmpa_str _prop
2644   }
2645   \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
2646   \seq_map_inline:Nn \l__stex_features_structure_field_seq {
2647     \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
2648     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
2649       \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
2650       \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
2651       {!} \l_tmpa_tl

```

```

2652     \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
2653       \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
2654       \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
2655       \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
2656     }{
2657       \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
2658       \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
2659       \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
2660         \l_tmpa_tl
2661       \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
2662         \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
2663         \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
2664       }{
2665         \tl_clear:N \l_tmpb_tl
2666       }
2667     }
2668   }{
2669     \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
2670     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
2671       \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
2672       \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
2673       \tl_clear:N \l_tmpa_tl
2674     }{
2675       % TODO throw error
2676     }
2677   }
2678   % \l_tmpa_str: name
2679   % \l_tmpa_tl: definiens
2680   % \l_tmpb_tl: notation
2681   \tl_if_empty:NT \l__stex_features_structure_field_str {
2682     % TODO throw error
2683   }
2684   \str_clear:N \l_tmpb_str
2685
2686   \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
2687   \seq_map_inline:Nn \l_tmpa_seq {
2688     \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
2689     \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
2690     \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
2691       \seq_map_break:n {
2692         \str_set:Nn \l_tmpb_str { ####1 }
2693       }
2694     }
2695   }
2696   \prop_get:cnN { g_stex_symdecl_ \l_tmpb_str _prop } {args}
2697   \l_tmpb_str
2698
2699   \tl_if_empty:NNTF \l_tmpb_tl {
2700     \tl_if_empty:NF \l_tmpa_tl {
2701       \exp_args:Nx \use:n {
2702         \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fe
2703       }
2704     }
2705   }{

```

```

2706 \tl_if_empty:NTF \l_tmpa_tl {
2707   \exp_args:Nx \use:n {
2708     \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\
2709   }
2710
2711 }{
2712   \exp_args:Nx \use:n {
2713     \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
2714     \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
2715   }
2716 }
2717 }
2718 % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
2719 % \prop_item:Nn \l_stex_current_module_prop {name} ?
2720 % #3/\l__stex_features_structure_field_str
2721 % \par
2722 % \expandafter\present\csname
2723 %   g_stex_symdecl_
2724 %   \prop_item:Nn \l_stex_current_module_prop {ns} ?
2725 %   \prop_item:Nn \l_stex_current_module_prop {name} ?
2726 %   #3/\l__stex_features_structure_field_str
2727 %   _prop
2728 % \endcsname
2729 }
2730
2731 \tl_clear:N \l__stex_features_structure_def_tl
2732
2733 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
2734 \seq_map_inline:Nn \l_tmpa_seq {
2735   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
2736   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
2737   \exp_args:Nx \use:n {
2738     \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
2739
2740     }
2741   }
2742
2743   \prop_if_exist:cF {
2744     g_stex_symdecl_
2745     \prop_item:Nn \l_stex_current_module_prop {ns} ?
2746     \prop_item:Nn \l_stex_current_module_prop {name} ?
2747     #3/\l_tmpa_str
2748     _prop
2749   }{
2750     \prop_get:cnN { g_stex_symdecl_ ##1 _prop } {args}
2751     \l_tmpb_str
2752     \exp_args:Nx \use:n {
2753       \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
2754     }
2755   }
2756 }
2757
2758 \symdecl*[type={\STEXsymbol{module-type}}{
2759   \_stex_term_math_oms:nnnn {

```

```

2760     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
2761     \prop_item:Nn \l__stex_features_structure_prop {name}
2762     }{}{0}{}
2763   }}{#3}
2764
2765   % TODO: -> sms file
2766
2767   \tl_set:cx{ #3 }{
2768     \stex_invoke_structure:nnn {
2769       \prop_item:Nn \l_stex_current_module_prop {ns} ?
2770       \prop_item:Nn \l_stex_current_module_prop {name} ? #3
2771     } {
2772       \prop_item:Nn \l__stex_features_structure_prop {ns} ?
2773       \prop_item:Nn \l__stex_features_structure_prop {name}
2774     }
2775   }
2776
2777 }

```

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

\stex_invoke_structure:nnn

```

2778   % #1: URI of the instance
2779   % #2: URI of the instantiated module
2780   \cs_new_protected:Nn \stex_invoke_structure:nnn {
2781     \tl_if_empty:nTF{ #3 }{
2782       \prop_set_eq:Nc \l__stex_features_structure_prop {
2783         c_stex_feature_ #2 _prop
2784       }
2785       \tl_clear:N \l_tmpa_tl
2786       \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
2787       \seq_map_inline:Nn \l_tmpa_seq {
2788         \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
2789         \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
2790         \cs_if_exist:cT {
2791           stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
2792         }{
2793           \tl_if_empty:NF \l_tmpa_tl {
2794             \tl_put_right:Nn \l_tmpa_tl {,}
2795           }
2796           \tl_put_right:Nx \l_tmpa_tl {
2797             \stex_invoke_symbol:n {#1/\l_tmpa_str}!
2798           }
2799         }
2800       }
2801       \scalatexBREAK
2802       \exp_args:No \mathstruct \l_tmpa_tl
2803     }{
2804       \stex_invoke_symbol:n{#1/#3}
2805     }
2806   }

```

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

4.11 Put these somewhere

\MSC

```
2807 \NewDocumentCommand \MSC {m} {
2808   % TODO
2809 }
```

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

```
2810 \@ifpackageloaded{tikzinput}{
2811   \RequirePackage{stex-tikzinput}
2812 }{}
2813
2814 \AddToHook{begindocument}{
2815   \input{stex-metattheory}
2816 }
2817 \</package>
```

4.12 Metatheory

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

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

```
2818 \<*metatheory>
2819 \ExplSyntaxOn
2820 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
2821 \begin{@module}[ns=\c_stex_metatheory_ns_str,meta=NONE]{Metatheory}
2822   \ExplSyntaxOff
2823
2824   % is-a (a:A, a \in A, a is an A, etc.)
2825   \symdecl[args=ai]{isa}
2826   \notation[typed]{isa}{#1 \comp: #2}{#1 \comp, #2}
2827   \notation[in]{isa}{#1 \comp\in #2}{#1 \comp, #2}
2828   \notation[pred]{isa}{#2\comp(#1 \comp)}{#1 \comp, #2}
2829
2830   % bind (\forall, \Pi, \lambda etc.)
2831   \symdecl[args=Bi]{bind}
2832   \notation[forall]{bind}{\comp\forall #1.\;#2}{#1 \comp, #2}
2833   \notation[\Pi]{bind}{\comp\prod_{#1}#2}{#1 \comp, #2}
2834   \notation[depfun]{bind}{\comp( #1 \comp{ }\;\to\; )}{#1 \comp, #2}
2835
2836   % dummy variable
2837   \symdecl{dummyvar}
2838   \notation[underscore]{dummyvar}{\comp\_}
2839   \notation[dot]{dummyvar}{\comp\cdot}
2840   \notation[dot]{dummyvar}{\comp\cdot}
2841   \notation[dash]{dummyvar}{\comp{\rm --}}
2842
```

```

2843 %fromto (function space, Hom-set, implication etc.)
2844 \symdecl[args=ai]{fromto}
2845 \notation[xarrow]{fromto}{#1 \comp\to #2}{#1 \comp\times #2}
2846 \notation[arrow]{fromto}{#1 \comp\to #2}{#1 \comp\to #2}
2847
2848 % mapto (lambda etc.)
2849 \symdecl[args=Bi]{mapto}
2850 %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
2851 %\notation[lambda]{mapto}{\comp\lambda #1 \comp.; #2}{#1 \comp, #2}
2852 %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp.; #2}{#1 \comp, #2}
2853
2854 % function/operator application
2855 \symdecl[args=ia]{apply}
2856 \notation[prec=0;0x\neginfprec,parens]{apply}{#1 \comp( #2 \comp)}{#1 \comp, #2}
2857 \notation[prec=0;0x\neginfprec,lambda]{apply}{#1 \; #2 }{#1 \; #2}
2858
2859 % ‘‘type’’ of all collections (sets, classes, types, kinds)
2860 \symdecl{collection}
2861 \notation[U]{collection}{\comp{\mathcal{U}}}
2862 \notation[set]{collection}{\comp{\textsf{Set}}}
2863
2864 % sequences
2865 \symdecl[args=1]{seqtype}
2866 \notation[kleene]{seqtype}{#1^{\comp\ast}}
2867
2868 \symdef[args=2,li]{sequence-index}{#1_{#2}}
2869 \notation[ui]{sequence-index}{#1^{#2}}
2870
2871 %\symdef[args=3,li]{sequence-from-to}{#1_{#2}\comp{\,\ellipses,}#1_{#3}}
2872 %\notation[ui]{sequence-from-to}{#1^{#2}\comp{\,\ellipses,}#1^{#3}}
2873 % ^ superceded by \aseqfromto and \livar/\uivar
2874
2875 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{\,\ellipses}}{#1\comp,#2}
2876 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{\,\ellipses\comp,}#2 }{#1\comp,#2}
2877
2878 % letin (‘‘let’’, local definitions, variable substitution)
2879 \symdecl[args=bii]{letin}
2880 \notation[let]{letin}{\comp{\rm let}}{#1\comp{=}#2\; \comp{\rm in}}{#3}
2881 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
2882 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
2883
2884 % structures
2885 \symdecl*[args=1]{module-type}
2886 \notation{module-type}{\mathtt{MOD} #1}
2887 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
2888 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{#1 \comp, #2}
2889
2890 \STEXexport{
2891   \let\nappa\apply
2892   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
2893   \def\livar{\csname sequence-index\endcsname[li]}
2894   \def\uivar{\csname sequence-index\endcsname[ui]}
2895   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
2896   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}

```



```

2897 }
2898
2899 \end{@module}
2900 \ExplSyntaxOff
2901 </metatheory>

```

4.13 Auxiliary Packages

4.13.1 tikzinput

```

2902 <*tikzinput>
2903 <@@=tikzinput>
2904 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
2905 \RequirePackage{l3keys2e}
2906
2907 \keys_define:nn { tikzinput } {
2908   image .bool_set:N = \c_tikzinput_image_bool
2909 }
2910
2911 \ProcessKeysOptions { tikzinput }
2912
2913 \bool_if:NTF \c_tikzinput_image_bool {
2914   \RequirePackage{graphicx}
2915
2916   \providecommand\usetikzlibrary[]{}
2917   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
2918 }{
2919   \RequirePackage{tikz}
2920   \RequirePackage{standalone}
2921
2922   \newcommand \tikzinput [2] [] {
2923     \setkeys{Gin}{#1}
2924     \ifx \Gin@width \Gin@exclamation
2925       \ifx \Gin@height \Gin@exclamation
2926         \input { #2 }
2927       \else
2928         \resizebox{!}{ \Gin@height }{
2929           \input { #2 }
2930         }
2931       \fi
2932     \else
2933       \ifx \Gin@height \Gin@exclamation
2934         \resizebox{ \Gin@width }{!}{
2935           \input { #2 }
2936         }
2937       \else
2938         \resizebox{ \Gin@width }{ \Gin@height }{
2939           \input { #2 }
2940         }
2941       \fi
2942     \fi
2943   }
2944 }
2945

```

```

2946 \newcommand \ctikzinput [2] [] {
2947   \begin{center}
2948     \tikzinput [#1] {#2}
2949   \end{center}
2950 }
2951
2952 \@ifpackageloaded{stex}{
2953   \RequirePackage{stex-tikzinput}
2954 }{}
2955 \</tikzinput>
2956
2957 \< *stex-tikzinput>
2958 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
2959 \RequirePackage{stex}
2960 \RequirePackage{tikzinput}
2961
2962 % TODO
2963 \< /stex-tikzinput>

```

4.13.2 sTeX1 Compatibility

```

2964 \< *smglom>
2965 \RequirePackage{expl3,l3keys2e}
2966 \ProvidesExplClass{smglom}{2021/08/01}{1.9}{sTeX1 compatibility}
2967 \LoadClass[border=1px,varwidth]{standalone}
2968 \setlength\textwidth{15cm}
2969 %\g@addto@macro{\@parboxrestore}{\setlength\parskip{\baselineskip}}
2970 \DeclareOption{mh}{\}
2971 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
2972 \ProcessOptions
2973
2974 \RequirePackage{stex-compatibility}
2975 \< /smglom>
2976
2977 \< *compat>
2978 \< @@=stex_deprec>
2979 \ProvidesExplPackage{stex-compatibility}{2021/08/01}{1.9}{bla}
2980 \RequirePackage[lang={de,en,ro,tr,fr}]{stex}
2981
2982 \NewDocumentEnvironment { mhmodnl } { 0 } { m m } {
2983   \msg_set:nnn{stex}{warning/deprecated}{
2984     \
2985     Environment~mhmodnl~is~deprected! \
2986     Please~update~module~#2~in~file~
2987     \stex_path_to_string:N \g_stex_currentfile_seq!
2988     \
2989   }
2990   \msg_warning:nn{stex}{warning/deprecated}
2991
2992   \begin{module} [#1,lang=#3] {#2}
2993     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
2994     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
2995     \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
2996     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
2997     \input { \stex_path_to_string:N \l_tmpa_seq / \l_tmpa_str.tex }

```

```

2998 } {
2999   \end{module}
3000 }
3001
3002 \NewDocumentEnvironment { modsig } { 0{} m } {
3003   \stex_if_in_module:TF {
3004     \prop_get:NnN \l_stex_current_module_prop {name} \l_tmpa_str
3005     \str_set:Nn \l_tmpb_str { #2 }
3006     \str_if_eq:NNTF \l_tmpa_str \l_tmpb_str {
3007       \prop_set_eq:NN \l_modsig_old_module_prop \l_stex_current_module_prop
3008       \begin{@module}{modsig-#2}
3009       % \prop_set_eq:NN \l_stex_current_module_prop \l_modsig_old_module_prop
3010     } {
3011       \begin{@module}{#2}
3012     }
3013   } {
3014     \begin{@module}{#2}
3015   }
3016 }{
3017   \end{@module}
3018   \AddToHookNext { env / modsig / after }{
3019     \stex_if_in_module:T {
3020       \prop_get:NnN \l_stex_current_module_prop {name} \l_tmpa_str
3021       \str_set:Nn \l_tmpb_str { #2 }
3022       \str_if_eq:NNT \l_tmpa_str \l_tmpb_str {
3023         % \xdef \g_stex_module_after_group_tl {
3024           \stex_if_smsmode:TF {
3025             \exp_args:Nx
3026             \stex_add_to_current_module:n {
3027               \stex_debug:n{Activating-signature-of-#2}
3028               \exp_not:N \prop_item:cn { c_stex_module_
3029               \prop_item:Nn \l_stex_current_module_prop {ns} ?
3030               \prop_item:Nn \l_stex_current_module_prop {name}
3031               / modsig-#2_prop } { content }
3032             }
3033           }
3034         {
3035           \gdef \g_stex_modsig_after_group_tl {
3036             \stex_activate_module:n {
3037               \prop_item:Nn \l_stex_current_module_prop {ns} ?
3038               \prop_item:Nn \l_stex_current_module_prop {name}
3039               / modsig-#2
3040             }
3041           }
3042           \exp_args:Nx
3043           \stex_add_to_current_module:n {
3044             \stex_activate_module:n {
3045               \prop_item:Nn \l_stex_current_module_prop {ns} ?
3046               \prop_item:Nn \l_stex_current_module_prop {name}
3047               / modsig-#2
3048             }
3049           }
3050         }
3051       \aftergroup \g_stex_modsig_after_group_tl

```

```

3052     }
3053   }
3054 }
3055 }
3056 }
3057
3058 \cs_new_protected:Npn \gimport {
3059   \peek_charcode_remove:NTF * {
3060     \gimport_do:
3061   } {
3062     \gimport_do:
3063   }
3064 }
3065
3066 \NewDocumentCommand \gimport_do: { O{} m } {
3067   \msg_set:nnn{stex}{warning/deprecated}{
3068     \\\
3069     \c_backslash_str gimport~is~deprecated! \\\
3070     Please~use~\c_backslash_str importmodule[#1]{#2}~instead!~(in~file~
3071     \stex_path_to_string:N \g_stex_currentfile_seq)
3072     \\\ \\\
3073   }
3074   \msg_warning:nn{stex}{warning/deprecated}
3075   \importmodule[#1]{#2}
3076 }
3077
3078 \cs_new_protected:Npn \guse {
3079   \peek_charcode_remove:NTF * {
3080     \guse_do:
3081   } {
3082     \guse_do:
3083   }
3084 }
3085
3086 \NewDocumentCommand \guse_do: { O{} m } {
3087   \msg_set:nnn{stex}{warning/deprecated}{
3088     \\\
3089     \c_backslash_str guse~is~deprecated! \\\
3090     Please~use~\c_backslash_str usemodule[#1]{#2}~instead!~(in~file~
3091     \stex_path_to_string:N \g_stex_currentfile_seq)
3092     \\\ \\\
3093   }
3094   \msg_warning:nn{stex}{warning/deprecated}
3095   \usemodule[#1]{#2}
3096 }
3097
3098 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3099
3100 \cs_new_protected:Npn \sympi {
3101   \peek_charcode_remove:NTF * {
3102     \sympi_do:
3103   } {
3104     \sympi_do:
3105   }

```

```

3106 }
3107
3108 \NewDocumentCommand \symi_do: { 0{} m } {
3109   \msg_set:nnn{stex}{warning/deprecated}{
3110     \\
3111     \c_backslash_str symi-is-deprecated! \\
3112     Please-use~\c_backslash_str symdecl[#1]{#2}~instead!~(in~file~
3113     \stex_path_to_string:N \g_stex_currentfile_seq)
3114     \\ \\
3115   }
3116   \msg_warning:nn{stex}{warning/deprecated}
3117   \symdecl*{#1}{#2}
3118 }
3119
3120 \cs_new_protected:Npn \symii {
3121   \peek_charcode_remove:NTF * {
3122     \symii_do:
3123   } {
3124     \symii_do:
3125   }
3126 }
3127
3128 \NewDocumentCommand \symii_do: { 0{} m m } {
3129   \msg_set:nnn{stex}{warning/deprecated}{
3130     \\
3131     \c_backslash_str symii-is-deprecated! \\
3132     Please-use~\c_backslash_str symdecl[#1]{#2-#3}~instead!~(in~file~
3133     \stex_path_to_string:N \g_stex_currentfile_seq)
3134     \\ \\
3135   }
3136   \msg_warning:nn{stex}{warning/deprecated}
3137   \symdecl*{#1}{#2-#3}
3138 }
3139
3140 \cs_new_protected:Npn \symiii {
3141   \peek_charcode_remove:NTF * {
3142     \symiii_do:
3143   } {
3144     \symiii_do:
3145   }
3146 }
3147
3148 \NewDocumentCommand \symiii_do: { 0{} m m m } {
3149   \msg_set:nnn{stex}{warning/deprecated}{
3150     \\
3151     \c_backslash_str symiii-is-deprecated! \\
3152     Please-use~\c_backslash_str symdecl[#1]{#2-#3-#4}~instead!~(in~file~
3153     \stex_path_to_string:N \g_stex_currentfile_seq)
3154     \\ \\
3155   }
3156   \msg_warning:nn{stex}{warning/deprecated}
3157   \symdecl*{#1}{#2-#3-#4}
3158 }
3159

```

```

3160 \keys_define:nn { stex / deprec / defi } {
3161   name .tl_set_x:N = \l_tmpa_str
3162 }
3163
3164 \cs_new_protected:Npn \defi {
3165   \peek_charcode_remove:NTF * {
3166     \defi_do:
3167   } {
3168     \defi_do:
3169   }
3170 }
3171
3172 \NewDocumentCommand \defi_do: { 0{ } m } {
3173   \str_clear:N \l_tmpa_str
3174   \keys_set:nn { stex / deprec / defi } { #1 }
3175
3176   \str_if_empty:NTF \l_tmpa_str {
3177     \msg_set:nnn{stex}{warning/deprecated}{
3178       \\\
3179       \c_backslash_str defi-is~deprecated! \\\
3180       Please~use~\c_backslash_str STEXsymbol{#2}![#2]~instead!~(in~file~
3181       \stex_path_to_string:N \g_stex_currentfile_seq)
3182       \\\ \\\
3183     }
3184     \msg_warning:nn{stex}{warning/deprecated}
3185     \STEXsymbol { #2 }![ \comp{#2} ]
3186   } {
3187     \msg_set:nnn{stex}{warning/deprecated}{
3188       \\\
3189       \c_backslash_str defi-is~deprecated! \\\
3190       Please~use~\c_backslash_str STEXsymbol { \l_tmpa_str }[ #2 ]~instead!~(in~file~
3191       \stex_path_to_string:N \g_stex_currentfile_seq)
3192       \\\ \\\
3193     }
3194     \msg_warning:nn{stex}{warning/deprecated}
3195     \exp_args:No \STEXsymbol { \l_tmpa_str }![ \comp{#2} ]
3196   }
3197 }
3198
3199
3200 \cs_new_protected:Npn \Defi {
3201   \peek_charcode_remove:NTF * {
3202     \Defi_do:
3203   } {
3204     \Defi_do:
3205   }
3206 }
3207
3208 \NewDocumentCommand \Defi_do: { 0{ } m } {
3209   \str_clear:N \l_tmpa_str
3210   \keys_set:nn { stex / deprec / defi } { #1 }
3211
3212   \str_if_empty:NTF \l_tmpa_str {
3213     \msg_set:nnn{stex}{warning/deprecated}{

```

```

3214     \\\
3215     \c_backslash_str Defi-is~deprecated! \\\
3216     Please~use~\c_backslash_str STEXsymbol{#2}![\exp_after:wN \stex_capitalize:n #2]~inste
3217     \stex_path_to_string:N \g_stex_currentfile_seq)
3218     \\\ \\\
3219   }
3220   \msg_warning:nn{stex}{warning/deprecated}
3221   \STEXsymbol { #2 }![ \comp{\exp_after:wN \stex_capitalize:n #2} ]
3222 } {
3223   \msg_set:nnn{stex}{warning/deprecated}{
3224     \\\
3225     \c_backslash_str Defi-is~deprecated! \\\
3226     Please~use~\c_backslash_str STEXsymbol { \l_tmpa_str }[ \exp_after:wN \stex_capitalize
3227     \stex_path_to_string:N \g_stex_currentfile_seq)
3228     \\\ \\\
3229   }
3230   \msg_warning:nn{stex}{warning/deprecated}
3231   \exp_args:No \STEXsymbol { \l_tmpa_str }![ \comp{\exp_after:wN \stex_capitalize:n #2} ]
3232 }
3233 }
3234
3235 \cs_new_protected:Npn \adefi {
3236   \peek_charcode_remove:NTF * {
3237     \adefi_do:
3238   } {
3239     \adefi_do:
3240   }
3241 }
3242
3243 \NewDocumentCommand \adefi_do: { 0{} m m } {
3244   \str_clear:N \l_tmpa_str
3245   \keys_set:nn { stex / deprec / def } { #1 }
3246
3247   \str_if_empty:NTF \l_tmpa_str {
3248     \msg_set:nnn{stex}{warning/deprecated}{
3249       \\\
3250       \c_backslash_str adefi-is~deprecated! \\\
3251       Please~use~\c_backslash_str STEXsymbol{#3}![#2]~instead!~(in~file~
3252       \stex_path_to_string:N \g_stex_currentfile_seq)
3253       \\\ \\\
3254     }
3255     \msg_warning:nn{stex}{warning/deprecated}
3256     \STEXsymbol { #3 }![ \comp{#2} ]
3257   } {
3258     \msg_set:nnn{stex}{warning/deprecated}{
3259       \\\
3260       \c_backslash_str adefi-is~deprecated! \\\
3261       Please~use~\c_backslash_str STEXsymbol { \l_tmpa_str }[ #2 ]~instead!~(in~file~
3262       \stex_path_to_string:N \g_stex_currentfile_seq)
3263       \\\ \\\
3264     }
3265     \msg_warning:nn{stex}{warning/deprecated}
3266     \exp_args:No \STEXsymbol { \l_tmpa_str }![ \comp{#2} ]
3267   }

```

```

3268 }
3269
3270 \cs_new_protected:Npn \defis {
3271   \peek_charcode_remove:NTF * {
3272     \defis_do:
3273   } {
3274     \defis_do:
3275   }
3276 }
3277
3278 \NewDocumentCommand \defis_do: { 0{ } m } {
3279   \str_clear:N \l_tmpa_str
3280   \keys_set:nn { stex / deprec / defi } { #1 }
3281
3282   \str_if_empty:NTF \l_tmpa_str {
3283     \msg_set:nnn{stex}{warning/deprecated}{
3284       \\\
3285       \c_backslash_str defis-is-deprecated! \\\
3286       Please~use~\c_backslash_str STEXsymbol{#2}![#2s]~instead!~(in~file~
3287       \stex_path_to_string:N \g_stex_currentfile_seq)
3288       \\\ \\\
3289     }
3290     \msg_warning:nn{stex}{warning/deprecated}
3291     \STEXsymbol { #2 }![ \comp{#2s} ]
3292   } {
3293     \msg_set:nnn{stex}{warning/deprecated}{
3294       \\\
3295       \c_backslash_str defis-is-deprecated! \\\
3296       Please~use~\c_backslash_str STEXsymbol { \l_tmpa_str }[ #2s ]~instead!~(in~file~
3297       \stex_path_to_string:N \g_stex_currentfile_seq)
3298       \\\ \\\
3299     }
3300     \msg_warning:nn{stex}{warning/deprecated}
3301     \exp_args:No \STEXsymbol { \l_tmpa_str }![ \comp{#2s} ]
3302   }
3303 }
3304
3305 \cs_new_protected:Npn \defii {
3306   \peek_charcode_remove:NTF * {
3307     \defii_do:
3308   } {
3309     \defii_do:
3310   }
3311 }
3312
3313 \NewDocumentCommand \defii_do: { 0{ } m m } {
3314   \str_clear:N \l_tmpa_str
3315   \keys_set:nn { stex / deprec / defi } { #1 }
3316   \str_if_empty:NTF \l_tmpa_str {
3317     \msg_set:nnn{stex}{warning/deprecated}{
3318       \\\
3319       \c_backslash_str defii-is-deprecated! \\\
3320       Please~use~\c_backslash_str STEXsymbol{#2~#3}![#2~#3]~instead!~(in~file~
3321       \stex_path_to_string:N \g_stex_currentfile_seq)

```



```

3322     \\\ \\\
3323   }
3324   \msg_warning:nn{stex}{warning/deprecated}
3325   \STEXsymbol { #2-#3 }![ \comp{#2~#3} ]
3326 } {
3327   \msg_set:nnn{stex}{warning/deprecated}{
3328     \\\
3329     \c_backslash_str defii-is-deprecated! \\\
3330     Please~use~\c_backslash_str STEXsymbol { \l_tmpa_str }[ #2~#3 ]~instead!~(in~file~
3331     \stex_path_to_string:N \g_stex_currentfile_seq)
3332     \\\ \\\
3333   }
3334   \msg_warning:nn{stex}{warning/deprecated}
3335   \exp_args:No \STEXsymbol { \l_tmpa_str }![ \comp{#2~#3} ]
3336 }
3337 }
3338
3339
3340 \cs_new_protected:Npn \defiis {
3341   \peek_charcode_remove:NTF * {
3342     \defiis_do:
3343   } {
3344     \defiis_do:
3345   }
3346 }
3347
3348 \NewDocumentCommand \defiis_do: { 0{} m m } {
3349   \str_clear:N \l_tmpa_str
3350   \keys_set:nn { stex / deprec / defi } { #1 }
3351   \str_if_empty:NTF \l_tmpa_str {
3352     \msg_set:nnn{stex}{warning/deprecated}{
3353       \\\
3354       \c_backslash_str defiis-is-deprecated! \\\
3355       Please~use~\c_backslash_str STEXsymbol{#2-#3}![#2~#3s]~instead!~(in~file~
3356       \stex_path_to_string:N \g_stex_currentfile_seq)
3357       \\\ \\\
3358     }
3359     \msg_warning:nn{stex}{warning/deprecated}
3360     \STEXsymbol { #2-#3 }![ \comp{#2~#3s} ]
3361   } {
3362     \msg_set:nnn{stex}{warning/deprecated}{
3363       \\\
3364       \c_backslash_str defiis-is-deprecated! \\\
3365       Please~use~\c_backslash_str STEXsymbol { \l_tmpa_str }[ #2~#3s ]~instead!~(in~file~
3366       \stex_path_to_string:N \g_stex_currentfile_seq)
3367       \\\ \\\
3368     }
3369     \msg_warning:nn{stex}{warning/deprecated}
3370     \exp_args:No \STEXsymbol { \l_tmpa_str }![ \comp{#2~#3s} ]
3371   }
3372 }
3373
3374
3375 \cs_new_protected:Npn \defiii {

```

```

3376 \peek_charcode_remove:NTF * {
3377   \defiii_do:
3378 } {
3379   \defiii_do:
3380 }
3381 }
3382
3383 \NewDocumentCommand \defiii_do: { O{} m m m } {
3384   \str_clear:N \l_tmpa_str
3385   \keys_set:nn { stex / deprec / defi } { #1 }
3386   \str_if_empty:NTF \l_tmpa_str {
3387     \msg_set:nnn{stex}{warning/deprecated}{
3388       \\\
3389       \c_backslash_str defiii~is~deprecated! \\\
3390       Please~use~\c_backslash_str STExsymbol{#2~#3~#4}![#2~#3~#4]~instead!~(in~file~
3391       \stex_path_to_string:N \g_stex_currentfile_seq)
3392       \\\ \\\
3393     }
3394     \msg_warning:nn{stex}{warning/deprecated}
3395     \STExsymbol { #2~#3~#4 }![ \comp{#2~#3~#4} ]
3396   } {
3397     \msg_set:nnn{stex}{warning/deprecated}{
3398       \\\
3399       \c_backslash_str defiii~is~deprecated! \\\
3400       Please~use~\c_backslash_str STExsymbol { \l_tmpa_str }[ #2~#3~#4 ]~instead!~(in~file~
3401       \stex_path_to_string:N \g_stex_currentfile_seq)
3402       \\\ \\\
3403     }
3404     \msg_warning:nn{stex}{warning/deprecated}
3405     \exp_args:No \STExsymbol { \l_tmpa_str }![ \comp{#2~#3~#4} ]
3406   }
3407 }
3408
3409 %\RequirePackage[hyperref]{ntheorem}
3410 %\theoremstyle{plain}
3411 %\RequirePackage{amsthm}
3412
3413 \NewDocumentEnvironment {definition} { O{} } {
3414   \begin{STExdefinition}{}
3415 }{
3416   \end{STExdefinition}
3417 }
3418
3419 \NewDocumentCommand \inlinedef { m } {
3420   \begingroup
3421   \let\definiendum\_stex_deprec_definiendum:w
3422   \let\definame\_stex_deprec_definame:w
3423   #1
3424   \endgroup
3425 }
3426
3427 \NewDocumentCommand \inlineass { m } { #1 }
3428
3429 \NewDocumentCommand \trefi { O{} m } {

```

```

3430 \str_set:Nn \l_tmpa_str { #1 }
3431 \str_if_empty:NTF \l_tmpa_str {
3432   \msg_set:nnn{stex}{warning/deprecated}{
3433     \\\
3434     \c_backslash_str trefi~is~deprecated! \\\
3435     Please~use~\c_backslash_str STEXsymbol{#2}![#2]~instead!~(in~file~
3436     \stex_path_to_string:N \g_stex_currentfile_seq)
3437     \\\ \\\
3438   }
3439   \msg_warning:nn{stex}{warning/deprecated}
3440   \STEXsymbol { #2 }![ \comp{#2} ]
3441 } {
3442   \msg_set:nnn{stex}{warning/deprecated}{
3443     \\\
3444     \c_backslash_str trefi~is~deprecated! \\\
3445     Please~use~\c_backslash_str STEXsymbol { #1?#2 }[ #2 ]~instead!~(in~file~
3446     \stex_path_to_string:N \g_stex_currentfile_seq)
3447     \\\ \\\
3448   }
3449   \msg_warning:nn{stex}{warning/deprecated}
3450   \STEXsymbol { #1 }![ \comp{#2} ]
3451 }
3452 }
3453
3454
3455 \NewDocumentCommand \Trefi { 0{} m } {
3456   \str_set:Nn \l_tmpa_str { #1 }
3457   \str_if_empty:NTF \l_tmpa_str {
3458     \msg_set:nnn{stex}{warning/deprecated}{
3459       \\\
3460       \c_backslash_str Trefi~is~deprecated! \\\
3461       Please~use~\c_backslash_str STEXsymbol{#2}![\exp_after:wN \stex_capitalize:n #2]~inste
3462       \stex_path_to_string:N \g_stex_currentfile_seq)
3463       \\\ \\\
3464     }
3465     \msg_warning:nn{stex}{warning/deprecated}
3466     \STEXsymbol { #2 }![ \comp{\exp_after:wN \stex_capitalize:n #2} ]
3467   } {
3468     \msg_set:nnn{stex}{warning/deprecated}{
3469       \\\
3470       \c_backslash_str Trefi~is~deprecated! \\\
3471       Please~use~\c_backslash_str STEXsymbol { #1 }[ \exp_after:wN \stex_capitalize:n #2 ]~i
3472       \stex_path_to_string:N \g_stex_currentfile_seq)
3473       \\\ \\\
3474     }
3475     \msg_warning:nn{stex}{warning/deprecated}
3476     \STEXsymbol { #1 }![ \comp{\exp_after:wN \stex_capitalize:n #2} ]
3477   }
3478 }
3479
3480 \NewDocumentCommand \trefis { 0{} m } {
3481   \str_set:Nn \l_tmpa_str { #1 }
3482   \str_if_empty:NTF \l_tmpa_str {
3483     \msg_set:nnn{stex}{warning/deprecated}{

```

```

3484     \\\
3485     \c_backslash_str trefi-is-deprecated! \\\
3486     Please~use~\c_backslash_str STEXsymbol{#2}![#2s]~instead!~(in~file~
3487     \stex_path_to_string:N \g_stex_currentfile_seq)
3488     \\\ \\\
3489   }
3490   \msg_warning:nn{stex}{warning/deprecated}
3491   \STEXsymbol { #2 }![ \comp{#2s} ]
3492 } {
3493   \msg_set:nnn{stex}{warning/deprecated}{
3494     \\\
3495     \c_backslash_str trefi-is-deprecated! \\\
3496     Please~use~\c_backslash_str STEXsymbol { #1 }[ #2s ]~instead!~(in~file~
3497     \stex_path_to_string:N \g_stex_currentfile_seq)
3498     \\\ \\\
3499   }
3500   \msg_warning:nn{stex}{warning/deprecated}
3501   \STEXsymbol { #1 }![ \comp{#2s} ]
3502 }
3503 }
3504
3505
3506 \NewDocumentCommand \Trefis { 0{} m } {
3507   \str_set:Nn \l_tmpa_str { #1 }
3508   \str_if_empty:NTF \l_tmpa_str {
3509     \msg_set:nnn{stex}{warning/deprecated}{
3510       \\\
3511       \c_backslash_str Trefis-is-deprecated! \\\
3512       Please~use~\c_backslash_str STEXsymbol{#2}![\exp_after:wN \stex_capitalize:n #2s]~inst
3513       \stex_path_to_string:N \g_stex_currentfile_seq)
3514       \\\ \\\
3515     }
3516     \msg_warning:nn{stex}{warning/deprecated}
3517     \STEXsymbol { #2 }![ \comp{\exp_after:wN \stex_capitalize:n #2s} ]
3518   } {
3519     \msg_set:nnn{stex}{warning/deprecated}{
3520       \\\
3521       \c_backslash_str Trefis-is-deprecated! \\\
3522       Please~use~\c_backslash_str STEXsymbol { #1 }[ \exp_after:wN \stex_capitalize:n #2s ]~
3523       \stex_path_to_string:N \g_stex_currentfile_seq)
3524       \\\ \\\
3525     }
3526     \msg_warning:nn{stex}{warning/deprecated}
3527     \STEXsymbol { #1 }![ \comp{\exp_after:wN \stex_capitalize:n #2s} ]
3528   }
3529 }
3530
3531 \NewDocumentCommand \trefii { 0{} m m } {
3532   \str_set:Nn \l_tmpa_str { #1 }
3533   \str_if_empty:NTF \l_tmpa_str {
3534     \msg_set:nnn{stex}{warning/deprecated}{
3535       \\\
3536       \c_backslash_str trefii-is-deprecated! \\\
3537       Please~use~\c_backslash_str STEXsymbol{#2-#3}![#2~#3]~instead!~(in~file~

```

```

3538     \stex_path_to_string:N \g_stex_currentfile_seq)
3539     \\\ \\\
3540   }
3541   \msg_warning:nn{stex}{warning/deprecated}
3542   \STEXsymbol { #2-#3 }![ \comp{#2-#3} ]
3543 } {
3544   \msg_set:nnn{stex}{warning/deprecated}{
3545     \\\
3546     \c_backslash_str trefiii~is-deprecated! \\\
3547     Please~use~\c_backslash_str STEXsymbol { #1 }[ #2-#3 ]~instead!~(in~file~
3548     \stex_path_to_string:N \g_stex_currentfile_seq)
3549     \\\ \\\
3550   }
3551   \msg_warning:nn{stex}{warning/deprecated}
3552   \STEXsymbol { #1 }![ \comp{#2-#3} ]
3553 }
3554 }
3555
3556 \NewDocumentCommand \trefiii { 0{} m m m } {
3557   \str_set:Nn \l_tmpa_str { #1 }
3558   \str_if_empty:NTF \l_tmpa_str {
3559     \msg_set:nnn{stex}{warning/deprecated}{
3560       \\\
3561       \c_backslash_str trefiii~is-deprecated! \\\
3562       Please~use~\c_backslash_str STEXsymbol{#2-#3-#4}![#2-#3-#4]~instead!~(in~file~
3563       \stex_path_to_string:N \g_stex_currentfile_seq)
3564       \\\ \\\
3565     }
3566     \msg_warning:nn{stex}{warning/deprecated}
3567     \STEXsymbol { #2-#3-#4 }![ \comp{#2-#3-#4} ]
3568   } {
3569     \msg_set:nnn{stex}{warning/deprecated}{
3570       \\\
3571       \c_backslash_str trefiii~is-deprecated! \\\
3572       Please~use~\c_backslash_str STEXsymbol { #1 }[ #2-#3-#4 ]~instead!~(in~file~
3573       \stex_path_to_string:N \g_stex_currentfile_seq)
3574       \\\ \\\
3575     }
3576     \msg_warning:nn{stex}{warning/deprecated}
3577     \STEXsymbol { #1 }![ \comp{#2-#3-#4} ]
3578   }
3579 }
3580
3581
3582 \NewDocumentCommand \trefiis { 0{} m m } {
3583   \str_set:Nn \l_tmpa_str { #1 }
3584   \str_if_empty:NTF \l_tmpa_str {
3585     \msg_set:nnn{stex}{warning/deprecated}{
3586       \\\
3587       \c_backslash_str trefiis~is-deprecated! \\\
3588       Please~use~\c_backslash_str STEXsymbol{#2-#3}![#2-#3s]~instead!~(in~file~
3589       \stex_path_to_string:N \g_stex_currentfile_seq)
3590       \\\ \\\
3591     }

```

```

3592 \msg_warning:nn{stex}{warning/deprecated}
3593 \STEXsymbol { #2-#3 }![ \comp{#2-#3s} ]
3594 } {
3595 \msg_set:nnn{stex}{warning/deprecated}{
3596 \\\
3597 \c_backslash_str trefiis-is-deprecated! \\\
3598 Please~use~\c_backslash_str STEXsymbol { #1 }[ #2-#3s ]~instead!~(in~file~
3599 \stex_path_to_string:N \g_stex_currentfile_seq)
3600 \\\ \\\
3601 }
3602 \msg_warning:nn{stex}{warning/deprecated}
3603 \STEXsymbol { #1 }![ \comp{#2-#3s} ]
3604 }
3605 }
3606
3607 \NewDocumentCommand \symvariant { 0{} m 0{0} m m } {
3608 \msg_set:nnn{stex}{warning/deprecated}{
3609 \\\
3610 \c_backslash_str symvariant-is-deprecated! \\\
3611 Please~use~\c_backslash_str notation[#4]{ #2 }~instead!~(in~file~
3612 \stex_path_to_string:N \g_stex_currentfile_seq)
3613 \\\ \\\
3614 }
3615 \msg_warning:nn{stex}{warning/deprecated}
3616
3617 \notation[variant=#4]{#2}{#5}
3618 }
3619
3620 \NewDocumentCommand \mixfixi { 0{} m m m } {
3621 \msg_set:nnn{stex}{warning/deprecated}{
3622 \c_backslash_str mixfixi-is-fatally-deprecated!\\
3623 Symbol:~\l__stex_term_highlight_uri_str\\
3624 Current~file:~\stex_path_to_string:N \g_stex_currentfile_seq
3625 }
3626 \msg_error:nn{stex}{warning/deprecated}
3627 }
3628
3629
3630 \NewDocumentCommand \infix {} {
3631 \msg_set:nnn{stex}{warning/deprecated}{
3632 \c_backslash_str infix-is-fatally-deprecated!\\
3633 Symbol:~\l__stex_term_highlight_uri_str\\
3634 Current~file:~\stex_path_to_string:N \g_stex_currentfile_seq
3635 }
3636 \msg_error:nn{stex}{warning/deprecated}
3637 }
3638
3639 \let\iprec\infprec
3640
3641 \NewDocumentCommand \inlineex { m } {
3642 \msg_set:nnn{stex}{warning/deprecated}{
3643 \c_backslash_str inlineex-is-deprecated!\\
3644 No~replacement~exists~yet.\\
3645 Current~file:~\stex_path_to_string:N \g_stex_currentfile_seq

```

```

3646 }
3647 \msg_warning:nn{stex}{warning/deprecated}
3648 #1
3649 }
3650
3651
3652 \NewDocumentCommand \term { m } {
3653   \msg_set:nnn{stex}{warning/deprecated}{
3654     \c_backslash_str term~is~deprecated!\\
3655     No~replacement~exists~yet.\\
3656     Current~file:~\stex_path_to_string:N \g_stex_currentfile_seq
3657   }
3658   \msg_warning:nn{stex}{warning/deprecated}
3659   #1
3660 }
3661
3662
3663 \NewDocumentCommand \Definame { 0{} m } {
3664   \stex_get_symbol:n { #2 }
3665   \str_set:Nx \l_tmpa_str {
3666     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3667   }
3668   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3669   \scalatex_if:TF {
3670     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3671       \l_tmpa_str
3672     }
3673   } {
3674     \@defemph {
3675       \exp_after:wN \stex_capitalize:n \l_tmpa_str
3676     } { \l_stex_get_symbol_uri_str }
3677   }
3678 }
3679
3680 \NewDocumentCommand \Symname { 0{} m }{
3681   \stex_symname_args:n { #1 }
3682   \stex_get_symbol:n { #2 }
3683   \str_set:Nx \l_tmpa_str {
3684     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3685   }
3686   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3687   \exp_args:NNx \use:nn
3688   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }![
3689     \exp_after:wN \stex_capitalize:n \l_tmpa_str
3690     \l_stex_symname_post_str
3691   ] }
3692 }
3693
3694
3695 \seq_gput_right:Nx \g_stex_smsmode_allowedenvs_seq { \tl_to_str:n { mhmodnl } }
3696 \seq_gput_right:Nx \g_stex_smsmode_allowedenvs_seq { \tl_to_str:n { modsig } }
3697 \tl_gput_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\gimport\syml\symii\symiii\symiv\
3698
3699 % omtex:

```

```

3700 \cs_new_protected:Npn \lec #1 {
3701   \strut\hfil\strut\null\hfill(#1)
3702 }
3703 \cs_new_protected:Npn \nlex #1 {
3704   \textcolor{green}{\s1 #1}}
3705 }
3706
3707
3708 </compat>

```