

stex.sty: \TeX 2.0*

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

August 29, 2021

Abstract

TODO

1 Introduction

TODO

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

Contents

1	Introduction	1
2	Manual	3
2.1	Semantic Macros	3
2.2	Archives and Imports	4
3	Documentation	5
3.1	Utils	5
3.2	Files, Paths, URIs	6
3.3	MathHub Archives	7
3.4	The Module System	9
3.5	Symbols and Terms	15
4	Implementation	19
4.1	The sTeX document class	19
4.2	Preliminaries	19
4.3	Files, Paths and URIs	24
4.4	MathHub Repositories	28
4.5	Module System	31
4.6	Symbol Declarations	46
4.7	Notations	50
4.8	Terms	58

2 Manual

2.1 Semantic Macros

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 several notations, such as `\notation{mult}{#1 \; #2}`. Typing `\mult{a}{b}` (in math mode) subsequently yields ab .

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

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

For example, we can write the `\mult[product of]{a}{b}`, or `\mult*{a}[\text{the product of } a \text{ and } b]`.

In custom mode, prefixing an argument with a star will not print that argument, but still export it to OMDoc, e.g. `\mult*{a}[multiplying by]{b}` yields $a \cdot b$. A 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 `\forevery*[2]{P}{x} holds for every x in A` to obtain the correctly annotated sentence $P \text{ holds for every } x \in A$.

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; e.g. we can semantically annotate the word *multiplication* with `\mult![multiplication]` or e.g. `the operation \mult![+]`.

2.1.1 Notations and 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 + #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.

\LaTeX 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 *smaller* 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 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 larger than B s argument precedences.

For example, we could set

```
\notation[prec=50]{plus}{#1 + #2}
```

and

```
\notation[prec=100]{times}{#1 \cdot #2}
```

then $\text{\texttt{\$}\texttt{plus}\texttt{\{a\}\texttt{\{\times\{b\}\{c\}\}}}\texttt{\$}}$ would yield $a + b \cdot c$, and $\text{\texttt{\$}\texttt{times}\texttt{\{a\}\texttt{\{plus\{b\}\{c\}\}}}\texttt{\$}}$ would yield $a \cdot (b + c)$.

2.2 Archives and Imports

2.2.1 Namespaces

Ideally, $\text{\texttt{\textsf{S}}\text{\texttt{TEX}}}$ 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, $\text{\texttt{\textsf{T}}\text{\texttt{EX}}}$ 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 $\text{\texttt{\textsf{S}}\text{\texttt{TEX}}}$ can resolve them accordingly. Largely, users need not concern themselves with namespaces at all, but for completeness sake, we describe how they are constructed:

- If $\text{\texttt{\textbackslash begin}\{module\}\{Foo\}}$ occurs in a file $\texttt{/path/to/file/Foo[.⟨lang⟩].tex}$ which does not belong to an archive, the namespace is $\texttt{file://path/to/file}$.
- If the same statement occurs in a file $\texttt{/path/to/file/bar[.⟨lang⟩].tex}$, the namespace is $\texttt{file://path/to/file/bar}$.

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

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

2.2.2 Paths in Import-Statements

Conversely, here is how namespaces/URIs and file paths are computed in import statements, exemplary $\text{\texttt{\textbackslash importmodule}}$:

- $\text{\texttt{\textbackslash importmodule}\{Foo\}}$ 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 $\texttt{Foo[.⟨lang⟩].tex}$ in the same directory.
- 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 $\texttt{Foo[.⟨lang⟩].tex}$ directly in the archive’s **source**-folder.

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

- 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 SCA_{La}TeX, L_AT_EXML and H_TM_L Annotations

<code>\if@latexml</code>	L _A T _E X2e and L _A T _E X3 conditionals for L _A T _E XML.
<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 SCA_{La}TeX) 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.

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: http://mathhub.info/tests/Foo/Bar
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`

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

`\stex_add_constant_to_current_module:n`

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

`\stex_add_import_to_current_module:n`

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

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

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

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

<code>\stex_modules_current_namespace:</code>	
---	--

Computes the current namespace

Test 3

```

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

```

```

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

```

3.4.1 The module-environment

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

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

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

Test 4

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

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

Test 5

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

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

3.4.2 SMS Mode

“SMS Mode” is used when loading modules from external tex files. It deactivates any output and ignores all T_EX 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 6

```
\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 7

```
\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}
```

Module 3.2[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 3.3[Importtest]

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

`\stex_import_module_uri:nn`

`\stex_import_module_uri:nn {<archive-ID>} {<module-path>}`

Determines the URI of a module by splitting `<module-path>` into `<path>?<name>`. If `<module-path>` does *not* contain a `?`-character, we consider it to be the `<name>`, and `<path>` to be empty.

If `<archive-ID>` is empty, it is automatically set to the ID of the current archive (if one exists).

1. If `<archive-ID>` is empty:

(a) If `<path>` is empty, then `<name>` must have been declared earlier in the same file and retrievable from `\g_stex_modules_in_file_seq`, or a file with name `<name>.<lang>.tex` must exist in the same folder, containing a module `<name>`.

That module should have the same namespace as the current one.

(b) If `<path>` is not empty, it must point to the relative path of the containing file as well as the namespace.

2. Otherwise:

(a) If `<path>` is empty, then `<name>` must have been declared earlier in the same file and retrievable from `\g_stex_modules_in_file_seq`, or a file with name `<name>.<lang>.tex` must exist in the top `source` folder of the archive, containing a module `<name>`.

That module should lie directly in the namespace of the archive.

(b) If `<path>` 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 `\stex_require_module:nn` on the `source` directory of the archive to find the file.

`\stex_import_require_module:nnnn`

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

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.

`\g_stex_module_files_prop`

`\g_stex_modules_in_file_seq`

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).

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 $\text{\S}\text{\TeX}$, 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 $\text{\S}\text{\TeX}$ 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\text{\Nat}}{x\geq 0}`.

`\abbrdef` `\abbrdef[⟨args⟩]{⟨macroname⟩}{⟨term⟩}`

`\abbrdef` behaves like `\symdecl`, but adds the definiens `⟨term⟩` to the symbol. The latter is largely ignored and irrelevant to $\text{\S}\text{\TeX}$, but exported to OMDOC.

`\stex_symdecl_do:n`

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

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 8

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

```
Module 3.4[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}<
```

`\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...

`\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),
- `argprec` (sequence of integer strings)

Test 9

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

Module 3.5[NotationTest]

\symdef `\symdef[⟨args⟩]{⟨symbol⟩}{⟨notations+⟩}`

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

Test 10

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

Module 3.6[SymdefTest]
 $a + b + c$

\stex_term_oms:nnnn `⟨URI⟩⟨fragment⟩⟨precedence⟩⟨body⟩`

\stex_term_oma:nnnn
\stex_term_omb:nnnn

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

\stex_term_arg:nnn `\stex_term_arg:nnn⟨int⟩⟨prec⟩⟨body⟩`

Annotates $\langle body \rangle$ as the $\langle int \rangle$ th argument of the current OMA or OMBIND, with (downwards) argument precedence $\langle prec \rangle$.

\stex_term_assoc_arg:nnnn `\stex_term_arg:nnn⟨int⟩⟨prec⟩⟨notation⟩⟨body⟩`

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

\infprec
\neginfprec

Maximal and minimal notation precedences.

\STEXdobrackets**\STEXdobrackets** $\langle body \rangle$

Puts $\langle body \rangle$ in parentheses; scaled if in display mode unscaled otherwise. Uses the current \TeX brackets (by default (and)), which can be changed temporarily using **\STEXwithbrackets**.

\STEXwithbrackets**\STEXwithbrackets** $\langle left \rangle \langle right \rangle \langle body \rangle$

Temporarily (i.e. within $\langle body \rangle$) sets the brackets used by \TeX for automated bracketing (by default (and)) to $\langle left \rangle$ and $\langle right \rangle$.

Note that $\langle left \rangle$ and $\langle right \rangle$ need to be allowed after **\left** and **\right** in display-mode.

Test 11

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

Module 3.7[MathTest1]
 $\langle a^b_c \rangle$ and $\langle a^b_c \rangle$.

Test 12

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

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

Module 3.8[MathTest2]
 $\langle a \mid [b:c;d:e,f]^g \rangle$ and $\langle a \mid [b:c]^g \rangle$ and $\langle a \mid [b]^c \rangle$
 $a + b \cdot c$ and $a \cdot (\frac{a}{b} + \frac{a}{c})$
 $a + b \cdot c$ and $a \cdot (\frac{a}{b} + \frac{a}{c})$
 $a + b \cdot c$ and $a \cdot [\frac{a}{b} + \frac{a}{c}]$

\stex_term_custom:nn**\stex_term_custom:nn**{ $\langle URI \rangle$ }{ $\langle args \rangle$ }

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 13

```
\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![\mathhtt{bar}]\$

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

\bar![bar]

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

\end{module}
```

```
Module 3.9[TextTest]
  some a and some b and also some c here.
  some a and some b and also some c here.
  bar
  or just some c
  bar
  or first b, then c, and finally a
```

4 Implementation

4.1 The \TeX document class

```
1 <*\cls>
2 \RequirePackage{expl3, l3keys2e}
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 }
```

```

23 \ProcessKeysOptions { stex }

\TeX The  $\TeX$  logo:
24 \protected\def\stex{%
25   \@ifundefined{texorpdfstring}%
26   {\let\texorpdfstring\@firstoftwo}%
27   }%
28   \texorpdfstring{\raisebox{-.5ex}S\kern-.5ex\TeX}{sTeX}\xspace%
29 }
30 \def\TeX{\stex}

```

(End definition for \TeX. This function is documented on page 5.)

Messages

```

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

```

\stex_debug:n Debug mode

```

38 \cs_new_protected:Nn \stex_debug:n {
39   \bool_if:nT{\c_stex_debug_bool}{
40     \exp_args:Nnn\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 5.)

\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_addtoms:n`. This function is documented on page 5.)

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 L^AT_EX_ML:

```
\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 5.)

4.2.2 HTML Annotations

```
77 <@=stex_annotate>
```

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

```
\c__stex_annotate_emptyarg_tl
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_&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:nnx`

We define four macros for introducing attributes in the HTML output. The definitions depend on the “backend” used (L^AT_EX_ML, S^CA_LT_EX, pdf_lat_ex).

The pdf_lat_ex-macros largely do nothing; the S^CA_LT_EX-implementations are pretty clear in what they do, the L^AT_EX_ML-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 6.)

4.2.3 Languages

```

173 <@@=stex_language>

```

`\c_stex_languages_prop` We store language abbreviations in two (mutually inverse) property lists:
`\c_stex_language_abbrevs_prop`

```

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 6.)

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 6.)

`\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 6.)

`\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:NNTF \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 6.)

```

\stex_path_if_absolute_p:N
\stex_path_if_absolute:N $\underline{TF}$ 

```

```

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

```

(End definition for `\stex_path_if_absolute:N \underline{TF}` . This function is documented on page 6.)

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 5.)

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 7.)

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/\g_file_curr_name_str.tex`
 309 `}`

(End definition for `\c_stex_mainfile_seq`. This variable is documented on page 7.)

`\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 7.)

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     \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_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
348   \stex_debug:n {MathHub:~\str_use:N\c_stex_mathhub_str}
349 }

```

(End definition for `\mathhub`, `\c_stex_mathhub_seq`, and `\c_stex_mathhub_str`. These variables are documented on page 7.)

```

\__stex_mathhub_do_manifest:n
350 \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
351   \str_set:Nx \l_tmpa_str { #1 }
352   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
353     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
354     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
355     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
356     \__stex_mathhub_find_manifest:N \l_tmpa_seq
357     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
358       \msg_set:nnn{stex}{error/norepository}{
359         No~archive~#1~found~in~
360         \stex_path_to_string:N \c_stex_mathhub_str
361       }
362       \msg_error:nn{stex}{error/norepository}
363     } {
364       \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
365     }
366   }
367 }

```

(End definition for `__stex_mathhub_do_manifest:n`.)

```

\l__stex_mathhub_manifest_file_seq
368 \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`:

```

369 \cs_new_protected:Nn \__stex_mathhub_find_manifest:N {
370   \seq_set_eq:NN\l_tmpa_seq #1

```

```

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

```

(End definition for `__stex_mathhub_find_manifest:N`.)

`\c_stex_mathhub_manifest_ior` File variable used for MANIFEST-files

```

404 \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:

```

405 \cs_new_protected:Nn \_stex_mathhub_parse_manifest:n {
406   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
407   \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
408   \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
409     \str_set:Nn \l_tmpa_str {##1}
410     \exp_args:NNoo \seq_set_split:Nnn
411       \l_tmpb_seq \c_colon_str \l_tmpa_str
412     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
413       \exp_args:NNe \str_set:Nn \l_tmpb_tl {
414         \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
415       }
416       \exp_args:No \str_case:nnTF \l_tmpa_tl {

```

```

417 {id} {
418   \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
419   { id } \l_tmpb_tl
420 }
421 {narration-base} {
422   \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
423   { narr } \l_tmpb_tl
424 }
425 {source-base} {
426   \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
427   { ns } \l_tmpb_tl
428 }
429 {ns} {
430   \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
431   { ns } \l_tmpb_tl
432 }
433 {dependencies} {
434   \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
435   { deps } \l_tmpb_tl
436 }
437 }{}{}
438 }{}
439 }
440 \ior_close:N \c__stex_mathhub_manifest_ior
441 }

```

(End definition for `_stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

442 \cs_new_protected:Nn \stex_set_current_repository:n {
443   \stex_require_repository:n { #1 }
444   \prop_set_eq:Nc \l_stex_current_repository_prop {
445     c_stex_mathhub_#1_manifest_prop
446   }
447 }

```

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

`\stex_require_repository:n`

```

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

```

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

`\l_stex_current_repository_prop` Current MathHub repository and a hook for `\begin{document}` to set it initially.

```

462 \prop_new:N \l_stex_current_repository_prop
463 \AddToHook{begindocument}{
464   \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
465   \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
466     \stex_debug:n{Not~currently~in~a~MathHub~repository}
467   } {
468     \__stex_mathhub_parse_manifest:n { main }
469     \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
470     \l_tmpa_str
471     \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str _manifest_prop }
472     \stex_set_current_repository:n { main }
473     \stex_debug:n{Current~repository:~
474     \prop_item:Nn \l_stex_current_repository_map {id}
475   }
476 }
477 }

```

(End definition for `\l_stex_current_repository_prop`. This variable is documented on page 8.)

4.5 Module System

```

478 <@@=stex_module>

```

`\l_stex_current_module_prop`

```

479 \prop_new:N \l_stex_current_module_prop

```

(End definition for `\l_stex_current_module_prop`. This variable is documented on page 9.)

`stex_if_in_module_p:`

`stex_if_in_module:TF`

```

480 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
481   \prop_if_empty:NTF \l_stex_current_module_prop
482   \prg_return_false: \prg_return_true:
483 }

```

(End definition for `stex_if_in_module:TF`. This function is documented on page 9.)

`stex_if_module_exists_p:n`

`stex_if_module_exists:nTF`

```

484 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
485   \prop_if_exist:cTF { c_stex_module_#1_prop }
486   \prg_return_true: \prg_return_false:
487 }

```

(End definition for `stex_if_module_exists:nTF`. This function is documented on page 9.)

`\stex_add_to_current_module:n`

```

488 \cs_new_protected:Nn \stex_add_to_current_module:n {
489   \prop_get:NnN \l_stex_current_module_prop { content } \l_tmpa_tl
490   \tl_put_right:Nn \l_tmpa_tl { #1 }
491   \prop_put:Nno \l_stex_current_module_prop { content } \l_tmpa_tl
492 }

```

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

`\stex_add_constant_to_current_module:n`

```
493 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {  
494   \str_set:Nx \l_tmpa_str { #1 }  
495   \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq  
496   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }  
497   \prop_put:Nno \l_stex_current_module_prop { constants } \l_tmpa_seq  
498 }
```

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

`\stex_add_import_to_current_module:n`

```
499 \cs_new_protected:Nn \stex_add_import_to_current_module:n {  
500   \str_set:Nx \l_tmpa_str { #1 }  
501   \prop_get:NnN \l_stex_current_module_prop { imports } \l_tmpa_seq  
502   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }  
503   \prop_put:Nno \l_stex_current_module_prop { imports } \l_tmpa_seq  
504 }
```

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

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

```
\l_stex_modules_ns_str 505 \str_new:N \l_stex_modules_ns_str  
  
506 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {  
507   \str_set:Nx \l_tmpa_str { #1 }  
508   \seq_set_eq:NN \l_tmpa_seq #2  
509   % split off file extension  
510   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str  
511   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str  
512   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str  
513   \seq_put_right:No \l_tmpa_seq \l_tmpb_str  
514  
515   \bool_set_true:N \l_tmpa_bool  
516   \bool_while_do:Nn \l_tmpa_bool {  
517     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str  
518     \exp_args:No \str_case:nnTF { \l_tmpb_str } {  
519       {source} { \bool_set_false:N \l_tmpa_bool }  
520     }{}{  
521       \seq_if_empty:NT \l_tmpa_seq {  
522         \bool_set_false:N \l_tmpa_bool  
523       }  
524     }  
525   }  
526  
527   \seq_if_empty:NTF \l_tmpa_seq {  
528     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str  
529   }{  
530     \str_set:Nx \l_stex_modules_ns_str {  
531       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq  
532     }  
533   }  
534 }
```


(End definition for `\stex_modules_compute_namespace:nN` and `\l_stex_modules_ns_str`. These functions are documented on page 10.)

`\stex_modules_current_namespace:`

```

535 \cs_new_protected:Nn \stex_modules_current_namespace: {
536   \prop_get:NnNTF \l_stex_current_repository_prop { ns } \l_tmpa_str {
537     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
538   }{
539     % split off file extension
540     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
541     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
542     \exp_args:NNo \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
543     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
544     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
545     \str_set:Nx \l_stex_modules_ns_str {
546       file:/\stex_path_to_string:N \l_tmpa_seq
547     }
548   }
549 }

```

(End definition for `\stex_modules_current_namespace:.` This function is documented on page 10.)

4.5.1 The module environment

module module arguments:

```

550 \keys_define:nn { stex / module } {
551   title .tl_set_x:N = \l_stex_module_title_str ,
552   ns .tl_set_x:N = \l_stex_module_ns_str ,
553   lang .tl_set_x:N = \l_stex_module_lang_str ,
554   sig .tl_set_x:N = \l_stex_module_sig_str ,
555   meta .tl_set_x:N = \l_stex_module_meta_str
556 }
557
558 % module parameters here? In the body?
559
560 \cs_new_protected:Nn \__stex_module_args:n {
561   \str_clear:N \l_stex_module_title_str
562   \str_clear:N \l_stex_module_ns_str
563   \str_clear:N \l_stex_module_lang_str
564   \str_clear:N \l_stex_module_sig_str
565   \str_clear:N \l_stex_module_meta_str
566   \keys_set:nn { stex / module } { #1 }
567   \exp_args:NNo \str_set:Nn \l_stex_module_title_str
568     \l_stex_module_title_str
569   \exp_args:NNo \str_set:Nn \l_stex_module_ns_str
570     \l_stex_module_ns_str
571   \exp_args:NNo \str_set:Nn \l_stex_module_lang_str
572     \l_stex_module_lang_str
573   \exp_args:NNo \str_set:Nn \l_stex_module_sig_str
574     \l_stex_module_sig_str
575   \exp_args:NNo \str_set:Nn \l_stex_module_meta_str
576     \l_stex_module_meta_str
577 }

```

```

\__stex_module_begin_module: implements \begin{module}

578 \cs_new_protected:Nn \__stex_module_begin_module: {
579   % Nested module?
580   \stex_if_in_module:TF {
581     % Nested module
582     \prop_get:NnN \l_stex_current_module_prop
583       { ns } \l_stex_module_ns_str
584     \str_set:Nx \l_stex_module_name_str {
585       \prop_item:Nn \l_stex_current_module_prop
586         { name } / \l_stex_module_name_str
587     }
588   }{
589     % not nested:
590     \str_if_empty:NT \l_stex_module_ns_str {
591       \stex_modules_current_namespace:
592       \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
593       \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
594         / {\l_stex_module_ns_str}
595       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
596       \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
597         \str_set:Nx \l_stex_module_ns_str {
598           \stex_path_to_string:N \l_tmpa_seq
599         }
600       }
601     }
602   }
603
604   % language
605   \str_if_empty:NF \l_stex_module_lang_str {
606     \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
607       \l_tmpa_str {
608         \exp_args:Nx \selectlanguage { \l_tmpa_str }
609       } {
610         \msg_set:nnn{stex}{error/unknownlanguage}{
611           Unknown~language~\l_tmpa_str
612         }
613         \msg_error:nn{stex}{error/unknownlanguage}
614       }
615   }
616
617   % signature
618   \str_if_empty:NF \l_stex_module_sig_str {
619     \str_if_empty:NT \l_stex_module_lang_str {
620       \msg_set:nnn{stex}{error/siglanguage}{
621         Module~\l_stex_module_ns_str?\l_stex_module_name_str~
622         declares~signature~\l_stex_module_sig_str,~but~does~not~
623         declare~its~language
624       }
625       \msg_error:nn{stex}{error/siglanguage}
626     }
627   }
628
629   % metatheory
630   % \str_if_empty:NTF \l_stex_module_meta_str {

```

```

631 %
632 % } {
633 %
634 % }
635
636 \str_clear:N \l_tmpa_str
637 \seq_clear:N \l_tmpa_seq
638 \tl_clear:N \l_tmpa_tl
639 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
640   name      = \l_stex_module_name_str ,
641   ns        = \l_stex_module_ns_str ,
642   imports   = \exp_not:o { \l_tmpa_seq } ,
643   constants = \exp_not:o { \l_tmpa_seq } ,
644   content   = \exp_not:o { \l_tmpa_tl } ,
645   file      = \exp_not:o { \g_stex_currentfile_seq } ,
646   lang      = \l_stex_module_lang_str ,
647   sig       = \l_stex_module_sig_str ,
648   meta      = \l_stex_module_meta_str
649 }
650
651 \stex_debug:n{
652   New~module:\\
653   Namespace:~\l_stex_module_ns_str\\
654   Name:~\l_stex_module_name_str\\
655   Language:~\l_stex_module_lang_str\\
656   Signature:~\l_stex_module_sig_str\\
657   Metatheory:~\l_stex_module_meta_str\\
658   File:~\stex_path_to_string:N \g_stex_currentfile_seq
659 }
660
661 \seq_gput_right:Nx \g_stex_modules_in_file_seq
662   { \l_stex_module_ns_str ? \l_stex_module_name_str }
663
664 \stex_if_smsmode:TF {
665   \stex_smsmode_set_codes:
666 } {
667   \begin{stex_annotate_env} {theory} {
668     \l_stex_module_ns_str ? \l_stex_module_name_str
669   }
670
671   \stex_annotate_invisible:nnn{header}{} {
672     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
673     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
674     \str_if_empty:NT \l_stex_module_meta_str {
675       % TODO metatheory
676     }
677   }
678 }
679 }
680 \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}

```

681 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
682 \cs_new_protected:Nn \__stex_module_end_module: {
683   \str_set:Nx \l_tmpa_str {
684     c_stex_module_
685     \prop_item:Nn \l_stex_current_module_prop { ns } ?
686     \prop_item:Nn \l_stex_current_module_prop { name }
687     _prop
688   }
689   \prop_new:c { \l_tmpa_str }
690   \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
691   \stex_if_smsmode:TF {
692     \exp_args:Nx \stex_addtosms:n {
693       \prop_gset_from_keyval:cn {
694         c_stex_module_
695         \prop_item:Nn \l_stex_current_module_prop { ns } ?
696         \prop_item:Nn \l_stex_current_module_prop { name }
697         _prop
698       } {
699         name      = \prop_item:cn { \l_tmpa_str } { name } ,
700         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
701         imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
702         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
703         content   = \prop_item:cn { \l_tmpa_str } { content } ,
704         file      = \prop_item:cn { \l_tmpa_str } { file } ,
705         lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
706         sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
707         meta      = \prop_item:cn { \l_tmpa_str } { meta }
708       }
709     }
710   }{
711     \end{stex_annotate_env}
712   }
713 }

```

(End definition for __stex_module_end_module:.)

@module The core environment, with no header

```

714 \NewDocumentEnvironment { @module } { 0{} m } {
715   \str_set:Nx \l_stex_module_name_str { #2 }
716   \par
717   \__stex_module_args:n { #1 }
718   \__stex_module_begin_module:
719 } {
720   \__stex_module_end_module:
721 }

```

\stex_modules_heading: Code for document headers

```

722 \cs_if_exist:NTF \thesection {
723   \newcounter{module}[section]
724 }{
725   \newcounter{module}
726 }
727
728 \bool_if:NT \c_stex_showmods_bool {

```

```

729 \latexml_if:F { \RequirePackage{mdframed} }
730 }
731
732 \cs_new_protected:Nn \stex_modules_heading: {
733   \stepcounter{module}
734   \par
735   \bool_if:NT \c_stex_showmods_bool {
736     \noindent{\textbf{Module} ~
737       \cs_if_exist:NT \thesection {\thesection.}
738       \themodule ~ [\l_stex_module_name_str]
739     }
740     % TODO references
741     % \sref@label@id{Module \thesection.\themodule [\module@name]]%
742     \str_if_empty:NTF \l_stex_module_title_str {
743       }{
744         \quad(\l_stex_module_title_str)\hfill
745       }\par
746     }
747   }

```

(End definition for `\stex_modules_heading:`. This function is documented on page 10.)

Finally:

```

748 \NewDocumentEnvironment { module } { 0{} m } {
749   \bool_if:NT \c_stex_showmods_bool {
750     \begin{mdframed}
751   }
752   \begin{@module}[#1]{#2}
753   \stex_modules_heading:
754   }{
755     \end{@module}
756     \bool_if:NT \c_stex_showmods_bool {
757       \end{mdframed}
758     }
759   }

```

4.5.2 SMS Mode

```

760 <@@=stex_smsmode>

```

```

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq
761 \tl_new:N \g_stex_smsmode_allowedmacros_tl
762 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
763 \seq_new:N \g_stex_smsmode_allowedenvs_seq
764
765 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
766   \makeatletter
767   \makeatother
768   \ExplSyntaxOn
769   \ExplSyntaxOff
770 }
771
772 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
773   \symdef
774   \abbrdef

```

```

775 % \module@export
776 \importmodule
777 % \mmt@symdecl
778 % \instantiates
779 % \setnotation
780 % \importmhmodule
781 % \gimport
782 % \symvariant
783 % \structural@feature
784 % \symi
785 % \symii
786 % \symiii
787 % \symiv
788 \notation
789 \symdecl
790 % \defi
791 % \defii
792 % \defiii
793 % \defiv
794 % \adefi
795 % \adefii
796 % \adefiii
797 % \adefiv
798 % \defis
799 % \defiis
800 % \defiiis
801 % \defivs
802 % \Defi
803 % \Defii
804 % \Defiii
805 % \Defiv
806 % \Defis
807 % \Defiis
808 % \Defiiis
809 % \Defivs
810 }
811
812 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
813   \tl_to_str:n {
814     module,
815     @module
816 %   modsig,
817 %   mhmodsig,
818 %   mhmodnl,
819 %   modnl,
820 %   @structural@feature
821   }
822 }

```

(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 11.)

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

```

823 \bool_new:N \g__stex_smsmode_bool

```

```

824 \bool_set_false:N \g__stex_smsmode_bool
825 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
826   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
827 }

```

(End definition for \stex_if_smsmode:TF. This function is documented on page 12.)

```

\stex_smsmode_if_catcodes_p: Checks whether the SMS mode category code scheme is active.
__stex_smsmode_if_catcodes:TF
828 \bool_new:N \g__stex_smsmode_catcode_bool
829 \bool_set_false:N \g__stex_smsmode_catcode_bool
830 \prg_new_conditional:Nnn __stex_smsmode_if_catcodes: { p, T, F, TF } {
831   \bool_if:NTF \g__stex_smsmode_catcode_bool
832     \prg_return_true: \prg_return_false:
833 }

```

(End definition for __stex_smsmode_if_catcodes:TF.)

\stex_smsmode_set_codes:

```

834 \cs_new_protected:Nn \stex_smsmode_set_codes: {
835   \stex_if_smsmode:T {
836     __stex_smsmode_if_catcodes:F {
837       \bool_gset_true:N \g__stex_smsmode_catcode_bool
838       \exp_after:wN \char_gset_active_eq:NN
839         \c_backslash_str __stex_smsmode_cs:
840       \tex_global:D \char_set_catcode_active:N \
841       \tex_global:D \char_set_catcode_other:N $
842       \tex_global:D \char_set_catcode_other:N ^
843       \tex_global:D \char_set_catcode_other:N _
844       \tex_global:D \char_set_catcode_other:N &
845       \tex_global:D \char_set_catcode_other:N ##
846     }
847   }
848 } \iffalse $ \fi % to make syntax highlighting work again

```

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

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

```

849 \cs_new_protected:Nn __stex_smsmode_unset_codes: {
850   __stex_smsmode_if_catcodes:T {
851     \bool_gset_false:N \g__stex_smsmode_catcode_bool
852     \exp_after:wN \tex_global:D \exp_after:wN
853       \char_set_catcode_escape:N \c_backslash_str
854     \tex_global:D \char_set_catcode_math_toggle:N $
855     \tex_global:D \char_set_catcode_math_superscript:N ^
856     \tex_global:D \char_set_catcode_math_subscript:N _
857     \tex_global:D \char_set_catcode_alignment:N &
858     \tex_global:D \char_set_catcode_parameter:N ##
859   }
860 } \iffalse $ \fi % to make syntax highlighting work again

```

(End definition for __stex_smsmode_unset_codes:.)

`\stex_in_smsmode:nn`

```

861 \cs_new_protected:Nn \stex_in_smsmode:nn {
862   \vbox_set:Nn \l_tmpa_box {
863     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
864     \bool_gset_true:N \g__stex_smsmode_bool
865     \stex_smsmode_set_codes:
866     #2
867     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
868     \stex_if_smsmode:F {
869       \__stex_smsmode_unset_codes:
870     }
871   }
872   \box_clear:N \l_tmpa_box
873 }

```

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

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

```

874 \str_const:Nn \c__stex_smsmode_begin_str { begin }
875 \str_const:Nn \c__stex_smsmode_end_str { end }
876
877 \cs_new_protected:Nn \__stex_smsmode_cs: {
878   \str_clear:N \l_tmpa_str
879   \peek_analysis_map_inline:n {
880     % #1: token (one expansion)
881     % #2: charcode
882     % #3 catcode
883     \token_if_eq_charcode:NNTF ##3 B {
884       % token is a letter
885       \exp_args:NNo \str_put_right:Nn \l_tmpa_str { ##1 }
886     } {
887       \str_if_empty:NTF \l_tmpa_str {
888         % we don't allow (or need) single non-letter CSs
889         % for now
890         \peek_analysis_map_break:
891       }{
892         \str_if_eq:nnTF \l_tmpa_str \c_stex_begin_str {
893           \peek_analysis_map_break:n {
894             \exp_after:wN \__stex_smsmode_checkbegin:n ##1
895           }
896         } {
897           \str_if_eq:nnTF \l_tmpa_str \c_stex_end_str {
898             \peek_analysis_map_break:n {
899               \exp_after:wN \__stex_smsmode_checkend:n ##1
900             }
901           } {
902             \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
903             \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
904               \g_stex_smsmode_allowedmacros_tl
905               { \use:c{\l_tmpa_str} } {
906               \peek_analysis_map_break:n {
907                 \exp_after:wN \l_tmpa_tl ##1
908               }
909             }
910           }
911         }
912       }
913     }
914   }
915 }

```



```

909     } {
910         \exp_args:NNNo \exp_args:NNo \tl_if_in:NnTF
911         \g_stex_smsmode_allowedmacros_escape_tl
912         { \use:c{\l_tmpa_str} } {
913             \exp_args:NNNo \exp_args:No
914             \token_if_eq_charcode_p:NNTF \c_backslash_str ##1 {
915                 \peek_analysis_map_break:n {
916                     \__stex_smsmode_unset_codes:
917                     \__stex_smsmode_rescan_cs:
918                 }
919             } {
920                 \peek_analysis_map_break:n {
921                     \__stex_smsmode_unset_codes:
922                     \exp_after:wN \l_tmpa_tl ##1
923                 }
924             }
925         } {
926             \peek_analysis_map_break:n { ##1 }
927         }
928     }
929 }
930 }
931 }
932 }
933 }
934 }

```

(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:

```

935 \cs_new_protected:Nn \__stex_smsmode_rescan_cs: {
936     \str_clear:N \l_tmpb_str
937     \peek_analysis_map_inline:n {
938         \token_if_eq_charcode:NNTF ##3 B {
939             % token is a letter
940             \exp_args:NNo \str_put_right:Nn \l_tmpb_str { ##1 }
941         } {
942             \peek_analysis_map_break:n {
943                 \exp_after:wN \use:c \exp_after:wN {
944                     \exp_after:wN \l_tmpa_str\exp_after:wN
945                 } \use:c { \l_tmpb_str \exp_after:wN } ##1
946             }
947         }
948     }
949 }

```

(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.

```

950 \cs_new_protected:Nn \__stex_smsmode_checkbegin:n {
951     \str_set:Nn \l_tmpa_str { #1 }
952     \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
953         \__stex_smsmode_unset_codes:

```

```

954     \begin{#1}
955   }
956 }

```

(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.

```

957 \cs_new_protected:Nn \_stex_smsmode_checkend:n {
958   \str_set:Nn \l_tmpa_str { #1 }
959   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
960     \end{#1}
961   }
962 }

```

(End definition for `_stex_smsmode_checkend:n`.)

4.5.3 Inheritance

```

963 <@@=stex_importmodule>

```

`\stex_import_module_uri:nn`

```

964 \cs_new_protected:Nn \stex_import_module_uri:nn {
965   \str_set:Nx \l__stex_importmodule_archive_str { #1 }
966   \str_set:Nx \l__stex_importmodule_path_str { #2 }
967   \str_if_empty:NT \l__stex_importmodule_archive_str {
968     \prop_if_empty:NF \l_stex_current_repository_prop {
969       \prop_get:NnN \l_stex_current_repository_prop { id } \l__stex_importmodule_archive_str
970     }
971   }
972
973   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l__stex_importmodule_path_str }
974   \seq_pop_right:NN \l_tmpb_seq \l__stex_importmodule_name_str
975   \str_set:Nx \l__stex_importmodule_path_str { \seq_use:Nn \l_tmpa_seq ? }
976
977   \str_if_empty:NTF \l_tmpa_str {
978     \stex_modules_current_namespace:
979     \str_if_empty:NF \l__stex_importmodule_path_str {
980       \str_set:Nx \l_stex_module_ns_str {
981         \l_stex_module_ns_str / \l__stex_importmodule_path_str
982       }
983     }
984   }{
985     \stex_require_repository:n \l__stex_importmodule_archive_str
986     \prop_get:cnN { c_stex_mathhub_\l__stex_importmodule_archive_str _manifest_prop } { ns }
987     \l_stex_module_ns_str
988     \str_if_empty:NF \l__stex_importmodule_path_str {
989       \str_set:Nx \l__stex_importmodule_module_ns_str {
990         \l_stex_module_ns_str / \l__stex_importmodule_path_str ? \l__stex_importmodule_name_
991       }
992     }
993   }
994 }

```

(End definition for `\stex_import_module_uri:nn`. This function is documented on page 14.)

```

\l__stex_importmodule_name_str Store the return values of \stex_import_module_uri:nn.
\l__stex_importmodule_archive_str 995 \str_new:N \l__stex_importmodule_name_str
\l__stex_importmodule_path_str 996 \str_new:N \l__stex_importmodule_archive_str
997 \str_new:N \l__stex_importmodule_path_str

(End definition for \l__stex_importmodule_name_str, \l__stex_importmodule_archive_str, and \l__
_stex_importmodule_path_str.)

\stex_import_require_module:nnnn {<ns>} {<archive-ID>} {<path>} {<name>}
998 \cs_new_protected:Nn \stex_import_require_module:nnnn {
999 \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1000 % archive
1001 \str_set:Nx \l_tmpa_str { #2 }
1002 \str_if_empty:NTF \l_tmpa_str {
1003 \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1004 } {
1005 \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1006 \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1007 \seq_put_right:Nn \l_tmpa_seq { source }
1008 }
1009
1010 \stex_debug:n{Arguments: #1, #2, #3, #4}
1011
1012 % path
1013 \str_set:Nx \l_tmpb_str { #3 }
1014 \str_if_empty:NT \l_tmpb_str {
1015 \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1016
1017 \cs_if_exist:NTF \language_name {
1018 \prop_get:NnN \c_stex_language_abbrevs_prop
1019 { \language_name } \l_tmpb_str
1020 }
1021
1022 \stex_debug:n{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1023 \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1024 \str_set:Nx \l_tmpa_str { \l_tmpa_str.\l_tmpb_str.tex }
1025 }{
1026 \stex_debug:n{Checking~\l_tmpa_str.tex}
1027 \IfFileExists{ \l_tmpa_str.tex }{
1028 \str_set:Nx \l_tmpa_str { \l_tmpa_str.tex }
1029 }{
1030 % try english as default
1031 \stex_debug:n{Checking~\l_tmpa_str.en.tex}
1032 \IfFileExists{ \l_tmpa_str.en.tex }{
1033 \str_set:Nx \l_tmpa_str { \l_tmpa_str.en.tex }
1034 }{
1035 \msg_new:nnn{stex}{error/modulemissing}{
1036 No~file~for~module~#1?#4~found
1037 }
1038 \msg_error:nn{stex}{error/modulemissing}
1039 }
1040 }
1041 }
1042

```

```

1043 } {
1044   \stex_path_from_string:NV \l_tmpb_seq \l_tmpb_str
1045   \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1046
1047   \cs_if_exist:NTF \language_name {
1048     \prop_get:NnN \c_stex_language_abbrevs_prop
1049       { \language_name } \l_tmpb_str
1050   }
1051
1052   \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq }
1053
1054   \stex_debug:n{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1055   \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1056     \str_set:Nx \l_tmpa_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1057   }{
1058     \stex_debug:n{Checking~\l_tmpa_str/#4.tex}
1059     \IfFileExists{ \l_tmpa_str/#4.tex }{
1060       \str_set:Nx \l_tmpa_str { \l_tmpa_str/#4.tex }
1061     }{
1062       % try english as default
1063       \stex_debug:n{Checking~\l_tmpa_str/#4.en.tex}
1064       \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1065         \str_set:Nx \l_tmpa_str { \l_tmpa_str/#4.en.tex }
1066       }{
1067         \stex_debug:n{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1068         \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1069           \str_set:Nx \l_tmpa_str { \l_tmpa_str.\l_tmpb_str.tex }
1070         }{
1071           \stex_debug:n{Checking~\l_tmpa_str.tex}
1072           \IfFileExists{ \l_tmpa_str.tex }{
1073             \str_set:Nx \l_tmpa_str { \l_tmpa_str.tex }
1074           }{
1075             % try english as default
1076             \stex_debug:n{Checking~\l_tmpa_str.en.tex}
1077             \IfFileExists{ \l_tmpa_str.en.tex }{
1078               \str_set:Nx \l_tmpa_str { \l_tmpa_str.en.tex }
1079             }{
1080               \msg_new:nnn{stex}{error/modulemissing}{
1081                 No~file~for~module~#1?#4~found
1082               }
1083               \msg_error:nn{stex}{error/modulemissing}
1084             }
1085           }
1086         }
1087       }
1088     }
1089   }
1090 }
1091
1092 \seq_set_eq:NN \l_tmpa_seq \g_stex_modules_in_file_seq
1093 \seq_clear:N \g_stex_modules_in_file_seq
1094 \exp_args:No \stex_in_smsmode:nn { \l_tmpa_str } {
1095   \str_set:Nx \l_tmpb_str { #2 }
1096   \str_if_empty:NF \l_tmpb_str {

```

```

1097     \stex_set_current_repository:n { #2 }
1098   }
1099   \input { \l_tmpa_str }
1100 }
1101 \prop_gput:Noo \g_stex_module_files_prop
1102   \l_tmpa_str \g_stex_modules_in_file_seq
1103 \seq_set_eq:NN \g_stex_modules_in_file_seq \l_tmpa_seq
1104
1105 \stex_if_module_exists:nF { #1 ? #4 } {
1106   \msg_new:nnn{stex}{error/modulemissing}{
1107     Module~#1?#4~not~found~in~file~\l_tmpa_str
1108   }
1109   \msg_error:nn{stex}{error/modulemissing}
1110 }
1111 }
1112 % activate
1113 \stex_debug:n{Activating~module~#1?#4}
1114 \prop_item:cn { c_stex_module_#1?#4_prop } { content }
1115 }

```

(End definition for `\stex_import_require_module:nnnn`. This function is documented on page 14.)

`\importmodule`

```

1116 \NewDocumentCommand \importmodule { 0{} m } {
1117   \stex_import_module_uri:nn { #1 } { #2 }
1118   \stex_debug:n{Importing~module:~
1119     \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1120   }
1121   \stex_if_smsmode:F {
1122     \stex_import_require_module:nnnn
1123     { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1124     { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1125     \stex_annotate_invisible:nnn
1126     {import} { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } {}
1127   }
1128   \exp_args:Nx \stex_add_to_current_module:n {
1129     \stex_import_require_module:nnnn
1130     { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1131     { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1132   }
1133   \exp_args:Nx \stex_add_import_to_current_module:n {
1134     \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1135   }
1136   \stex_smsmode_set_codes:
1137 }

```

(End definition for `\importmodule`. This function is documented on page 12.)

`\usemodule`

```

1138 \NewDocumentCommand \usemodule { 0{} m } {
1139   \stex_if_smsmode:F {
1140     \stex_import_module_uri:nn { #1 } { #2 }
1141     \stex_import_require_module:nnnn
1142     { \l__stex_importmodule_module_ns_str } { \l__stex_importmodule_archive_str }
1143     { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }

```

```

1144 \stex_annotate_invisible:nnn
1145 {usemodule} {\l_stex_module_ns_str ? \l__stex_importmodule_name_str} {}
1146 }
1147 \stex_smsmode_set_codes:
1148 }

```

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

`\g_stex_modules_in_file_seq`
`\g_stex_module_files_prop`

```

1149 \seq_new:N \g_stex_modules_in_file_seq
1150 \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 14.)

4.6 Symbol Declarations

```

1151 <@@=stex_symdecl>

symdecl arguments:
1152 \keys_define:nn { stex / symdecl } {
1153   name .tl_set_x:N = \l_stex_symdecl_name_str ,
1154   local .bool_set:N = \l_stex_symdecl_local_bool ,
1155   args .tl_set_x:N = \l_stex_symdecl_args_str ,
1156   type .tl_set:N = \l_stex_symdecl_type_tl
1157 }
1158
1159 \cs_new_protected:Nn \__stex_symdecl_args:n {
1160   \str_clear:N \l_stex_symdecl_name_str
1161   \str_clear:N \l_stex_symdecl_args_str
1162   \bool_set_false:N \l_stex_symdecl_local_bool
1163   \tl_clear:N \l_stex_symdecl_type_tl
1164
1165   \keys_set:nn { stex / symdecl } { #1 }
1166
1167   \exp_args:NNo \str_set:Nn \l_stex_symdecl_name_str
1168     \l_stex_symdecl_name_str
1169   \exp_args:NNo \str_set:Nn \l_stex_symdecl_args_str
1170     \l_stex_symdecl_args_str
1171 }

```

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

```

1172 \NewDocumentCommand \symdecl { 0{ } m } {
1173   \__stex_symdecl_args:n { #1 }
1174   \tl_clear:N \l_stex_symdecl_definiens_tl
1175   \stex_symdecl_do:n { #2 }
1176 }

```

(End definition for `\symdecl`. This function is documented on page 15.)

`\abbrdef`

```

1177 \NewDocumentCommand \abbrdef { 0{ } m m } {
1178   \__stex_symdecl_args:n { #1 }
1179   \tl_set:Nn \l_stex_symdecl_definiens_tl { #3 }
1180   \stex_symdecl_do:n { #2 }
1181 }

```

(End definition for \abbrdef. This function is documented on page 15.)

`\stex_symdecl_do:n`

```

1182 \cs_new_protected:Nn \stex_symdecl_do:n {
1183   \stex_if_in_module:F {
1184     % TODO throw error? some default namespace?
1185   }
1186
1187   \str_if_empty:NT \l_stex_symdecl_name_str {
1188     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1189   }
1190
1191   \prop_if_exist:cT { g_stex_symdecl_
1192     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1193     \prop_item:Nn \l_stex_current_module_prop {name} ?
1194     \l_stex_symdecl_name_str
1195     _prop
1196   }{
1197     % TODO throw error (beware of circular dependencies)
1198   }
1199
1200   \prop_clear:N \l_tmpa_prop
1201   \prop_put:Nnx \l_tmpa_prop { module } {
1202     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1203     \prop_item:Nn \l_stex_current_module_prop {name}
1204   }
1205   \seq_clear:N \l_tmpa_seq
1206   \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
1207   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1208   \prop_put:Nno \l_tmpa_prop { local } \l_stex_symdecl_local_bool
1209   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1210
1211   \exp_args:No \stex_add_constant_to_current_module:n {
1212     \l_stex_symdecl_name_str
1213   }
1214
1215   % arity/args
1216   \int_zero:N \l_tmpb_int
1217
1218   \bool_set_true:N \l_tmpa_bool
1219   \str_map_inline:Nn \l_stex_symdecl_args_str {
1220     \token_case_meaning:NnF ##1 {
1221       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1222       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1223       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1224       {\tl_to_str:n a} {
1225         \bool_set_false:N \l_tmpa_bool
1226         \int_incr:N \l_tmpb_int
1227       }
1228     }
1229     \msg_set:nnn{stex}{error/wrongargs}{
1230       args~value~in~symbol~declaration~for~
1231       \prop_item:Nn \l_stex_current_module_prop {ns} ?
1232       \prop_item:Nn \l_stex_current_module_prop {name} ?

```

```

1233     \l_stex_symdecl_name_str ~
1234     needs~to~be~
1235     i,~a~or~b,~but~##1~given
1236   }
1237   \msg_error:nn{stex}{error/wrongargs}
1238 }
1239 }
1240 \bool_if:NTF \l_tmpa_bool {
1241   % possibly numeric
1242   \str_if_empty:NTF \l_stex_symdecl_args_str {
1243     \prop_put:Nnn \l_tmpa_prop { args } {}
1244     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1245   }{
1246     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1247     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1248     \str_clear:N \l_tmpa_str
1249     \int_step_inline:nn \l_tmpa_int {
1250       \str_put_right:Nn \l_tmpa_str i
1251     }
1252     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1253   }
1254 } {
1255   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1256   \prop_put:Nnx \l_tmpa_prop { arity }
1257   { \str_count:N \l_stex_symdecl_args_str }
1258 }
1259 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
1260
1261
1262 % semantic macro
1263
1264 \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1265   \prop_item:Nn \l_tmpa_prop { module } ?
1266   \prop_item:Nn \l_tmpa_prop { name }
1267 } }
1268
1269 \bool_if:NF \l_stex_symdecl_local_bool {
1270   \exp_args:Nx \stex_add_to_current_module:n {
1271     \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1272       \prop_item:Nn \l_tmpa_prop { module } ?
1273       \prop_item:Nn \l_tmpa_prop { name }
1274     } }
1275   }
1276 }
1277
1278
1279 \stex_debug:n{New~symbol:~
1280   \prop_item:Nn \l_tmpa_prop { module } ?
1281   \prop_item:Nn \l_tmpa_prop { name } ^^J
1282   Type:~\exp_not:o { \l_stex_symdecl_type_tl } ^^J
1283   Args:~\prop_item:Nn \l_tmpa_prop { args }
1284 }
1285
1286 \prop_gset_eq:cN {

```



```

1287   g_stex_symdecl_
1288   \prop_item:Nn \l_tmpa_prop { module } ?
1289   \prop_item:Nn \l_tmpa_prop { name }
1290   _prop
1291 } \l_tmpa_prop
1292
1293 \stex_if_smsmode:TF {
1294   \bool_if:NF \l_stex_symdecl_local_bool {
1295     \exp_args:Nx \stex_addtosms:n {
1296       \prop_gset_from_keyval:cn {
1297         g_stex_symdecl_
1298         \prop_item:Nn \l_tmpa_prop { module } ?
1299         \prop_item:Nn \l_tmpa_prop { name }
1300         _prop
1301       } {
1302         name      = \prop_item:Nn \l_tmpa_prop { name }
1303         module    = \prop_item:Nn \l_tmpa_prop { module }
1304         notations = \prop_item:Nn \l_tmpa_prop { notations }
1305         local     = \prop_item:Nn \l_tmpa_prop { local }
1306         type      = \prop_item:Nn \l_tmpa_prop { type }
1307         args      = \prop_item:Nn \l_tmpa_prop { args }
1308         arity     = \prop_item:Nn \l_tmpa_prop { arity }
1309         assocs    = \prop_item:Nn \l_tmpa_prop { assocs }
1310       }
1311     }
1312   }
1313   \stex_smsmode_set_codes:
1314 }{
1315   \stex_annotate_invisible:nnn {symdecl} {
1316     \prop_item:Nn \l_tmpa_prop { module } ?
1317     \prop_item:Nn \l_tmpa_prop { name }
1318   } {
1319     \stex_annotate_invisible:nnn{type}{-}{\l_stex_symdecl_type_tl$}
1320     \stex_annotate_invisible:nnn{args}{-}{
1321       \prop_item:Nn \l_tmpa_prop { args }
1322     }
1323     \stex_annotate_invisible:nnn{macroname}{-}{#1}
1324     \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
1325       \stex_annotate_invisible:nnn{definiens}{-}{
1326         {\l_stex_symdecl_definiens_tl$}
1327       }
1328     }
1329   }
1330 }

```

(End definition for `\stex_symdecl_do:n`. This function is documented on page 15.)

`\stex_get_symbol:n`

```

1331 \str_new:N \l_stex_get_symbol_uri_str
1332
1333 \cs_new_protected:Nn \stex_get_symbol:n {
1334   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
1335     \__stex_symdecl_get_symbol_from_cs:n { #1 }
1336   }{

```

```

1337 % argument is a string
1338 % is it a command name?
1339 \cs_if_exist:cTF { #1 }{
1340   \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
1341 }{
1342   % TODO
1343   % argument is not a command name
1344 }
1345 }
1346 }
1347
1348 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
1349   \tl_set:Nx \l_tmpa_tl { #1 }
1350   \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
1351     \stex_invoke_symbol:n {
1352       \exp_args:NNx \tl_set:Nn \l_tmpa_tl
1353         { \tl_tail:N \l_tmpa_tl }
1354       \tl_if_single:NTF \l_tmpa_tl {
1355         \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
1356           \exp_after:wN \str_set:Nn \exp_after:wN
1357             \l_stex_get_symbol_uri_str \l_tmpa_tl
1358         }{
1359           % TODO
1360           % tail is not a single group
1361         }
1362       }{
1363         % TODO
1364         % tail is not a single group
1365       }
1366     }{
1367       % TODO
1368       % head is not \stex_invoke_symbol:n
1369     }
1370 }

```

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

4.7 Notations

```

1371 <@=stex_notation>
      notation arguments:
1372 \keys_define:nn { stex / notation } {
1373   lang      .tl_set_x:N = \l__stex_notation_lang_str ,
1374   variant   .tl_set_x:N = \l__stex_notation_variant_str ,
1375   prec      .tl_set_x:N = \l__stex_notation_prec_str ,
1376   unknown   .code:n      = \str_set:Nx
1377     \l__stex_notation_variant_str \l_keys_key_str
1378 }
1379
1380 \cs_new_protected:Nn \__stex_notation_args:n {
1381   \str_clear:N \l__stex_notation_lang_str
1382   \str_clear:N \l__stex_notation_variant_str
1383   \str_clear:N \l__stex_notation_prec_str
1384

```

```

1385 \keys_set:nn { stex / notation } { #1 }
1386
1387 \str_set:Nx \l__stex_notation_lang_str \l__stex_notation_lang_str
1388 \str_set:Nx \l__stex_notation_variant_str \l__stex_notation_variant_str
1389 \str_set:Nx \l__stex_notation_prec_str \l__stex_notation_prec_str
1390 }

```

\notation

```

1391 \NewDocumentCommand \notation { 0{} m } {
1392   \__stex_notation_args:n { #1 }
1393   \tl_clear:N \l_stex_symdecl_definiens_tl
1394   \stex_get_symbol:n { #2 }
1395   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
1396 }

```

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

\stex_notation_do:nn

```

1397 \cs_new_protected:Nn \stex_notation_do:nn {
1398   \prop_set_eq:Nc \l_tmpa_prop {
1399     g_stex_symdecl_ #1 _prop
1400   }
1401
1402   \prop_clear:N \l_tmpb_prop
1403   \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
1404   \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
1405   \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
1406
1407   % precedences
1408   \seq_clear:N \l_tmpb_seq
1409   \exp_args:NNno
1410   \str_if_empty:NTF \l__stex_notation_prec_str {
1411     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1412     \int_compare:nNnTF \l_tmpa_str = 0 {
1413       \exp_args:NNnx
1414       \prop_put:Nnn \l_tmpb_prop { opprec }
1415       { \int_use:N \infprec }
1416     }{
1417       \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
1418     }
1419   } {
1420     \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
1421     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
1422       \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
1423       \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
1424         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
1425         \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
1426         \seq_map_inline:Nn \l_tmpa_seq {
1427           \seq_put_right:Nn \l_tmpb_seq { ##1 }
1428         }
1429       }
1430       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1431     }{
1432       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1433       \int_compare:nNnTF \l_tmpa_str = 0 {

```

```

1434         \exp_args:NNnx
1435         \prop_put:Nnn \l_tmpb_prop { opprec }
1436         { \int_use:N \infprec }
1437     }{
1438         \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
1439     }
1440 }
1441 }
1442
1443 \seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
1444 \int_step_inline:nn { \l_tmpa_str } {
1445     \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
1446         \exp_args:NNx
1447         \seq_put_right:Nn \l_tmpb_seq {
1448             \prop_item:Nn \l_tmpb_prop { opprec }
1449         }
1450     }
1451 }
1452
1453 \prop_put:Nno \l_tmpb_prop { argprec } \l_tmpb_seq
1454
1455 \int_compare:nNnTF \l_tmpa_str = 0 {
1456     \cs_set:Npx \l__stex_notation_macrocode_cs {} {
1457         \stex_term_oms:nnnn { #1 }
1458         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1459         { \prop_item:Nn \l_tmpb_prop { opprec } }
1460         { #2 }
1461     }
1462     \__stex_notation_final:
1463 }{
1464     \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
1465     \str_if_in:NnTF \l_tmpb_str b {
1466         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1467         \cs_set:Npx \l_tmpa_str {
1468             \stex_term_omb:nnnn { #1 }
1469             { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1470             { \prop_item:Nn \l_tmpb_prop { opprec } }
1471             { #2 }
1472         }
1473     }{
1474         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1475         \cs_set:Npx \l_tmpa_str {
1476             \stex_term_oma:nnnn { #1 }
1477             { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1478             { \prop_item:Nn \l_tmpb_prop { opprec } }
1479             { #2 }
1480         }
1481     }
1482
1483     \int_zero:N \l_tmpa_int
1484     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
1485     \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
1486     \tl_clear:N \l_tmpa_tl
1487     \__stex_notation_arguments:

```

```

1488 }
1489 }

```

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

`__stex_notation_arguments:` Takes care of annotating the arguments in a notation macro

```

1490 \cs_new_protected:Nn \__stex_notation_arguments: {
1491   \int_incr:N \l_tmpa_int
1492   \str_if_empty:NTF \l_tmpa_str {
1493     \__stex_notation_final:
1494   }{
1495     \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
1496     \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
1497     \str_if_eq:VnTF \l_tmpb_str a {
1498       \__stex_notation_argument_assoc:n
1499     }{
1500       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1501       \tl_put_right:Nx \l_tmpa_tl {
1502         { \stex_term_arg:nnn
1503           { \int_use:N \l_tmpa_int }
1504           { \l_tmpb_str }
1505           { ####\int_use:N \l_tmpa_int }
1506         }
1507       }
1508       \__stex_notation_arguments:
1509     }
1510   }
1511 }

```

(End definition for `__stex_notation_arguments:.`)

`__stex_notation_argument_assoc:n`

```

1512 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
1513   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1514   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
1515   \tl_put_right:Nx \l_tmpa_tl {
1516     { \stex_term_assoc_arg:nnnn
1517       { \int_use:N \l_tmpa_int }
1518       { \l_tmpb_str }
1519       { \l_tmpa_cs {#####1} {#####2} }
1520       { ####\int_use:N \l_tmpa_int }
1521     }
1522   }
1523   \__stex_notation_arguments:
1524 }

```

(End definition for `__stex_notation_argument_assoc:n`.)

`__stex_notation_final:` Called after processing all notation arguments

```

1525 \cs_new_protected:Nn \__stex_notation_final: {
1526   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
1527   \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
1528   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
1529   \cs_generate_from_arg_count:cNnn {

```

```

1530     stex_notation_ \l_tmpa_str \c_hash_str
1531     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1532     _cs
1533 }
1534 \cs_set:Npx \l_tmpb_str {
1535     \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl
1536 }
1537
1538 \stex_debug:n{
1539     Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1540     ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
1541     Operator~precedence:~
1542     \prop_item:Nn \l_tmpb_prop { opprec }^^J
1543     Argument~precedences:~
1544     \seq_use:Nn \l_tmpa_seq {,~}^^J
1545     Notation: \cs_meaning:c {
1546         stex_notation_ \l_tmpa_str \c_hash_str
1547         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1548         _cs
1549     }
1550 }
1551
1552 \prop_gset_eq:cN {
1553     g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
1554     \c_hash_str \l__stex_notation_lang_str _prop
1555 } \l_tmpb_prop
1556
1557 \exp_args:Nx
1558 \stex_add_to_current_module:n {
1559     \prop_get:cnN {
1560         g_stex_symdecl_
1561         \prop_item:Nn \l_tmpb_prop { symbol }
1562         _prop
1563     } { notations } \exp_not:N \l_tmpa_seq
1564     \seq_put_right:Nn \exp_not:N \l_tmpa_seq {
1565         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1566     }
1567     \prop_put:cno {
1568         g_stex_symdecl_
1569         \prop_item:Nn \l_tmpb_prop { symbol }
1570         _prop
1571     } { notations } \exp_not:N \l_tmpa_seq
1572 }
1573
1574 \stex_if_smsmode:TF {
1575     \stex_smsmode_set_codes:
1576     \exp_args:Nx \stex_addtosms:n {
1577         \prop_gset_from_keyval:cn {
1578             g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
1579             \c_hash_str \l__stex_notation_lang_str _prop
1580         } {
1581             symbol = \prop_item:Nn \l_tmpb_prop { symbol }
1582             language = \prop_item:Nn \l_tmpb_prop { language }
1583             variant = \prop_item:Nn \l_tmpb_prop { variant }

```

```

1584         opprec      = \prop_item:Nn \l_tmpb_prop { opprec }
1585         argprec     = \prop_item:Nn \l_tmpb_prop { argprec }
1586     }
1587 }
1588 ){
1589     \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
1590     \seq_put_right:Nx \l_tmpa_seq {
1591         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1592     }
1593     \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
1594     \prop_set_eq:cN {
1595         g_stex_symdecl_ \l_tmpa_str _prop
1596     } \l_tmpa_prop
1597
1598     % HTML annotations
1599     \stex_annotate_invisible:nnn { notation }
1600     { \prop_item:Nn \l_tmpb_prop { symbol } } {
1601         \stex_annotate_invisible:nnn { notationfragment }
1602         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
1603     }
1604     \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
1605     \stex_annotate_invisible:nnn { precedence }
1606     { \prop_item:Nn \l_tmpb_prop { opprec };
1607       \seq_use:Nn \l_tmpa_seq { x }
1608     }{}
1609
1610     \int_zero:N \l_tmpa_int
1611     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
1612     \tl_clear:N \l_tmpa_tl
1613     \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
1614         \int_incr:N \l_tmpa_int
1615         \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
1616         \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
1617         \str_if_eq:VnTF \l_tmpb_str a {
1618             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
1619                 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
1620                 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
1621             } }
1622         }{
1623             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
1624                 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
1625             } }
1626         }
1627     }
1628     \stex_annotate_invisible:nnn { notationcomp }{}{
1629         $ \exp_args:Nno \use:nn { \use:c {
1630             stex_notation_ \prop_item:Nn \l_tmpb_prop { symbol }
1631             \c_hash_str \l__stex_notation_variant_str
1632             \c_hash_str \l__stex_notation_lang_str _cs
1633         } } { \l_tmpa_tl } $
1634     }
1635 }
1636 }

```

(End definition for __stex_notation_final:.)

\symdef

```
1637 \keys_define:nn { stex / symdef } {
1638   name .tl_set_x:N = \l_stex_symdecl_name_str ,
1639   local .bool_set:N = \l_stex_symdecl_local_bool ,
1640   args .tl_set_x:N = \l_stex_symdecl_args_str ,
1641   type .tl_set:N = \l_stex_symdecl_type_tl ,
1642   lang .tl_set_x:N = \l__stex_notation_lang_str ,
1643   variant .tl_set_x:N = \l__stex_notation_variant_str ,
1644   prec .tl_set_x:N = \l__stex_notation_prec_str ,
1645   unknown .code:n = \str_set:Nx
1646     \l__stex_notation_variant_str \l_keys_key_str
1647 }
1648
1649 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
1650   \str_clear:N \l_stex_symdecl_name_str
1651   \str_clear:N \l_stex_symdecl_args_str
1652   \bool_set_false:N \l_stex_symdecl_local_bool
1653   \tl_clear:N \l_stex_symdecl_type_tl
1654   \str_clear:N \l__stex_notation_lang_str
1655   \str_clear:N \l__stex_notation_variant_str
1656   \str_clear:N \l__stex_notation_prec_str
1657
1658   \keys_set:nn { stex / symdef } { #1 }
1659
1660   \exp_args:NNo \str_set:Nn \l_stex_symdecl_name_str
1661     \l_stex_symdecl_name_str
1662   \exp_args:NNo \str_set:Nn \l_stex_symdecl_args_str
1663     \l_stex_symdecl_args_str
1664   \exp_args:NNo \str_set:Nn \l__stex_notation_lang_str
1665     \l__stex_notation_lang_str
1666   \exp_args:NNo \str_set:Nn \l__stex_notation_variant_str
1667     \l__stex_notation_variant_str
1668   \exp_args:NNo \str_set:Nn \l__stex_notation_prec_str
1669     \l__stex_notation_prec_str
1670 }
1671
1672 \NewDocumentCommand \symdef { 0{} m } {
1673   \__stex_notation_symdef_args:n { #1 }
1674   \tl_clear:N \l_stex_symdecl_definiens_tl
1675   \stex_symdecl_do:n { #2 }
1676   \exp_args:Nx \stex_notation_do:nn {
1677     \prop_item:Nn \l_tmpa_prop { module } ?
1678     \prop_item:Nn \l_tmpa_prop { name }
1679   }
1680 }
```

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

\stex_invoke_symbol:n Invokes a semantic macro

```
1681 \cs_new_protected:Nn \stex_invoke_symbol:n {
1682   \peek_charcode_remove:NTF ! {
1683     \stex_term_custom:nn { #1 } { }
1684   } {
1685     \if_mode_math:
```



```

1686     \exp_after:wN \_\stex_notation_invoke_math:n
1687   \else:
1688     \exp_after:wN \_\stex_notation_invoke_text:n
1689     \fi: { #1 }
1690   }
1691 }

```

(End definition for \stex_invoke_symbol:n. This function is documented on page 16.)

_\stex_notation_invoke_math:n

```

1692 \cs_new_protected:Nn \_\stex_notation_invoke_math:n {
1693   \peek_charcode_remove:NTF * {
1694     \_\stex_notation_invoke_text:n { #1 }
1695   }{
1696     \peek_charcode:NTF [ {
1697       \_\stex_notation_invoke_math:nw { #1 }
1698     }{
1699       \_\stex_notation_invoke_math:nw { #1 } []
1700     }
1701   }
1702 }

```

(End definition for _\stex_notation_invoke_math:n.)

_\stex_notation_invoke_math:nw

```

1703 \cs_new_protected:Npn \_\stex_notation_invoke_math:nw #1 [#2] {
1704   \_\stex_notation_args:n { #2 }
1705   \prop_set_eq:Nc \l_tmpa_prop {
1706     g_stex_symdecl_ #1 _prop
1707   }
1708   \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
1709   \seq_if_empty:NTF \l_tmpa_seq {
1710     \msg_set:nnn{stex}{error/nonotations}{
1711       Symbol~#1~used,~but~has~no~notations!
1712     }
1713     \msg_error:nn{stex}{error/nonotations}
1714   } {
1715     \seq_if_in:NxTF \l_tmpa_seq
1716       { \l_\stex_notation_variant_str \c_hash_str \l_\stex_notation_lang_str }{
1717       \use:c{
1718         stex_notation_ #1 \c_hash_str
1719         \l_\stex_notation_variant_str \c_hash_str \l_\stex_notation_lang_str
1720         _cs
1721       }
1722     }{
1723       \str_if_empty:NTF \l_\stex_notation_variant_str {
1724         \str_if_empty:NTF \l_\stex_notation_lang_str {
1725           \seq_get_left:NN \l_tmpa_seq \l_tmpa_str
1726           \use:c{
1727             stex_notation_ #1 \c_hash_str \l_tmpa_str
1728             _cs
1729           }
1730         }{
1731           \msg_set:nnn{stex}{error/wrongnotation}{
1732             Symbol~#1~has~no~notation~

```

```

1733         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1734     }
1735     \msg_error:nn{stex}{error/wrongnotation}
1736 }
1737 }{
1738     \msg_set:nnn{stex}{error/wrongnotation}{
1739         Symbol~#1~has~no~notation~
1740         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1741     }
1742     \msg_error:nn{stex}{error/wrongnotation}
1743 }
1744 }
1745 }
1746 }

```

(End definition for `__stex_notation_invoke_math:nw`.)

`_stex_notation_invoke_text:n`

```

1747 \cs_new_protected:Nn \__stex_notation_invoke_text:n {
1748     \prop_set_eq:Nc \l_tmpa_prop {
1749         g_stex_symdecl_ #1 _prop
1750     }
1751     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
1752     \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
1753 }

```

(End definition for `_stex_notation_invoke_text:n`.)

4.8 Terms

1754 $\langle @@ = \text{stex_term} \rangle$

Precedences:

```

\infprec
\neginfprec
1755 \int_const:Nn \infprec {\c_max_int}
1756 \int_const:Nn \neginfprec {-\c_max_int}
1757 \int_new:N \l__stex_term_downprec
1758 \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 17.)

Bracketing:

```

\l__stex_term_left_bracket_str
\l__stex_term_right_bracket_str
1759 \tl_set:Nn \l__stex_term_left_bracket_str (
1760 \tl_set:Nn \l__stex_term_right_bracket_str )
1761 \RequirePackage{scalerel}

```

(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

```

1762 \cs_new_protected:Nn \__stex_term_maybe_brackets:nn {
1763     \int_compare:nNnTF { #1 } < \l__stex_term_downprec {
1764         \STEXdobrackets { #2 }
1765     }{ #2 }
1766 }

```

(End definition for `_stex_term_maybe_brackets:nn`.)

`\STEXdobrackets`

```

1767 \cs_new_protected:Npn \STEXdobrackets #1 {
1768   \ThisStyle{\if D\m@switch
1769     \exp_args:Nnx \use:nn
1770     { \left\l__stex_term_left_bracket_str #1 }
1771     { \right\l__stex_term_right_bracket_str }
1772   \else
1773     \exp_args:Nnx \use:nn
1774     { \l__stex_term_left_bracket_str #1 }
1775     { \l__stex_term_right_bracket_str }
1776   \fi}
1777 }
```

(End definition for `\STEXdobrackets`. This function is documented on page 18.)

`\STEXwithbrackets`

```

1778 \cs_new_protected:Npn \STEXwithbrackets #1 #2 #3 {
1779   \exp_args:Nnx \use:nn
1780   {
1781     \tl_set:Nx \l__stex_term_left_bracket_str { #1 }
1782     \tl_set:Nx \l__stex_term_right_bracket_str { #2 }
1783     #3
1784   }
1785   {
1786     \tl_set:Nn \exp_not:N \l__stex_term_left_bracket_str
1787     { \l__stex_term_left_bracket_str }
1788     \tl_set:Nn \exp_not:N \l__stex_term_right_bracket_str
1789     { \l__stex_term_right_bracket_str }
1790   }
1791 }
```

(End definition for `\STEXwithbrackets`. This function is documented on page 18.)

OMDOC terms:

`\stex_term_oms:nnnn`

```

1792 \cs_new_protected:Nn \stex_term_oms:nnnn {
1793   \_stex_term_maybe_brackets:nn { #3 }{
1794     \stex_annotate:nnn{OMID}{#1\c_hash_str#2}{#4}
1795   }
1796 }
```

(End definition for `\stex_term_oms:nnnn`. This function is documented on page 17.)

`\stex_term_oma:nnnn`

```

1797 \cs_new_protected:Nn \stex_term_oma:nnnn {
1798   \_stex_term_maybe_brackets:nn { #3 }{
1799     \stex_annotate:nnn{OMA}{#1\c_hash_str#2}{#4}
1800   }
1801 }
```

(End definition for `\stex_term_oma:nnnn`. This function is documented on page 17.)

\stex_term_omb:nnnn

```

1802 \cs_new_protected:Nn \stex_term_omb:nnnn {
1803   \__stex_term_maybe_brackets:nn { #3 }{
1804     \stex_annotate:nnn{OMBIND}{#1}\c_hash_str#2}{#4}
1805   }
1806 }

```

(End definition for \stex_term_omb:nnnn. This function is documented on page 17.)

\stex_term_arg:nnn

```

1807 \cs_new_protected:Nn \stex_term_arg:nnn {
1808   \exp_args:Nnx \use:nn
1809   { \int_set:Nn \l__stex_term_downprec { #2 }
1810     \stex_annotate:nnn{arg}{#1}{#3} }
1811   { \int_set:Nn \l__stex_term_downprec { \int_use:N \l__stex_term_downprec } }
1812 }

```

(End definition for \stex_term_arg:nnn. This function is documented on page 17.)

\stex_term_assoc_arg:nnnn

```

1813 \cs_new_protected:Nn \stex_term_assoc_arg:nnnn {
1814   \seq_set_split:Nnn \l_tmpa_seq , { #4 }
1815   \int_compare:nNnTF { \seq_count:N \l_tmpa_seq } < 2 {
1816     \tl_set:Nn \l_tmpa_tl { #4 }
1817   }{
1818     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
1819     \seq_reverse:N \l_tmpa_seq
1820     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_tl
1821     \tl_set:No \l_tmpa_tl { \l_tmpb_tl }
1822     \seq_map_inline:Nn \l_tmpa_seq {
1823       \tl_set:Nx \l_tmpa_tl {
1824         \exp_args:Nno
1825         \l_tmpa_cs { ##1 } { \l_tmpa_tl }
1826       }
1827     }
1828   }
1829   \exp_args:Nnno
1830   \stex_term_arg:nnn{#1}{#2}{ \l_tmpa_tl }
1831 }

```

(End definition for \stex_term_assoc_arg:nnnn. This function is documented on page 17.)

\stex_term_custom:nn

```

1832 \cs_new_protected:Nn \stex_term_custom:nn {
1833   \str_set:Nn \l__stex_term_custom_uri { #1 }
1834   \str_set:Nn \l_tmpa_str { #2 }
1835   \tl_clear:N \l_tmpa_tl
1836   \int_zero:N \l_tmpa_int
1837   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
1838   \__stex_term_custom_loop:
1839 }

```

(End definition for \stex_term_custom:nn. This function is documented on page 18.)

_stex_term_custom_loop:

```

1840 \cs_new_protected:Nn \_stex_term_custom_loop: {
1841   \bool_set_false:N \l_tmpa_bool
1842   \bool_while_do:nn {
1843     \str_if_eq_p:ee X {
1844       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
1845     }
1846   }{
1847     \int_incr:N \l_tmpa_int
1848   }
1849
1850   \peek_charcode:NTF [ {
1851     % notation/text component
1852     \_stex_term_custom_component:w
1853   } {
1854     \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
1855       % all arguments read => finish
1856       \_stex_term_custom_final:
1857     } {
1858       % arguments missing
1859       \peek_charcode_remove:NTF * {
1860         % invisible, specific argument position or both
1861         \peek_charcode:NTF [ {
1862           % visible specific argument position
1863           \_stex_term_custom_arg:wn
1864         } {
1865           % invisible
1866           \peek_charcode_remove:NTF * {
1867             % invisible specific argument position
1868             \_stex_term_custom_arg_inv:wn
1869           } {
1870             % invisible next argument
1871             \_stex_term_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
1872           }
1873         }
1874       } {
1875         % next normal argument
1876         \_stex_term_custom_arg:wn [ \l_tmpa_int + 1 ]
1877       }
1878     }
1879   }
1880 }

```

(End definition for _stex_term_custom_loop:.)

_stex_term_custom_arg_inv:wn

```

1881 \cs_new_protected:Npn \_stex_term_custom_arg_inv:wn [ #1 ] #2 {
1882   \bool_set_true:N \l_tmpa_bool
1883   \_stex_term_custom_arg:wn [ #1 ] { #2 }
1884 }

```

(End definition for _stex_term_custom_arg_inv:wn.)

_stex_term_custom_arg:wn

```

1885 \cs_new_protected:Npn \__stex_term_custom_arg:wn [ #1 ] #2 {
1886   \str_set:Nx \l_tmpb_str {
1887     \str_item:Nn \l_tmpa_str { #1 }
1888   }
1889   \str_case:NnTF \l_tmpb_str {
1890     { X } { } % TODO throw error
1891     { i } { \__stex_term_custom_set_X:n { #1 } }
1892     { b } { \__stex_term_custom_set_X:n { #1 } }
1893     { a } { } % TODO ?
1894   }{}{
1895     % TODO throw error
1896   }
1897
1898   \bool_if:nTF \l_tmpa_bool {
1899     \tl_put_right:Nx \l_tmpa_tl {
1900       \stex_annotate_invisible:nnn { arg }{ \int_eval:n { #1 } }
1901       \exp_not:n { { #2 } }
1902     }
1903   } {
1904     \tl_put_right:Nx \l_tmpa_tl {
1905       \stex_annotate:nnn { arg }{ \int_eval:n { #1 } }
1906       \exp_not:n { { #2 } }
1907     }
1908   }
1909
1910   \__stex_term_custom_loop:
1911 }

```

(End definition for __stex_term_custom_arg:wn.)

__stex_term_custom_set_X:n

```

1912 \cs_new_protected:Nn \__stex_term_custom_set_X:n {
1913   \str_set:Nx \l_tmpa_str {
1914     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
1915     X
1916     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
1917   }
1918 }

```

(End definition for __stex_term_custom_set_X:n.)

__stex_term_custom_component:

```

1919 \cs_new_protected:Npn \__stex_term_custom_component:w [ #1 ] {
1920   \tl_put_right:Nn \l_tmpa_tl { #1 }
1921   \__stex_term_custom_loop:
1922 }

```

(End definition for __stex_term_custom_component:.)

__stex_term_custom_final:

```

1923 \cs_new_protected:Nn \__stex_term_custom_final: {
1924   \int_compare:nNnTF \l_tmpb_int = 0 {
1925     \exp_args:Nnno \stex_annotate:nnn{OMID}
1926   }{
1927     \str_if_in:NnTF \l_tmpa_str {b} {

```

```

1928     \exp_args:Nnno \stex_annotate:nnn{OMBIND}
1929   } {
1930     \exp_args:Nnno \stex_annotate:nnn{OMA}
1931   }
1932 }
1933 { \l__stex_term_custom_uri } { \l_tmpa_tl }
1934 }

```

(End definition for _stex_term_custom_final:.)