

stex.sty: \TeX 2.0*

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

2021-11-24

Abstract

TODO

1 Introduction

TODO

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

Contents

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

2 Manual

2.1 Modules

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

2.2 Semantic Macros and Notations

Semantic macros invoke a formally declared symbol.

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

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

Example 1

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

(ab)

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

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

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

Example 2

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

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

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

¹EdNOTE: TODO

Outside of math mode, or by using the starred variant `\foo*`, allows to provide a custom notation, where notational (or textual) components can be given explicitly in square brackets.

Example 3

```
$\mult*{a}[\comp{\ast}]{b}$ is the
\mult[\comp{product of}]{a}[\comp{and}]{b}
```

$a*b$ is the *product of a and b*

In custom mode, prefixing an argument with a star will not print that argument, but still export it to OMDoc:

Example 4

```
\mult[\comp{Multiplying}]*{$\mult{a}{b}$} again by {$b$} yields...
```

Multiplying again by b yields...

The syntax `*[int]` allows switching the order of arguments. For example, given a 2-ary semantic macro `\forevery` with exemplary notation `\forall #1. #2`, we can write

Example 5

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

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

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

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

Example 6

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

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

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

Example 7

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

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

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

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

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

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

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

2.2.1 Other Argument Types

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

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

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

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

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

Example 8

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

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

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

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

Example 9

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

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

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

² ³

2.2.2 Precedences

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

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

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

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

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

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

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

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

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

For example:

Example 10

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

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

2.3 Archives and Imports

2.3.1 Namespaces

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

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

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

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

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

2.3.2 Paths in Import-Statements

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

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

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

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

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

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

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

3 Documentation

3.1 Utils

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

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

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

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

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

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

| | |
|-----------------------------|--|
| <code>\if@latexml</code> | LaTeX2e and LaTeX3 conditionals for LaTeXML. |
| <code>\latexml_if_p:</code> | |
| <code>\latexml_if:T</code> | |
| <code>\latexml_if:F</code> | |
| <code>\latexml_if:TF</code> | |

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

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

Annotates the HTML generated by `⟨content⟩` with

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

3.1.2 Languages

| |
|--|
| <code>\c_stex_languages_prop</code> |
| <code>\c_stex_language_abbrevs_prop</code> |

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

3.2 Files, Paths, URIs

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

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

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

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

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

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

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

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

`\g_stex_currentfile_seq`

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

Test 1

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

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

3.3 MathHub Archives

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

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

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

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

`\l_stex_current_repository_prop`

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

`\stex_set_current_repository:n`

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

`\stex_require_repository:n`

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

`\libinput`

`\libinput{<filename>}`

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

Test 2

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

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

3.4 The Module System

`\l_stex_current_module_prop`

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

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

Additionally, it stores:

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

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

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

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

`\stex_add_to_current_module:n`
`\STEXexport`

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

`\stex_add_constant_to_current_module:n`

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

`\stex_add_import_to_current_module:n`

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

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

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

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

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

Computes the current namespace

Test 3

```

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

```

```

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

```

3.4.1 The module-environment

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

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

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

Test 4

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

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

Test 5

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

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

\l_stex_all_modules_seq

Stores full URIs for all modules currently in scope.

\STEXModule

\STEXModule {*fragment*}

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

`\stex_invoke_module:n`

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

Test 6

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

Module 3.2[STEXModuleTest1]

Module 3.3[STEXModuleTest2]

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

3.4.2 SMS Mode

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

`\g_stex_smsmode_allowedmacros_tl`

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

`\g_stex_smsmode_allowedmacros_escape_tl`

Macros that are executed with the category codes restored.

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

`\g_stex_smsmode_allowedenvs_seq`

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

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

Tests whether SMS mode is currently active.

`\stex_smsmode_set_codes:`

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

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

`\stex_in_smsmode:nn`

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

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

Test 7

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

3.4.3 Imports and Inheritance

`\importmodule`

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

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

Test 8

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

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

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

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

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

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

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

Test 9

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

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

Module 3.8[UseTest1]

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

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

Test 10

Circular dependencies:

```

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

```

Circular dependencies:

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

| | |
|---|--|
| <hr/> <hr/> <code>\stex_import_module_uri:nn</code> | <code>\stex_import_module_uri:nn {<archive-ID>} {<module-path>}</code> Determines the URI of a module by splitting <code><module-path></code> into <code><path>?<name></code> . If <code><module-path></code> does <i>not</i> contain a <code>?</code> -character, we consider it to be the <code><name></code> , and <code><path></code> to be empty. If <code><archive-ID></code> is empty, it is automatically set to the ID of the current archive (if one exists). |
| | <ol style="list-style-type: none"> 1. If <code><archive-ID></code> is empty: <ol style="list-style-type: none"> (a) If <code><path></code> is empty, then <code><name></code> must have been declared earlier in the same file and retrievable from <code>\g_stex_modules_in_file_seq</code>, or a file with name <code><name>.<lang>.tex</code> must exist in the same folder, containing a module <code><name></code>. That module should have the same namespace as the current one. (b) If <code><path></code> is not empty, it must point to the relative path of the containing file as well as the namespace. 2. Otherwise: <ol style="list-style-type: none"> (a) If <code><path></code> is empty, then <code><name></code> must have been declared earlier in the same file and retrievable from <code>\g_stex_modules_in_file_seq</code>, or a file with name <code><name>.<lang>.tex</code> must exist in the top <code>source</code> folder of the archive, containing a module <code><name></code>. That module should lie directly in the namespace of the archive. (b) If <code><path></code> is not empty, it must point to the path of the containing file as well as the namespace, relative to the namespace of the archive. If a module by that namespace exists, it is returned. Otherwise, we call <code>\stex_require_module:nn</code> on the <code>source</code> directory of the archive to find the file. |

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

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

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

| | |
|--|--|
| <code>\g_stex_module_files_prop</code> | |
| <code>\g_stex_modules_in_file_seq</code> | |

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

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

3.5 Symbols and Terms

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

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

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

`\stex_symdecl_do:n`

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

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

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

Test 11

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

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

`\l_stex_all_symbols_seq`

Stores full URIs for all modules currently in scope.

`\stex_get_symbol:n`

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

`\STEXsymbol`

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

`\symref`

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

`\stex_invoke_symbol:n`

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

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

`\notation`

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

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

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

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

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

Test 12

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

Module 3.13[NotationTest]

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

Combines $\backslash\text{symdecl}$ and $\backslash\text{notation}$ by introducing a new symbol and assigning a new notation for it.

Test 13

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

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

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

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

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

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

Test 16

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

| |
|--|
| <div> <div>Module 3.17[TextTest]</div> <div> some a and some b and also some c here. some <i>a</i> and some <i>b</i> and also some <i>c</i> here. or just some <i>c</i> bar or first <i>b</i>, then <i>c</i>, and finally <i>a</i> </div> </div> |
|--|

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

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

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

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

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

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

3.6 Structural Features

| | |
|------------------------|--|
| <code>symboldoc</code> | <pre>\begin{<symboldoc>}{<symbols>} <text> \end{<symboldoc>}</pre> <p>Declares <i><text></i> to be a (natural language, encyclopaedic) description of <i>{<symbols>}</i> (a comma separated list of symbol identifiers).</p> |
|------------------------|--|

3.6.1 Structures

| | |
|------------------------|------|
| <code>structure</code> | TODO |
|------------------------|------|

Test 17

```

\begin{module}{StructureTest1}
\begin{structure}[name=Magma]{magma}
\symdef{universe}{{\comp M}}
\symdef[ args=2]{op}{#1 \comp\circ #2}
$ \isa{\op ab} \universe$
\end{structure}

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

\present\magma

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

Test: $\mM{op}ab$

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

```

Module 3.18[StructureTest1]

```

(aob:(M))
file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?StructureTest1/Magma-feature?universe,file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?StructureTest1/Magma-feature?op
>macro->stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?StructureTest1?Magma}
Test: (a+b)
Test2: ((U,+))

```

4 Implementation

4.1 The `sTeX` document class

```

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

```

```

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

```

4.2 Preliminaries

```

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

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

```

\sTeX The sTeX logo:

```

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

```

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

Messages

```

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

```

\stex_debug:n Debug mode

```

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

```

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

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

(End definition for `\c__stex_sms_iow`.)

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

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

4.2.1 L^AT_EX_ML and S^CA_LT_EX

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

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

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

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

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

4.2.2 HTML Annotations

77 `<@@=stex_annotate>`

`\l__stex_annotate_arg_tl`
`\c__stex_annotate_emptyarg_tl`

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

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

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

`__stex_annotate_checkempty:n`

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

(End definition for `__stex_annotate_checkempty:n`.)

`\stex_annotate:enw`

`\stex_annotate_invisible:n`

`\stex_annotate_invisible:nnn`

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

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

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

```

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

```

```

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

```

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

4.2.3 Languages

```

173 <@@=stex_language>

```

`\c_stex_languages_prop`

`\c_stex_language_abbrevs_prop`

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

```

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

```

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

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

```

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

```

4.3 Files, Paths and URIs

214 `<@@=stex_path>`

4.3.1 Generic Path Handling

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

```

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

```

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

```

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

```

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

```

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

```

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

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

```

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

```

```

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

```

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

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

```

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

```

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

4.3.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

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

```



```

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

```

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

We determine the PWD

`\c_stex_pwd_seq`
`\c_stex_pwd_str`

```

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

```

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

4.3.3 File Hooks and Tracking

```

305 <@=stex_files>

```

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

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

`\g__stex_files_stack` keeps track of file changes

```

306 \seq_gclear_new:N\g__stex_files_stack

```

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`

```

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

```

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

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

```

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

```

```

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

```

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

4.4 MathHub Repositories

```

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

```

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

```

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

```

```

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

```

(End definition for __stex_mathhub_do_manifest:n.)

\l_stex_mathhub_manifest_file_seq

```

373 \str_new:N\l__stex_mathhub_manifest_file_seq

```

(End definition for \l_stex_mathhub_manifest_file_seq.)

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

```

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

```

(End definition for `_stex_mathhub_find_manifest:N`.)

`\c_stex_mathhub_manifest_ior` File variable used for MANIFEST-files

409 `\ior_new:N \c__stex_mathhub_manifest_ior`

(End definition for `\c_stex_mathhub_manifest_ior`.)

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

```

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

```

(End definition for `_stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

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

```

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

`\stex_require_repository:n`

```

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

```

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

`\l_stex_current_repository_prop`

Current MathHub repository

```

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

```

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

`\inputref`

```

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

```

```

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

```

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

`\mhp`

```

508 \def \mhp #1 #2 {
509   \str_if_eq:nnTF{#1}{#2}{
510     \c_stex_mathhub_str /
511     \prop_item:Nn \l_stex_current_repository_prop { id }
512     / source / #2
513   }{
514     \c_stex_mathhub_str / #1 / source / #2
515   }
516 }

```

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

`\libinput`

```

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

```

```

543 \bool_set_true:N \l_tmpa_bool
544 \tl_put_right:Nx \l_tmpa_tl {
545   \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
546     / \l_tmpa_str / lib / #1.tex}
547 }
548 }{}
549 \bool_if:NF \l_tmpa_bool {
550   \msg_set:nnn{stex}{error/nofile}{
551     \c_backslash_str libinput~no~file~#1.tex~found!
552   }
553   \msg_error:nn{stex}{error/nofile}
554 }
555 \scalatexBREAK
556 \l_tmpa_tl
557 }

```

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

4.5 Module System

```

558 <@@=stex_module>

```

`\l_stex_current_module_prop`

```

559 \prop_new:N \l_stex_current_module_prop

```

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

`stex_if_in_module_p:`

`stex_if_in_module:TF`

```

560 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
561   \prop_if_empty:NTF \l_stex_current_module_prop
562   \prg_return_false: \prg_return_true:
563 }

```

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

`stex_if_module_exists_p:n`

`stex_if_module_exists:nTF`

```

564 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
565   \prop_if_exist:cTF { c_stex_module_#1_prop }
566   \prg_return_true: \prg_return_false:
567 }

```

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

`\stex_add_to_current_module:n`

`\STEXexport`

```

568 \cs_new_protected:Nn \stex_add_to_current_module:n {
569   \prop_get:NnN \l_stex_current_module_prop { content } \l_tmpa_tl
570   \tl_put_right:Nn \l_tmpa_tl { #1 }
571   \prop_put:Nno \l_stex_current_module_prop { content } { \l_tmpa_tl }
572 }
573 \NewDocumentCommand \STEXexport { m }{
574   \stex_smsmode_set_codes:
575   \stex_add_to_current_module:n { #1 }
576   #1
577 }

```

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

`\stex_add_constant_to_current_module:n`

```

578 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
579   \str_set:Nx \l_tmpa_str { #1 }
580   \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
581   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
582   \prop_put:Nno \l_stex_current_module_prop { constants } \l_tmpa_seq
583 }

```

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

`\stex_add_import_to_current_module:n`

```

584 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
585   \str_set:Nx \l_tmpa_str { #1 }
586   \prop_get:NnN \l_stex_current_module_prop { imports } \l_tmpa_seq
587   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
588   \prop_put:Nno \l_stex_current_module_prop { imports } \l_tmpa_seq
589 }

```

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

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

`\l_stex_modules_ns_str`

```

590 \str_new:N \l_stex_modules_ns_str

591 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
592   \str_set:Nx \l_tmpa_str { #1 }
593   \seq_set_eq:NN \l_tmpa_seq #2
594   % split off file extension
595   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
596   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
597   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
598   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
599
600   \bool_set_true:N \l_tmpa_bool
601   \bool_while_do:Nn \l_tmpa_bool {
602     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
603     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
604       {source} { \bool_set_false:N \l_tmpa_bool }
605     }{}{
606       \seq_if_empty:NT \l_tmpa_seq {
607         \bool_set_false:N \l_tmpa_bool
608       }
609     }
610   }

611
612   \seq_if_empty:NTF \l_tmpa_seq {
613     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
614   }{
615     \str_set:Nx \l_stex_modules_ns_str {
616       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq

```



```

617     }
618   }
619 }

```

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

`\stex_modules_current_namespace:`

```

620 \cs_new_protected:Nn \stex_modules_current_namespace: {
621   \prop_get:NnNTF \l_stex_current_repository_prop { ns } \l_tmpa_str {
622     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
623   }{
624     % split off file extension
625     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
626     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
627     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
628     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
629     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
630     \str_set:Nx \l_stex_modules_ns_str {
631       file:/\stex_path_to_string:N \l_tmpa_seq
632     }
633   }
634 }

```

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

4.5.1 The module environment

`\l_stex_all_modules_seq` Stores all available modules

```

635 \seq_new:N \l_stex_all_modules_seq

```

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

`\STEXModule`

`\stex_invoke_module:n`

```

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

```

```

657 \l_tmpa_tl
658 }
659
660 \cs_new_protected:Nn \stex_invoke_module:n {
661   \stex_debug:n{Invoking~module~#1}
662   \peek_charcode_remove:NTF ! {
663     \__stex_module_invoke_uri:nN { #1 }
664   } {
665     \peek_charcode_remove:NTF ? {
666       \__stex_module_invoke_symbol:nn { #1 }
667     } {
668       \msg_set:nnn{stex}{error/syntax}{
669         Syntax~error:~?~or~!~expected~after~
670         \c_backslash_str STEXModule{#1}
671       }
672       \msg_error:nn{stex}{error/syntax}
673     }
674   }
675 }
676
677 \cs_new_protected:Nn \__stex_module_invoke_uri:nN {
678   \str_set:Nn #2 { #1 }
679 }
680
681 \cs_new_protected:Nn \__stex_module_invoke_symbol:nn {
682   \stex_invoke_symbol:n{#1?#2}
683 }

```

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

module module arguments:

```

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

```

```

706     \l_stex_module_title_str
707 \exp_args:NNo \str_set:Nn \l_stex_module_ns_str
708     \l_stex_module_ns_str
709 \exp_args:NNo \str_set:Nn \l_stex_module_lang_str
710     \l_stex_module_lang_str
711 \exp_args:NNo \str_set:Nn \l_stex_module_sig_str
712     \l_stex_module_sig_str
713 \exp_args:NNo \str_set:Nn \l_stex_module_meta_str
714     \l_stex_module_meta_str
715 \exp_args:NNo \str_set:Nn \l_stex_module_creators_str
716     \l_stex_module_creators_str
717 \exp_args:NNo \str_set:Nn \l_stex_module_contributors_str
718     \l_stex_module_contributors_str
719 }

```

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

```

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

```

```

758
759 \str_if_empty:NF \l_stex_module_lang_str {
760   \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
761   \l_tmpa_str {
762     \ltx@ifpackageloaded{babel}{
763       \exp_args:Nx \selectlanguage { \l_tmpa_str }
764     }{}
765   } {
766     \msg_set:nnn{stex}{error/unknownlanguage}{
767       Unknown~language~\l_tmpa_str
768     }
769     \msg_error:nn{stex}{error/unknownlanguage}
770   }
771 }
772
773 % signature
774 \str_if_empty:NTF \l_stex_module_sig_str {
775   \str_clear:N \l_tmpa_str
776   \seq_clear:N \l_tmpa_seq
777   \tl_clear:N \l_tmpa_tl
778   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
779     name      = \l_stex_module_name_str ,
780     ns        = \l_stex_module_ns_str ,
781     imports   = \exp_not:o { \l_tmpa_seq } ,
782     constants = \exp_not:o { \l_tmpa_seq } ,
783     content   = \exp_not:o { \l_tmpa_tl } ,
784     file      = \exp_not:o { \g_stex_currentfile_seq } ,
785     lang      = \l_stex_module_lang_str ,
786     sig       = \l_stex_module_sig_str ,
787     meta      = \l_stex_module_meta_str
788   }
789 }{
790   \str_if_empty:NT \l_stex_module_lang_str {
791     \msg_set:nnn{stex}{error/siglanguage}{
792       Module~\l_stex_module_ns_str?\l_stex_module_name_str~
793       declares~signature~\l_stex_module_sig_str,~but~does~not~
794       declare~its~language
795     }
796     \msg_error:nn{stex}{error/siglanguage}
797   }
798
799   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
800   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
801   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
802   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
803   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
804   \str_set:Nx \l_tmpa_str {
805     \stex_path_to_string:N \l_tmpa_seq /
806     \l_tmpa_str . \l_stex_module_sig_str .tex
807   }
808   \IfFileExists \l_tmpa_str {
809     \exp_args:No \stex_in_smsmode:nn { \l_tmpa_str } {
810       \seq_clear:N \l_stex_all_modules_seq
811       \prop_clear:N \l_stex_current_module_prop

```

```

812     \stex_debug:n{Loading~signature~\l_tmpa_str}
813     \input { \l_tmpa_str }
814   }
815   ){
816     \msg_set:nnn{stex}{error/modulemissing}{
817       No~file~for~signature~module~\l_tmpa_str~found
818     }
819     \msg_error:nn{stex}{error/modulemissing}
820   }
821   \stex_activate_module:n {
822     \l_stex_module_ns_str ? \l_stex_module_name_str
823   }
824   \prop_set_eq:Nc \l_stex_current_module_prop {
825     c_stex_module_
826     \l_stex_module_ns_str ?
827     \l_stex_module_name_str
828     _prop
829   }
830 }
831
832 % metatheory
833 \str_if_empty:NT \l_stex_module_meta_str {
834   \str_set:Nx \l_stex_module_meta_str {
835     \c_stex_metatheory_ns_str ? Metatheory
836   }
837 }
838
839
840 \stex_debug:n{
841   New~module:\\
842   Namespace:~\l_stex_module_ns_str\\
843   Name:~\l_stex_module_name_str\\
844   Language:~\l_stex_module_lang_str\\
845   Signature:~\l_stex_module_sig_str\\
846   Metatheory:~\l_stex_module_meta_str\\
847   File:~\stex_path_to_string:N \g_stex_currentfile_seq
848 }
849
850 \seq_put_right:Nx \l_stex_all_modules_seq {
851   \l_stex_module_ns_str ? \l_stex_module_name_str
852 }
853
854 \seq_gput_right:Nx \g_stex_modules_in_file_seq
855   { \l_stex_module_ns_str ? \l_stex_module_name_str }
856
857 \stex_if_smsmode:TF {
858   \stex_smsmode_set_codes:
859 } {
860   \begin{stex_annotate_env} {theory} {
861     \l_stex_module_ns_str ? \l_stex_module_name_str
862   }
863
864   \stex_annotate_invisible:nnn{header}{} {
865     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}

```

```

866     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
867     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
868         \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
869     }
870 }
871 }
872
873 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
874     \exp_args:Nx \STEXexport{
875         \stex_activate_module:n { \l_stex_module_meta_str }
876     }
877 }
878 % TODO: Inherit metatheory for nested modules?
879 }
880 \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}`

```

881 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
882 \cs_new_protected:Nn \_stex_module_end_module: {
883     \str_set:Nx \l_tmpa_str {
884         c_stex_module_
885         \prop_item:Nn \l_stex_current_module_prop { ns } ?
886         \prop_item:Nn \l_stex_current_module_prop { name }
887         _prop
888     }
889     %^^A \prop_new:c { \l_tmpa_str }
890     \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
891     \stex_debug:n{Closing module~\prop_item:Nn \l_stex_current_module_prop { name }}
892     \stex_if_smsmode:TF {
893         \exp_args:Nx \stex_addtosms:n {
894             \prop_gset_from_keyval:cn {
895                 c_stex_module_
896                 \prop_item:Nn \l_stex_current_module_prop { ns } ?
897                 \prop_item:Nn \l_stex_current_module_prop { name }
898                 _prop
899             } {
900                 name      = \prop_item:cn { \l_tmpa_str } { name } ,
901                 ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
902                 imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
903                 constants = \prop_item:cn { \l_tmpa_str } { constants } ,
904                 content    = \prop_item:cn { \l_tmpa_str } { content } ,
905                 file      = \prop_item:cn { \l_tmpa_str } { file } ,
906                 lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
907                 sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
908                 meta      = \prop_item:cn { \l_tmpa_str } { meta }
909             }
910         }
911     }{
912         \end{stex_annotate_env}
913     }
914 }

```

(End definition for `_stex_module_end_module:.`)

`@module` The core environment, with no header

```

915 \NewDocumentEnvironment { @module } { 0{} m } {
916   \str_set:Nx \l_stex_module_name_str { #2 }
917   \par
918   \__stex_module_args:n { #1 }
919   \__stex_module_begin_module:
920 } {
921   \__stex_module_end_module:
922 }
```

`\stex_modules_heading:` Code for document headers

```

923 \cs_if_exist:NTF \thesection {
924   \newcounter{module}[section]
925 }{
926   \newcounter{module}
927 }
928
929 \bool_if:NT \c_stex_showmods_bool {
930   \latexml_if:F { \RequirePackage{mdframed} }
931 }
932
933 \cs_new_protected:Nn \stex_modules_heading: {
934   \stepcounter{module}
935   \par
936   \bool_if:NT \c_stex_showmods_bool {
937     \noindent{\textbf{Module} ~
938       \cs_if_exist:NT \thesection {\thesection.}
939       \themodule ~ [\l_stex_module_name_str]
940     }
941     % TODO references
942     % \sref@label@id{Module \thesection.\themodule [\module@name]]%
943     \str_if_empty:NTF \l_stex_module_title_str {
944       }{
945         \quad(\l_stex_module_title_str)\hfill
946       }\par
947     }
948 }
```

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

Finally:

```

949 \NewDocumentEnvironment { module } { 0{} m } {
950   \bool_if:NT \c_stex_showmods_bool {
951     \begin{mdframed}
952   }
953   \begin{@module}[#1]{#2}
954   \stex_modules_heading:
955 }{
956   \end{@module}
957   \bool_if:NT \c_stex_showmods_bool {
958     \end{mdframed}
959   }
960 }
```

4.5.2 SMS Mode

961 $\langle @@=\text{stex_smsmode} \rangle$

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

```

962 \tl_new:N \g_stex_smsmode_allowedmacros_tl
963 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
964 \seq_new:N \g_stex_smsmode_allowedenvs_seq
965
966 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
967   \makeatletter
968   \makeatother
969   \ExplSyntaxOn
970   \ExplSyntaxOff
971 }
972
973 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
974   \symdef
975   \importmodule
976   \notation
977   \symdecl
978   \STEXexport
979 }
980
981 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
982   \tl_to_str:n {
983     module,
984     @module
985   }
986 }

```

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

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

```

987 \bool_new:N \g__stex_smsmode_bool
988 \bool_set_false:N \g__stex_smsmode_bool
989 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
990   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
991 }

```

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

$\backslash\text{__stex_smsmode_if_catcodes_p}$:
 $\backslash\text{__stex_smsmode_if_catcodes}$: \underline{TF}

Checks whether the SMS mode category code scheme is active.

```

992 \bool_new:N \g__stex_smsmode_catcode_bool
993 \bool_set_false:N \g__stex_smsmode_catcode_bool
994 \prg_new_conditional:Nnn \__stex_smsmode_if_catcodes: { p, T, F, TF } {
995   \bool_if:NTF \g__stex_smsmode_catcode_bool
996     \prg_return_true: \prg_return_false:
997 }

```

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

`\stex_smsmode_set_codes:`

```
998 \cs_new_protected:Nn \stex_smsmode_set_codes: {
999   \stex_if_smsmode:T {
1000     \__stex_smsmode_if_catcodes:F {
1001       \bool_gset_true:N \g__stex_smsmode_catcode_bool
1002       \exp_after:wN \char_gset_active_eq:NN
1003       \c_backslash_str \__stex_smsmode_cs:
1004       \tex_global:D \char_set_catcode_active:N \
1005       \tex_global:D \char_set_catcode_other:N $
1006       \tex_global:D \char_set_catcode_other:N ^
1007       \tex_global:D \char_set_catcode_other:N _
1008       \tex_global:D \char_set_catcode_other:N &
1009       \tex_global:D \char_set_catcode_other:N ##
1010     }
1011   }
1012 } \iffalse $ \fi % to make syntax highlighting work again
```

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

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

```
1013 \cs_new_protected:Nn \__stex_smsmode_unset_codes: {
1014   \__stex_smsmode_if_catcodes:T {
1015     \bool_gset_false:N \g__stex_smsmode_catcode_bool
1016     \exp_after:wN \tex_global:D \exp_after:wN
1017     \char_set_catcode_escape:N \c_backslash_str
1018     \tex_global:D \char_set_catcode_math_toggle:N $
1019     \tex_global:D \char_set_catcode_math_superscript:N ^
1020     \tex_global:D \char_set_catcode_math_subscript:N _
1021     \tex_global:D \char_set_catcode_alignment:N &
1022     \tex_global:D \char_set_catcode_parameter:N ##
1023   }
1024 } \iffalse $ \fi % to make syntax highlighting work again
```

(End definition for __stex_smsmode_unset_codes:.)

`\stex_in_smsmode:nn`

```
1025 \cs_new_protected:Nn \stex_in_smsmode:nn {
1026   \vbox_set:Nn \l_tmpa_box {
1027     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1028     \bool_gset_true:N \g__stex_smsmode_bool
1029     \stex_smsmode_set_codes:
1030     #2
1031     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1032     \stex_if_smsmode:F {
1033       \__stex_smsmode_unset_codes:
1034     }
1035   }
1036   \box_clear:N \l_tmpa_box
1037 }
```

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

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

```

1038 \cs_new_protected:Nn \__stex_smsmode_cs: {
1039   \str_clear:N \l_tmpa_str
1040   \peek_analysis_map_inline:n {
1041     % #1: token (one expansion)
1042     % #2: charcode
1043     % #3 catcode
1044     \token_if_eq_charcode:NNTF ##3 B {
1045       % token is a letter
1046       \exp_args:NNNo \str_put_right:Nn \l_tmpa_str { ##1 }
1047     } {
1048       \str_if_empty:NNTF \l_tmpa_str {
1049         % we don't allow (or need) single non-letter CSs
1050         % for now
1051         \peek_analysis_map_break:
1052       } {
1053         \str_if_eq:ontf \l_tmpa_str { begin } {
1054           \peek_analysis_map_break:n {
1055             \exp_after:wN \__stex_smsmode_checkbegin:n ##1
1056           }
1057         } {
1058           \str_if_eq:ontf \l_tmpa_str { end } {
1059             \peek_analysis_map_break:n {
1060               \exp_after:wN \__stex_smsmode_checkend:n ##1
1061             }
1062           } {
1063             \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
1064             \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1065               \g_stex_smsmode_allowedmacros_tl
1066               { \use:c{\l_tmpa_str} } {
1067               \stex_debug:n{Executing~1:~\l_tmpa_str}
1068               \peek_analysis_map_break:n {
1069                 \exp_after:wN \l_tmpa_tl ##1
1070               }
1071             } {
1072               \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1073               \g_stex_smsmode_allowedmacros_escape_tl
1074               { \use:c{\l_tmpa_str} } {
1075               \stex_debug:n{Executing~2:~\l_tmpa_str}
1076               % TODO \__stex_smsmode_rescan_cs:
1077               \exp_after:wN \exp_after:wN \exp_after:wN
1078               \token_if_eq_charcode:NNTF \exp_after:wN \c_backslash_str ##1 {
1079               \peek_analysis_map_break:n {
1080                 \__stex_smsmode_unset_codes:
1081                 \__stex_smsmode_rescan_cs:
1082               }
1083             } {
1084               \peek_analysis_map_break:n {
1085                 \__stex_smsmode_unset_codes:
1086                 \exp_after:wN \l_tmpa_tl ##1
1087               }
1088             }
1089           } {
1090             \peek_analysis_map_break:n { ##1 }
1091           }

```

```

1092     }
1093   }
1094 }
1095 }
1096 }
1097 }
1098 }

```

(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:

```

1099 \cs_new_protected:Nn \_stex_smsmode_rescan_cs: {
1100   \str_clear:N \l_tmpb_str
1101   \peek_analysis_map_inline:n {
1102     \token_if_eq_charcode:NNTF ##3 B {
1103       % token is a letter
1104       \exp_args:NNo \str_put_right:Nn \l_tmpb_str { ##1 }
1105     } {
1106       \peek_analysis_map_break:n {
1107         \exp_after:wN \use:c \exp_after:wN {
1108           \exp_after:wN \l_tmpa_str\exp_after:wN
1109         } \use:c { \l_tmpb_str \exp_after:wN } ##1
1110       }
1111     }
1112   }
1113 }

```

(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.

```

1114 \cs_new_protected:Nn \_stex_smsmode_checkbegin:n {
1115   \str_set:Nn \l_tmpa_str { #1 }
1116   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1117     \_stex_smsmode_unset_codes:
1118     \begin{#1}
1119   }
1120 }

```

(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.

```

1121 \cs_new_protected:Nn \_stex_smsmode_checkend:n {
1122   \str_set:Nn \l_tmpa_str { #1 }
1123   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1124     \end{#1}
1125   }
1126 }

```

(End definition for `_stex_smsmode_checkend:n`.)

4.5.3 Inheritance

1127 <@@=stex_importmodule>

\stex_import_module_uri:nn

```

1128 \cs_new_protected:Nn \stex_import_module_uri:nn {
1129   \str_set:Nx \l__stex_importmodule_archive_str { #1 }
1130   \str_set:Nx \l__stex_importmodule_path_str { #2 }
1131   \str_if_empty:NT \l__stex_importmodule_archive_str {
1132     \prop_if_empty:NF \l_stex_current_repository_prop {
1133       \prop_get:NnN \l_stex_current_repository_prop { id } \l__stex_importmodule_archive_str
1134     }
1135   }
1136
1137   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l__stex_importmodule_path_str }
1138   \seq_pop_right:NN \l_tmpb_seq \l__stex_importmodule_name_str
1139   \str_set:Nx \l__stex_importmodule_path_str { \seq_use:Nn \l_tmpb_seq ? }
1140
1141   \str_if_empty:NTF \l__stex_importmodule_archive_str {
1142     \stex_modules_current_namespace:
1143     \str_if_empty:NF \l__stex_importmodule_path_str {
1144       \str_set:Nx \l_stex_module_ns_str {
1145         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1146       }
1147     }
1148   }{
1149     \stex_require_repository:n \l__stex_importmodule_archive_str
1150     \prop_get:cnN { c_stex_mathhub\l__stex_importmodule_archive_str _manifest_prop } { ns }
1151     \l_stex_module_ns_str
1152     \str_if_empty:NF \l__stex_importmodule_path_str {
1153       \str_set:Nx \l_stex_module_ns_str {
1154         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1155       }
1156     }
1157   }
1158 }

```

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

\l_stex_importmodule_name_str
\l_stex_importmodule_archive_str
\l_stex_importmodule_path_str
\l_stex_importmodule_file_str

Store the return values of \stex_import_module_uri:nn.

```

1159 \str_new:N \l__stex_importmodule_name_str
1160 \str_new:N \l__stex_importmodule_archive_str
1161 \str_new:N \l__stex_importmodule_path_str
1162 \str_new:N \g__stex_importmodule_file_str

```

(End definition for \l__stex_importmodule_name_str and others.)

\stex_import_require_module:nnnn

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

```

1163 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1164   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1165     % \stex_debug:n{Arguments: #1, #2, #3, #4}
1166
1167     % archive
1168     \str_set:Nx \l_tmpa_str { #2 }
1169     \str_if_empty:NTF \l_tmpa_str {

```

```

1170 \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1171 } {
1172 \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1173 \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1174 \seq_put_right:Nn \l_tmpa_seq { source }
1175 }
1176
1177 % path
1178 \str_set:Nx \l_tmpb_str { #3 }
1179 \str_if_empty:NTF \l_tmpb_str {
1180 \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1181
1182 \ltx@ifpackageloaded{babel} {
1183 \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1184 { \language } \l_tmpb_str {
1185 \msg_set:nnn{stex}{error/unknownlanguage}{
1186 Unknown-language-\language
1187 }
1188 \msg_error:nn{stex}{error/unknownlanguage}
1189 }
1190 } {
1191 \str_clear:N \l_tmpb_str
1192 }
1193
1194 \stex_debug:n{Checking-\l_tmpa_str.\l_tmpb_str.tex}
1195 \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1196 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1197 }{
1198 \stex_debug:n{Checking-\l_tmpa_str.tex}
1199 \IfFileExists{ \l_tmpa_str.tex }{
1200 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1201 }{
1202 % try english as default
1203 \stex_debug:n{Checking-\l_tmpa_str.en.tex}
1204 \IfFileExists{ \l_tmpa_str.en.tex }{
1205 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1206 }{
1207 \msg_set:nnn{stex}{error/modulemissing}{
1208 No-file-for-module-#1?#4-found
1209 }
1210 \msg_error:nn{stex}{error/modulemissing}
1211 }
1212 }
1213 }
1214
1215 } {
1216 \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1217 \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1218
1219 \ltx@ifpackageloaded{babel} {
1220 \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1221 { \language } \l_tmpb_str {
1222 \msg_set:nnn{stex}{error/unknownlanguage}{
1223 Unknown-language-\language

```

```

1224     }
1225     \msg_error:nn{stex}{error/unknownlanguage}
1226   }
1227 } {
1228   \str_clear:N \l_tmpb_str
1229 }
1230
1231 \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1232
1233 \stex_debug:n{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1234 \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1235   \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1236 }{
1237   \stex_debug:n{Checking~\l_tmpa_str/#4.tex}
1238   \IfFileExists{ \l_tmpa_str/#4.tex }{
1239     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1240   }{
1241     % try english as default
1242     \stex_debug:n{Checking~\l_tmpa_str/#4.en.tex}
1243     \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1244       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1245     }{
1246       \stex_debug:n{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1247       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1248         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1249       }{
1250         \stex_debug:n{Checking~\l_tmpa_str.tex}
1251         \IfFileExists{ \l_tmpa_str.tex }{
1252           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1253         }{
1254           % try english as default
1255           \stex_debug:n{Checking~\l_tmpa_str.en.tex}
1256           \IfFileExists{ \l_tmpa_str.en.tex }{
1257             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1258           }{
1259             \msg_set:nnn{stex}{error/modulemissing}{
1260               No~file~for~module~#1?#4~found
1261             }
1262             \msg_error:nn{stex}{error/modulemissing}
1263           }
1264         }
1265       }
1266     }
1267   }
1268 }
1269 }
1270
1271 \seq_set_eq:NN \l_tmpa_seq \g_stex_modules_in_file_seq
1272 \seq_clear:N \g_stex_modules_in_file_seq
1273 % \exp_args:Nnx \use:nn {
1274   \exp_args:No \stex_in_smsmode:nn { \g__stex_importmodule_file_str } {
1275     \seq_clear:N \l_stex_all_modules_seq
1276     \prop_clear:N \l_stex_current_module_prop
1277     \str_set:Nx \l_tmpb_str { #2 }

```

```

1278     \str_if_empty:NF \l_tmpb_str {
1279       \stex_set_current_repository:n { #2 }
1280     }
1281     \stex_debug:n{Loading~\g__stex_importmodule_file_str}
1282     \input { \g__stex_importmodule_file_str }
1283   }
1284 %   }{
1285
1286 %   }
1287   \prop_gput:Noo \g_stex_module_files_prop
1288   \g__stex_importmodule_file_str \g_stex_modules_in_file_seq
1289   \seq_set_eq:NN \g_stex_modules_in_file_seq \l_tmpa_seq
1290
1291   \stex_if_module_exists:nF { #1 ? #4 } {
1292     \msg_set:nnn{stex}{error/modulemissing}{
1293       Module~#1?#4~not~found~in~file~\g__stex_importmodule_file_str
1294     }
1295     \msg_error:nn{stex}{error/modulemissing}
1296   }
1297 }
1298 \stex_activate_module:n { #1 ? #4 }
1299 }

```

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

`\stex_activate_module:n`

```

1300 \cs_new_protected:Nn \stex_activate_module:n {
1301   \stex_debug:n{Activating~module~#1}
1302   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1303     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1304     \prop_item:cn { c_stex_module_#1_prop } { content }
1305   }
1306 }

```

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

`\importmodule`

```

1307 \NewDocumentCommand \importmodule { 0{} m } {
1308   \stex_import_module_uri:nn { #1 } { #2 }
1309   \stex_debug:n{Importing~module:~
1310     \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1311   }
1312   \stex_if_smsmode:F {
1313     \stex_import_require_module:nnnn
1314     { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1315     { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1316     \stex_annotate_invisible:nnn
1317     {import} { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } {}
1318   }
1319   \exp_args:Nx \stex_add_to_current_module:n {
1320     \stex_import_require_module:nnnn
1321     { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1322     { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1323   }
1324   \exp_args:Nx \stex_add_import_to_current_module:n {

```

```

1325 \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1326 }
1327 \stex_smsmode_set_codes:
1328 }

```

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

`\usemodule`

```

1329 \NewDocumentCommand \usemodule { 0{} m } {
1330 \stex_if_smsmode:F {
1331 \stex_import_module_uri:nn { #1 } { #2 }
1332 \stex_import_require_module:nnnn
1333 { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1334 { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1335 \stex_annotate_invisible:nnn
1336 {usemodule} {\l_stex_module_ns_str ? \l__stex_importmodule_name_str} {}
1337 }
1338 \stex_smsmode_set_codes:
1339 }

```

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

`\g_stex_modules_in_file_seq` `\g_stex_module_files_prop`

```

1340 \seq_new:N \g_stex_modules_in_file_seq
1341 \prop_new:N \g_stex_module_files_prop

```

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

4.6 Symbol Declarations

```

1342 <@@=stex_symdecl>

```

`\l_stex_all_symbols_seq` Stores all available symbols

```

1343 \seq_new:N \l_stex_all_symbols_seq

```

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

`\STEXsymbol`

```

1344 \NewDocumentCommand \STEXsymbol { m } {
1345 \stex_get_symbol:n { #1 }
1346 \exp_args:No
1347 \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1348 }

```

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

`symdecl` arguments:

```

1349 \keys_define:nn { stex / symdecl } {
1350 name .tl_set:x:N = \l_stex_symdecl_name_str ,
1351 local .bool_set:N = \l_stex_symdecl_local_bool ,
1352 args .tl_set:x:N = \l_stex_symdecl_args_str ,
1353 type .tl_set:N = \l_stex_symdecl_type_tl ,
1354 align .tl_set:N = \l_stex_symdecl_align_str , % TODO(?)
1355 gfc .tl_set:N = \l_stex_symdecl_gfc_str , % TODO(?)
1356 specializes .tl_set:N = \l_stex_symdecl_specializes_str , % TODO(?)

```



```

1357   def          .tl_set:N      = \l_stex_symdecl_definiens_tl
1358 }
1359
1360 \bool_new:N \l_stex_symdecl_make_macro_bool
1361
1362 \cs_new_protected:Nn \__stex_symdecl_args:n {
1363   \str_clear:N \l_stex_symdecl_name_str
1364   \str_clear:N \l_stex_symdecl_args_str
1365   \bool_set_false:N \l_stex_symdecl_local_bool
1366   \tl_clear:N \l_stex_symdecl_type_tl
1367   \tl_clear:N \l_stex_symdecl_definiens_tl
1368
1369   \keys_set:nn { stex /symdecl } { #1 }
1370
1371   \exp_args:NNo \str_set:Nn \l_stex_symdecl_name_str
1372     \l_stex_symdecl_name_str
1373   \exp_args:NNo \str_set:Nn \l_stex_symdecl_args_str
1374     \l_stex_symdecl_args_str
1375 }

```

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

```

1376
1377 \NewDocumentCommand \symdecl { s O{} m } {
1378   \__stex_symdecl_args:n { #2 }
1379   \IfBooleanTF #1 {
1380     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1381   } {
1382     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1383   }
1384   \stex_symdecl_do:n { #3 }
1385   \stex_smsmode_set_codes:
1386 }

```

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

\stex_symdecl_do:n

```

1387 \cs_new_protected:Nn \stex_symdecl_do:n {
1388   \stex_if_in_module:F {
1389     % TODO throw error? some default namespace?
1390   }
1391
1392   \str_if_empty:NT \l_stex_symdecl_name_str {
1393     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1394   }
1395
1396   \prop_if_exist:cT { g_stex_symdecl_
1397     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1398     \prop_item:Nn \l_stex_current_module_prop {name} ?
1399     \l_stex_symdecl_name_str
1400     _prop
1401   }{
1402     % TODO throw error (beware of circular dependencies)
1403   }

```

```

1404
1405 \prop_clear:N \l_tmpa_prop
1406 \prop_put:Nnx \l_tmpa_prop { module } {
1407   \prop_item:Nn \l_stex_current_module_prop {ns} ?
1408   \prop_item:Nn \l_stex_current_module_prop {name}
1409 }
1410 \seq_clear:N \l_tmpa_seq
1411 \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
1412 \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1413 \prop_put:Nno \l_tmpa_prop { local } \l_stex_symdecl_local_bool
1414 \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1415
1416 \exp_args:No \stex_add_constant_to_current_module:n {
1417   \l_stex_symdecl_name_str
1418 }
1419
1420 % arity/args
1421 \int_zero:N \l_tmpb_int
1422
1423 \bool_set_true:N \l_tmpa_bool
1424 \str_map_inline:Nn \l_stex_symdecl_args_str {
1425   \token_case_meaning:NnF ##1 {
1426     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1427     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1428     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1429     {\tl_to_str:n a} {
1430       \bool_set_false:N \l_tmpa_bool
1431       \int_incr:N \l_tmpb_int
1432     }
1433     {\tl_to_str:n B} {
1434       \bool_set_false:N \l_tmpa_bool
1435       \int_incr:N \l_tmpb_int
1436     }
1437   }{
1438     \msg_set:nnn{stex}{error/wrongargs}{
1439       args~value~in~symbol~declaration~for~
1440       \prop_item:Nn \l_stex_current_module_prop {ns} ?
1441       \prop_item:Nn \l_stex_current_module_prop {name} ?
1442       \l_stex_symdecl_name_str ~
1443       needs~to~be~
1444       i,~a,~b~or~B,~but~##1~given
1445     }
1446     \msg_error:nn{stex}{error/wrongargs}
1447   }
1448 }
1449 \bool_if:NTF \l_tmpa_bool {
1450   % possibly numeric
1451   \str_if_empty:NTF \l_stex_symdecl_args_str {
1452     \prop_put:Nnn \l_tmpa_prop { args } {}
1453     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1454   }{
1455     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1456     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1457     \str_clear:N \l_tmpa_str

```

```

1458     \int_step_inline:nn \l_tmpa_int {
1459       \str_put_right:Nn \l_tmpa_str i
1460     }
1461     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1462   }
1463 } {
1464   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1465   \prop_put:Nnx \l_tmpa_prop { arity }
1466     { \str_count:N \l_stex_symdecl_args_str }
1467 }
1468 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
1469
1470
1471 % semantic macro
1472
1473 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1474   \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1475     \prop_item:Nn \l_tmpa_prop { module } ?
1476     \prop_item:Nn \l_tmpa_prop { name }
1477   } }
1478
1479   \bool_if:NF \l_stex_symdecl_local_bool {
1480     \exp_args:Nx \stex_add_to_current_module:n {
1481       \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1482         \prop_item:Nn \l_tmpa_prop { module } ?
1483         \prop_item:Nn \l_tmpa_prop { name }
1484       } }
1485     }
1486   }
1487 }
1488
1489 % add to all symbols
1490
1491 \bool_if:NF \l_stex_symdecl_local_bool {
1492   \exp_args:Nx \stex_add_to_current_module:n {
1493     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1494       \prop_item:Nn \l_tmpa_prop { module } ?
1495       \prop_item:Nn \l_tmpa_prop { name }
1496     }
1497   }
1498 }
1499
1500 \stex_debug:n{New~symbol:~
1501   \prop_item:Nn \l_tmpa_prop { module } ?
1502   \prop_item:Nn \l_tmpa_prop { name }^^J
1503   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
1504   Args:~\prop_item:Nn \l_tmpa_prop { args }
1505 }
1506
1507 % circular dependencies require this:
1508
1509 \prop_if_exist:cF {
1510   g_stex_symdecl_
1511   \prop_item:Nn \l_tmpa_prop { module } ?

```

```

1512     \prop_item:Nn \l_tmpa_prop { name }
1513     _prop
1514   } {
1515     \prop_gset_eq:cN {
1516       g_stex_symdecl_
1517       \prop_item:Nn \l_tmpa_prop { module } ?
1518       \prop_item:Nn \l_tmpa_prop { name }
1519       _prop
1520     } \l_tmpa_prop
1521   }
1522
1523   \stex_if_smsmode:TF {
1524     \bool_if:NF \l_stex_symdecl_local_bool {
1525       \exp_args:Nx \stex_addtosms:n {
1526         \prop_gset_from_keyval:cn {
1527           g_stex_symdecl_
1528           \prop_item:Nn \l_tmpa_prop { module } ?
1529           \prop_item:Nn \l_tmpa_prop { name }
1530           _prop
1531         } {
1532           name      = \prop_item:Nn \l_tmpa_prop { name }      ,
1533           module    = \prop_item:Nn \l_tmpa_prop { module }    ,
1534           notations = \prop_item:Nn \l_tmpa_prop { notations } ,
1535           local     = \prop_item:Nn \l_tmpa_prop { local }     ,
1536           type      = \prop_item:Nn \l_tmpa_prop { type }      ,
1537           args      = \prop_item:Nn \l_tmpa_prop { args }      ,
1538           arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
1539           assocs    = \prop_item:Nn \l_tmpa_prop { assocs }
1540         }
1541         \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1542           \prop_item:Nn \l_tmpa_prop { module } ?
1543           \prop_item:Nn \l_tmpa_prop { name }
1544         }
1545       }
1546     }
1547   }{
1548     \exp_args:NNx \seq_put_right:Nn \l_stex_all_symbols_seq {
1549       \prop_item:Nn \l_tmpa_prop { module } ?
1550       \prop_item:Nn \l_tmpa_prop { name }
1551     }
1552     \stex_annotate_invisible:nnn {symdecl} {
1553       \prop_item:Nn \l_tmpa_prop { module } ?
1554       \prop_item:Nn \l_tmpa_prop { name }
1555     } {
1556       \stex_annotate_invisible:nnn{type}{}{\l_stex_symdecl_type_tl$}
1557       \stex_annotate_invisible:nnn{args}{}{
1558         \prop_item:Nn \l_tmpa_prop { args }
1559       }
1560       \stex_annotate_invisible:nnn{macroname}{}{#1}
1561       \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
1562         \stex_annotate_invisible:nnn{definiens}{}{
1563           {\l_stex_symdecl_definiens_tl$}
1564         }
1565       }

```

```

1566 }
1567 }

```

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

`\stex_get_symbol:n`

```

1568 \str_new:N \l_stex_get_symbol_uri_str
1569
1570 \cs_new_protected:Nn \stex_get_symbol:n {
1571   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
1572     \__stex_symdecl_get_symbol_from_cs:n { #1 }
1573   }{
1574     % argument is a string
1575     % is it a command name?
1576     \cs_if_exist:cTF { #1 }{
1577       \cs_set_eq:Nc \l_tmpa_tl { #1 }
1578       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
1579       \str_if_empty:NNTF \l_tmpa_str {
1580         \exp_args:Nx \cs_if_eq:NNTF {
1581           \tl_head:N \l_tmpa_tl
1582         } \stex_invoke_symbol:n {
1583           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
1584         }{
1585           \__stex_symdecl_get_symbol_from_string:n { #1 }
1586         }
1587       } {
1588         \__stex_symdecl_get_symbol_from_string:n { #1 }
1589       }
1590     }{
1591       % argument is not a command name
1592       \__stex_symdecl_get_symbol_from_string:n { #1 }
1593       % \l_stex_all_symbols_seq
1594     }
1595   }
1596 }
1597
1598 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
1599   \bool_set_false:N \l_tmpa_bool
1600   \stex_if_in_module:T {
1601     \prop_get:NnN \l_stex_current_module_prop
1602     { constants } \l_tmpa_seq
1603     \exp_args:NNo \seq_if_in:NnTF \l_tmpa_seq { \l_tmpa_str } {
1604       \bool_set_true:N \l_tmpa_bool
1605       \str_set:Nx \l_stex_get_symbol_uri_str {
1606         \prop_item:Nn \l_stex_current_module_prop { ns } ?
1607         \prop_item:Nn \l_stex_current_module_prop { name } ? #1
1608       }
1609     }
1610   }
1611   \bool_if:NF \l_tmpa_bool {
1612     \tl_set:Nn \l_tmpa_tl {
1613       \msg_set:nnn{stex}{error/unknownsymbol}{
1614         No~symbol~#1~found!
1615       }
1616     }
1617   }

```

```

1616     \msg_error:nn{stex}{error/unknownsymbol}
1617   }
1618   \str_set:Nn \l_tmpa_str { #1 }
1619   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1620   \seq_map_inline:Nn \l_stex_all_symbols_seq {
1621     \str_set:Nn \l_tmpb_str { ##1 }
1622     \str_if_eq:eeT { \l_tmpa_str } {
1623       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1624     } {
1625       \seq_map_break:n {
1626         \tl_set:Nn \l_tmpa_tl {
1627           \str_set:Nn \l_stex_get_symbol_uri_str {
1628             ##1
1629           }
1630         }
1631       }
1632     }
1633   }
1634   \l_tmpa_tl
1635 }
1636 }
1637
1638 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
1639   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
1640     { \tl_tail:N \l_tmpa_tl }
1641   \tl_if_single:NTF \l_tmpa_tl {
1642     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
1643       \exp_after:wN \str_set:Nn \exp_after:wN
1644         \l_stex_get_symbol_uri_str \l_tmpa_tl
1645     }{
1646       % TODO
1647       % tail is not a single group
1648     }
1649   }{
1650     % TODO
1651     % tail is not a single group
1652   }
1653 }

```

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

4.7 Notations

```

1654 <@@=stex_notation>
1655 notation arguments:
1656 \keys_define:nn { stex / notation } {
1657   lang .tl_set_x:N = \l__stex_notation_lang_str ,
1658   variant .tl_set_x:N = \l__stex_notation_variant_str ,
1659   prec .tl_set_x:N = \l__stex_notation_prec_str ,
1660   op .tl_set:N = \l__stex_notation_op_tl ,
1661   unknown .code:n = \str_set:Nx
1662     \l__stex_notation_variant_str \l_keys_key_str
1663 }
1664

```

```

1664 \cs_new_protected:Nn \__stex_notation_args:n {
1665   \str_clear:N \l__stex_notation_lang_str
1666   \str_clear:N \l__stex_notation_variant_str
1667   \str_clear:N \l__stex_notation_prec_str
1668   \tl_clear:N \l__stex_notation_op_tl
1669
1670   \keys_set:nn { stex / notation } { #1 }
1671
1672   \str_set:Nx \l__stex_notation_lang_str \l__stex_notation_lang_str
1673   \str_set:Nx \l__stex_notation_variant_str \l__stex_notation_variant_str
1674   \str_set:Nx \l__stex_notation_prec_str \l__stex_notation_prec_str
1675 }

```

\notation

```

1676 \NewDocumentCommand \notation { 0{} m } {
1677   \__stex_notation_args:n { #1 }
1678   \tl_clear:N \l_stex_symdecl_definiens_tl
1679   \stex_get_symbol:n { #2 }
1680   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
1681 }

```

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

\stex_notation_do:nn

```

1682 \cs_new_protected:Nn \stex_notation_do:nn {
1683   \prop_set_eq:Nc \l_tmpa_prop {
1684     g_stex_symdecl_ #1 _prop
1685   }
1686
1687   \prop_clear:N \l_tmpb_prop
1688   \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
1689   \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
1690   \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
1691
1692   % precedences
1693   \seq_clear:N \l_tmpb_seq
1694   \exp_args:NNno
1695   \str_if_empty:NTF \l__stex_notation_prec_str {
1696     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1697     \int_compare:nNnTF \l_tmpa_str = 0 {
1698       \exp_args:NNnx
1699       \prop_put:Nno \l_tmpb_prop { opprec }
1700       { \infprec }
1701     }{
1702       \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
1703     }
1704   } {
1705     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
1706       \exp_args:NNnx
1707       \prop_put:Nno \l_tmpb_prop { opprec }
1708       { \infprec }
1709       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1710       \int_step_inline:nn { \l_tmpa_str } {
1711         \exp_args:NNx
1712         \seq_put_right:Nn \l_tmpb_seq { \neginfprec }

```

```

1713     }
1714   }{
1715     \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
1716     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
1717       \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
1718       \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
1719         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
1720         \l_tmpa_seq {\tl_to_str:n{x} } { \l_tmpa_str }
1721         \seq_map_inline:Nn \l_tmpa_seq {
1722           \seq_put_right:Nn \l_tmpb_seq { ##1 }
1723         }
1724       }
1725       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1726     }{
1727       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1728       \int_compare:nNnTF \l_tmpa_str = 0 {
1729         \exp_args:NNnx
1730         \prop_put:Nno \l_tmpb_prop { opprec }
1731         { \infprec }
1732       }{
1733         \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
1734       }
1735     }
1736   }
1737 }
1738
1739 \seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
1740 \int_step_inline:nn { \l_tmpa_str } {
1741   \seq_pop_left:NNTF \l_tmpa_seq \l_tmpb_str {
1742     \exp_args:NNx
1743     \seq_put_right:Nn \l_tmpb_seq {
1744       \prop_item:Nn \l_tmpb_prop { opprec }
1745     }
1746   }
1747 }
1748
1749 \prop_put:Nno \l_tmpb_prop { argprec } \l_tmpb_seq
1750 \tl_clear:N \l_tmpa_tl
1751
1752 \int_compare:nNnTF \l_tmpa_str = 0 {
1753   \exp_args:NNe
1754   \cs_set:Npn \l__stex_notation_macrocode_cs {
1755     \stex_term_math_oms:nnnn { #1 }
1756     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1757     { \prop_item:Nn \l_tmpb_prop { opprec } }
1758     { \exp_not:n { #2 } }
1759   }
1760   \__stex_notation_final:
1761 }{
1762   \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
1763   \str_if_in:NnTF \l_tmpb_str b {
1764     \exp_args:Nne \use:nn
1765     {
1766       \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs

```



```

1767 \cs_set:Npn \l_tmpa_str } { {
1768   \stex_term_math_omb:nnnn { #1 }
1769   { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1770   { \prop_item:Nn \l_tmpb_prop { opprec } }
1771   { \exp_not:n { #2 } }
1772 } }
1773 }{
1774 \str_if_in:NnTF \l_tmpb_str B {
1775   \exp_args:Nne \use:nn
1776   {
1777     \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1778     \cs_set:Npn \l_tmpa_str } { {
1779       \stex_term_math_omb:nnnn { #1 }
1780       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1781       { \prop_item:Nn \l_tmpb_prop { opprec } }
1782       { \exp_not:n { #2 } }
1783     } }
1784   }{
1785     \exp_args:Nne \use:nn
1786     {
1787       \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1788       \cs_set:Npn \l_tmpa_str } { {
1789         \stex_term_math_oma:nnnn { #1 }
1790         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1791         { \prop_item:Nn \l_tmpb_prop { opprec } }
1792         { \exp_not:n { #2 } }
1793       } }
1794     }
1795   }
1796
1797 \int_zero:N \l_tmpa_int
1798 \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
1799 \prop_get:NnN \l_tmpb_prop { argprecs } \l_tmpa_seq
1800 \__stex_notation_arguments:
1801 }
1802 }

```

(End definition for `\stex_notation_do:nn`. This function is documented on page 22.)

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

```

1803 \cs_new_protected:Nn \__stex_notation_arguments: {
1804   \int_incr:N \l_tmpa_int
1805   \str_if_empty:NNTF \l_tmpa_str {
1806     \__stex_notation_final:
1807   }{
1808     \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
1809     \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
1810     \str_if_eq:VnTF \l_tmpb_str a {
1811       \__stex_notation_argument_assoc:n
1812     }{
1813       \str_if_eq:VnTF \l_tmpb_str B {
1814         \__stex_notation_argument_assoc:n
1815       }{
1816         \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str

```

```

1817 \tl_put_right:Nx \l_tmpa_tl {
1818   { \stex_term_math_arg:nnn
1819     { \int_use:N \l_tmpa_int }
1820     { \l_tmpb_str }
1821     { ####\int_use:N \l_tmpa_int }
1822   }
1823 }
1824 \__stex_notation_arguments:
1825 }
1826 }
1827 }
1828 }

```

(End definition for `__stex_notation_arguments:.`)

`__stex_notation_argument_assoc:n`

```

1829 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
1830   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1831   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
1832   \tl_put_right:Nx \l_tmpa_tl {
1833     { \stex_term_math_assoc_arg:nnnn
1834       { \int_use:N \l_tmpa_int }
1835       { \l_tmpb_str }
1836       \exp_args:No \exp_not:n
1837       {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
1838       { ####\int_use:N \l_tmpa_int }
1839     }
1840   }
1841   \__stex_notation_arguments:
1842 }

```

(End definition for `__stex_notation_argument_assoc:n.`)

`__stex_notation_final:` Called after processing all notation arguments

```

1843 \cs_new_protected:Nn \__stex_notation_final: {
1844   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
1845   \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
1846   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
1847   \exp_args:Nne \use:nn
1848   {
1849     \cs_generate_from_arg_count:cNnn {
1850       stex_notation_ \l_tmpa_str \c_hash_str
1851       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1852       _cs
1853     }
1854     \cs_gset:Npn \l_tmpb_str { { {
1855       \exp_after:wN \exp_after:wN \exp_after:wN
1856       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
1857       { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
1858     } } }
1859
1860     \tl_if_empty:NF \l__stex_notation_op_tl {
1861       \cs_gset:cpx {
1862         stex_op_notation_ \l_tmpa_str \c_hash_str

```

```

1863     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1864     _cs
1865   } {
1866     \stex_term_oms:nnn {
1867       \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
1868       \l__stex_notation_lang_str
1869     }{
1870       \l_tmpa_str
1871     }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
1872   }
1873 }
1874
1875
1876
1877 \stex_debug:n{
1878   Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1879   ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
1880   Operator~precedence:~
1881   \prop_item:Nn \l_tmpb_prop { opprec }^^J
1882   Argument~precedences:~
1883   \seq_use:Nn \l_tmpa_seq {,~}^^J
1884   Notation: \cs_meaning:c {
1885     stex_notation_ \l_tmpa_str \c_hash_str
1886     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1887     _cs
1888   }
1889 }
1890
1891 \prop_gset_eq:cN {
1892   g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
1893   \c_hash_str \l__stex_notation_lang_str _prop
1894 } \l_tmpb_prop
1895
1896 \exp_args:Nx
1897 \stex_add_to_current_module:n {
1898   \prop_get:cnN {
1899     g_stex_symdecl_
1900     \prop_item:Nn \l_tmpb_prop { symbol }
1901     _prop
1902   } { notations } \exp_not:N \l_tmpa_seq
1903   \seq_put_right:Nn \exp_not:N \l_tmpa_seq {
1904     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1905   }
1906   \prop_put:cno {
1907     g_stex_symdecl_
1908     \prop_item:Nn \l_tmpb_prop { symbol }
1909     _prop
1910   } { notations } \exp_not:N \l_tmpa_seq
1911 }
1912
1913 \stex_if_smsmode:TF {
1914   \stex_smsmode_set_codes:
1915   \exp_args:Nx \stex_addtosms:n {
1916     \prop_gset_from_keyval:cn {

```

```

1917     g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
1918     \c_hash_str \l__stex_notation_lang_str _prop
1919   } {
1920     symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
1921     language    = \prop_item:Nn \l_tmpb_prop { language }    ,
1922     variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,
1923     opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
1924     argprec     = \prop_item:Nn \l_tmpb_prop { argprec }     ,
1925   }
1926 }
1927 }{
1928   \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
1929   \seq_put_right:Nx \l_tmpa_seq {
1930     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1931   }
1932   \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
1933   \prop_set_eq:cN {
1934     g_stex_symdecl_ \l_tmpa_str _prop
1935   } \l_tmpa_prop
1936
1937   % HTML annotations
1938   \stex_annotate_invisible:nnn { notation }
1939   { \prop_item:Nn \l_tmpb_prop { symbol } } {
1940     \stex_annotate_invisible:nnn { notationfragment }
1941     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
1942     \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
1943     \stex_annotate_invisible:nnn { precedence }
1944     { \prop_item:Nn \l_tmpb_prop { opprec };
1945       \seq_use:Nn \l_tmpa_seq { x }
1946     }{}
1947
1948     \int_zero:N \l_tmpa_int
1949     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
1950     \tl_clear:N \l_tmpa_tl
1951     \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
1952       \int_incr:N \l_tmpa_int
1953       \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
1954       \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
1955       \str_if_eq:VnTF \l_tmpb_str a {
1956         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
1957           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
1958           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
1959         } }
1960       }{
1961         \str_if_eq:VnTF \l_tmpb_str B {
1962           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
1963             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
1964             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
1965           } }
1966         }{
1967           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
1968             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
1969           } }
1970         }

```

```

1971     }
1972   }
1973   \stex_annotate_invisible:nnn { notationcomp }{}{
1974     $ \exp_args:Nno \use:nn { \use:c {
1975       stex_notation_ \prop_item:Nn \l_tmpb_prop { symbol }
1976       \c_hash_str \l__stex_notation_variant_str
1977       \c_hash_str \l__stex_notation_lang_str _cs
1978     } } { \l_tmpa_tl } $
1979   }
1980 }
1981 }
1982 }

```

(End definition for `__stex_notation_final:`)

\symdef

```

1983 \keys_define:nn { stex / symdef } {
1984   name .tl_set_x:N = \l_stex_symdecl_name_str ,
1985   local .bool_set:N = \l_stex_symdecl_local_bool ,
1986   args .tl_set_x:N = \l_stex_symdecl_args_str ,
1987   type .tl_set:N = \l_stex_symdecl_type_tl ,
1988   def .tl_set:N = \l_stex_symdecl_definiens_tl ,
1989   op .tl_set:N = \l__stex_notation_op_tl ,
1990   lang .tl_set_x:N = \l__stex_notation_lang_str ,
1991   variant .tl_set_x:N = \l__stex_notation_variant_str ,
1992   prec .tl_set_x:N = \l__stex_notation_prec_str ,
1993   unknown .code:n = \str_set:Nx
1994     \l__stex_notation_variant_str \l_keys_key_str
1995 }
1996
1997 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
1998   \str_clear:N \l_stex_symdecl_name_str
1999   \str_clear:N \l_stex_symdecl_args_str
2000   \bool_set_false:N \l_stex_symdecl_local_bool
2001   \tl_clear:N \l_stex_symdecl_type_tl
2002   \tl_clear:N \l_stex_symdecl_definiens_tl
2003   \str_clear:N \l__stex_notation_lang_str
2004   \str_clear:N \l__stex_notation_variant_str
2005   \str_clear:N \l__stex_notation_prec_str
2006   \tl_clear:N \l__stex_notation_op_tl
2007
2008   \keys_set:nn { stex / symdef } { #1 }
2009
2010   \exp_args:Nno \str_set:Nn \l_stex_symdecl_name_str
2011     \l_stex_symdecl_name_str
2012   \exp_args:Nno \str_set:Nn \l_stex_symdecl_args_str
2013     \l_stex_symdecl_args_str
2014   \exp_args:Nno \str_set:Nn \l__stex_notation_lang_str
2015     \l__stex_notation_lang_str
2016   \exp_args:Nno \str_set:Nn \l__stex_notation_variant_str
2017     \l__stex_notation_variant_str
2018   \exp_args:Nno \str_set:Nn \l__stex_notation_prec_str
2019     \l__stex_notation_prec_str
2020 }

```

```

2021
2022 \NewDocumentCommand \symdef { 0{} m } {
2023   \_stex_notation_symdef_args:n { #1 }
2024   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2025   \stex_symdecl_do:n { #2 }
2026   \exp_args:Nx \stex_notation_do:nn {
2027     \prop_item:Nn \l_tmpa_prop { module } ?
2028     \prop_item:Nn \l_tmpa_prop { name }
2029   }
2030 }

```

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

`\stex_invoke_symbol:n` Invokes a semantic macro

```

2031 %\cs_new_protected:Nn \stex_invoke_symbol:n {
2032 %  \peek_charcode_remove:NTF ! {
2033 %    \stex_term_custom:nn { #1 } { }
2034 %  } {
2035 %    \if_mode_math:
2036 %      \exp_after:wN \_stex_notation_invoke_math:n
2037 %    \else:
2038 %      \exp_after:wN \_stex_notation_invoke_text:n
2039 %    \fi: { #1 }
2040 %  }
2041 %}
2042
2043 \cs_new_protected:Nn \stex_invoke_symbol:n {
2044   \if_mode_math:
2045     \exp_after:wN \_stex_notation_invoke_math:n
2046   \else:
2047     \exp_after:wN \_stex_notation_invoke_text:n
2048   \fi: { #1 }
2049 }

```

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

`_stex_notation_invoke_math:n`

```

2050 \cs_new_protected:Nn \_stex_notation_invoke_math:n {
2051   \peek_charcode_remove:NTF ! {
2052     \peek_charcode:NTF [ {
2053       \_stex_notation_invoke_op:nw { #1 }
2054     }{
2055       \_stex_notation_invoke_op:nw { #1 } []
2056     }
2057   }{
2058     \peek_charcode_remove:NTF * {
2059       \_stex_notation_invoke_text:n { #1 }
2060     }{
2061       \peek_charcode:NTF [ {
2062         \_stex_notation_invoke_math:nw { #1 }
2063       }{
2064         \_stex_notation_invoke_math:nw { #1 } []
2065       }
2066     }
2067   }

```

```

2067 }
2068 }

```

(End definition for `_stex_notation_invoke_math:n`.)

`_stex_notation_invoke_op:nw`

```

2069 \cs_new_protected:Npn \_stex_notation_invoke_op:nw #1 [#2] {
2070   \_stex_notation_args:n { #2 }
2071   \cs_if_exist:cTF {
2072     stex_op_notation_ #1 \c_hash_str
2073     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str _cs
2074   }{
2075     \csname stex_op_notation_ #1 \c_hash_str
2076       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str _cs
2077     \endcsname
2078   }{
2079     % TODO throw error
2080   }
2081 }

```

(End definition for `_stex_notation_invoke_op:nw`.)

`_stex_notation_invoke_math:nw`

```

2082 \cs_new_protected:Npn \_stex_notation_invoke_math:nw #1 [#2] {
2083   \_stex_notation_args:n { #2 }
2084   \prop_set_eq:Nc \l_tmpa_prop {
2085     g_stex_symdecl_ #1 _prop
2086   }
2087   \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
2088   \seq_if_empty:NTF \l_tmpa_seq {
2089     \msg_set:nnn{stex}{error/nonotations}{
2090       Symbol~#1~used,~but~has~no~notations!
2091     }
2092     \msg_error:nn{stex}{error/nonotations}
2093   } {
2094     \seq_if_in:NxTF \l_tmpa_seq
2095     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2096       \use:c{
2097         stex_notation_ #1 \c_hash_str
2098         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2099         _cs
2100       }
2101     }{
2102       \str_if_empty:NTF \l__stex_notation_variant_str {
2103         \str_if_empty:NTF \l__stex_notation_lang_str {
2104           \seq_get_left:NN \l_tmpa_seq \l_tmpa_str
2105           \use:c{
2106             stex_notation_ #1 \c_hash_str \l_tmpa_str
2107             _cs
2108           }
2109         }{
2110           \msg_set:nnn{stex}{error/wrongnotation}{
2111             Symbol~#1~has~no~notation~
2112             \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2113           }

```

```

2114         \msg_error:nn{stex}{error/wrongnotation}
2115     }
2116 }{
2117     \msg_set:nnn{stex}{error/wrongnotation}{
2118         Symbol~#1~has~no~notation~
2119         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2120     }
2121     \msg_error:nn{stex}{error/wrongnotation}
2122 }
2123 }
2124 }
2125 }

```

(End definition for `__stex_notation_invoke_math:nw`.)

`_stex_notation_invoke_text:n`

```

2126 \cs_new_protected:Nn \__stex_notation_invoke_text:n {
2127     \peek_charcode_remove:NTF ! {
2128         \stex_term_custom:nn { #1 } { }
2129     }{
2130         \prop_set_eq:Nc \l_tmpa_prop {
2131             g_stex_symdecl_ #1 _prop
2132         }
2133         \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2134         \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2135     }
2136 }

```

(End definition for `__stex_notation_invoke_text:n`.)

4.8 Terms

2137 `<@@=stex_term>`

Precedences:

```

\infprec
\neginfprec
\l__stex_term_downprec
2138 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2139 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2140 \int_new:N \l__stex_term_downprec
2141 \int_set_eq:NN \l__stex_term_downprec \neginfprec

```

(End definition for `\infprec`, `\neginfprec`, and `\l__stex_term_downprec`. These variables are documented on page 23.)

Bracketing:

```

\l_stex_term_left_bracket_str
\l_stex_term_right_bracket_str
2142 \tl_set:Nn \l__stex_term_left_bracket_str (
2143 \tl_set:Nn \l__stex_term_right_bracket_str )

```

(End definition for `\l__stex_term_left_bracket_str` and `\l__stex_term_right_bracket_str`.)

`_stex_term_maybe_brackets:nn`

Compares precedences and insert brackets accordingly

```

2144 \cs_new_protected:Nn \_stex_term_maybe_brackets:nn {
2145     \int_compare:nNnTF { #1 } > \l__stex_term_downprec {
2146         \bool_if:NTF \l_stex_inarray_bool { #2 }{

```



```

2147     \dobrackets { #2 }
2148   }
2149   }{ #2 }
2150 }

```

(End definition for `_stex_term_maybe_brackets:nn`.)

`\dobrackets`

```

2151 %\RequirePackage{scalerel}
2152 \cs_new_protected:Npn \dobrackets #1 {
2153   %\ThisStyle{\if D\m@switch
2154   %   \exp_args:Nnx \use:nn
2155   %   { \exp_after:wN \left\l__stex_term_left_bracket_str #1 }
2156   %   { \exp_not:N\right\l__stex_term_right_bracket_str }
2157   % \else
2158   %   \exp_args:Nnx \use:nn
2159   %   { \l__stex_term_left_bracket_str #1 }
2160   %   { \l__stex_term_right_bracket_str }
2161   %\fi}
2162 }

```

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

`\withbrackets`

```

2163 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2164   \exp_args:Nnx \use:nn
2165   {
2166     \tl_set:Nx \l__stex_term_left_bracket_str { #1 }
2167     \tl_set:Nx \l__stex_term_right_bracket_str { #2 }
2168     #3
2169   }
2170   {
2171     \tl_set:Nn \exp_not:N \l__stex_term_left_bracket_str
2172     { \l__stex_term_left_bracket_str }
2173     \tl_set:Nn \exp_not:N \l__stex_term_right_bracket_str
2174     { \l__stex_term_right_bracket_str }
2175   }
2176 }

```

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

`\STEXinvisible`

```

2177 \cs_new_protected:Npn \STEXinvisible #1 {
2178   \stex_annotate_invisible:n { #1 }
2179 }

```

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

OMDOC terms:

`_stex_term_math_oms:nnnn`

```

2180 \cs_new_protected:Nn \_stex_term_oms:nnn {
2181   \stex_annotate:nnn{ OMID }{ #2 }{
2182     \stex_highlight_term:nn { #1 } { #3 }
2183   }
2184 }

```

```

2185
2186 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2187   \_stex_term_maybe_brackets:nn { #3 }{
2188     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2189   }
2190 }

```

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

`_stex_term_math_oma:nnnn`

```

2191 \cs_new_protected:Nn \_stex_term_oma:nnn {
2192   \stex_annotate:nnn{ OMA }{ #2 }{
2193     \stex_highlight_term:nn { #1 } { #3 }
2194   }
2195 }
2196
2197 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2198   \_stex_term_maybe_brackets:nn { #3 }{
2199     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2200   }
2201 }

```

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

`_stex_term_math_omb:nnnn`

```

2202 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2203   \stex_annotate:nnn{ OMBIND }{ #2 }{
2204     \stex_highlight_term:nn { #1 } { #3 }
2205   }
2206 }
2207
2208 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2209   \_stex_term_maybe_brackets:nn { #3 }{
2210     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2211   }
2212 }

```

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

`_stex_term_math_arg:nnn`

```

2213 \cs_new_protected:Nn \_stex_term_arg:nn {
2214   \stex_unhighlight_term:n {
2215     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2216   }
2217 }
2218 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
2219   \exp_args:Nnx \use:nn
2220     { \int_set:Nn \l__stex_term_downprec { #2 }
2221       \_stex_term_arg:nn { #1 }{ #3 }
2222     }
2223     { \int_set:Nn \exp_not:N \l__stex_term_downprec { \int_use:N \l__stex_term_downprec } }
2224 }

```

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

`\stex_term_math_assoc_arg:nnnn`

```

2225 \cs_new_protected:Nn \stex_term_math_assoc_arg:nnnn {
2226   \seq_set_split:Nnn \l_tmpa_seq , { #4 }
2227   \int_compare:nNnTF { \seq_count:N \l_tmpa_seq } < 2 {
2228     \tl_set:Nn \l_tmpa_tl { #4 }
2229   }{
2230     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
2231     \seq_reverse:N \l_tmpa_seq
2232     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_tl
2233     \tl_set:No \l_tmpa_tl { \l_tmpb_tl }
2234
2235     \seq_map_inline:Nn \l_tmpa_seq {
2236       \exp_args:NNo \tl_set:No \l_tmpa_tl {
2237         \exp_args:Nno
2238         \l_tmpa_cs { ##1 } \l_tmpa_tl
2239       }
2240     }
2241
2242   }
2243   \exp_args:Nnno
2244   \stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2245 }

```

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

`\stex_term_custom:nn`

```

2246 \cs_new_protected:Nn \stex_term_custom:nn {
2247   \str_set:Nn \l__stex_term_custom_uri { #1 }
2248   \str_set:Nn \l_tmpa_str { #2 }
2249   \tl_clear:N \l_tmpa_tl
2250   \int_zero:N \l_tmpa_int
2251   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2252   \__stex_term_custom_loop:
2253 }

```

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

`__stex_term_custom_loop:`

```

2254 \cs_new_protected:Nn \__stex_term_custom_loop: {
2255   \bool_set_false:N \l_tmpa_bool
2256   \bool_while_do:nn {
2257     \str_if_eq_p:ee X {
2258       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2259     }
2260   }{
2261     \int_incr:N \l_tmpa_int
2262   }
2263
2264   \peek_charcode:NTF [ {
2265     % notation/text component
2266     \__stex_term_custom_component:w
2267   } {
2268     \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2269       % all arguments read => finish

```

```

2270     \_stex_term_custom_final:
2271 } {
2272     % arguments missing
2273     \peek_charcode_remove:NTF * {
2274         % invisible, specific argument position or both
2275         \peek_charcode:NTF [ {
2276             % visible specific argument position
2277             \_stex_term_custom_arg:wn
2278         } {
2279             % invisible
2280             \peek_charcode_remove:NTF * {
2281                 % invisible specific argument position
2282                 \_stex_term_custom_arg_inv:wn
2283             } {
2284                 % invisible next argument
2285                 \_stex_term_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2286             }
2287         }
2288     } {
2289         % next normal argument
2290         \_stex_term_custom_arg:wn [ \l_tmpa_int + 1 ]
2291     }
2292 }
2293 }
2294 }

```

(End definition for _stex_term_custom_loop:.)

_stex_term_custom_arg_inv:wn

```

2295 \cs_new_protected:Npn \_stex_term_custom_arg_inv:wn [ #1 ] #2 {
2296     \bool_set_true:N \l_tmpa_bool
2297     \_stex_term_custom_arg:wn [ #1 ] { #2 }
2298 }

```

(End definition for _stex_term_custom_arg_inv:wn.)

_stex_term_custom_arg:wn

```

2299 \cs_new_protected:Npn \_stex_term_custom_arg:wn [ #1 ] #2 {
2300     \str_set:Nx \l_tmpb_str {
2301         \str_item:Nn \l_tmpa_str { #1 }
2302     }
2303     \str_case:VnTF \l_tmpb_str {
2304         { X } { } % TODO throw error ?
2305         { i } { \_stex_term_custom_set_X:n { #1 } }
2306         { b } { \_stex_term_custom_set_X:n { #1 } }
2307         { a } { \_stex_term_custom_set_X:n { #1 } } % TODO ?
2308         { B } { \_stex_term_custom_set_X:n { #1 } } % TODO ?
2309     }{}{
2310         % TODO throw error
2311     }
2312
2313     \bool_if:nTF \l_tmpa_bool {
2314         \tl_put_right:Nx \l_tmpa_tl {
2315             \stex_annotate_invisible:n {
2316                 \_stex_term_arg:nn { \int_eval:n { #1 } }

```

```

2317         \exp_not:n { { #2 } }
2318     }
2319 }
2320 } {
2321     \tl_put_right:Nx \l_tmpa_tl {
2322         \stex_term_arg:nn { \int_eval:n { #1 } }
2323         \exp_not:n { { #2 } }
2324     }
2325 }
2326
2327 \__stex_term_custom_loop:
2328 }

```

(End definition for __stex_term_custom_arg:wn.)

__stex_term_custom_set_X:n

```

2329 \cs_new_protected:Nn \__stex_term_custom_set_X:n {
2330     \str_set:Nx \l_tmpa_str {
2331         \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
2332         X
2333         \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
2334     }
2335 }

```

(End definition for __stex_term_custom_set_X:n.)

__stex_term_custom_component:

```

2336 \cs_new_protected:Npn \__stex_term_custom_component:w [ #1 ] {
2337     \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
2338     \__stex_term_custom_loop:
2339 }

```

(End definition for __stex_term_custom_component:.)

__stex_term_custom_final:

```

2340 \cs_new_protected:Nn \__stex_term_custom_final: {
2341     \int_compare:nNnTF \l_tmpb_int = 0 {
2342         \exp_args:Nnno \stex_term_oms:nnn
2343     }{
2344         \str_if_in:NnTF \l_tmpa_str {b} {
2345             \exp_args:Nnno \stex_term_ombind:nnn
2346         } {
2347             \exp_args:Nnno \stex_term_oma:nnn
2348         }
2349     }
2350     { \l__stex_term_custom_uri } { \l__stex_term_custom_uri } { \l_tmpa_tl }
2351 }

```

(End definition for __stex_term_custom_final:.)

\symref

\symname

```

2352 \NewDocumentCommand \symref { m m }{
2353     \STEXsymbol{#1}![#2]
2354 }
2355

```

```

2356 \keys_define:nn { stex / symname } {
2357   post      .tl_set_x:N    = \l_stex_symname_post_str
2358 }
2359
2360 \cs_new_protected:Nn \stex_symname_args:n {
2361   \str_clear:N \l_stex_symname_post_str
2362   \keys_set:nn { stex / symname } { #1 }
2363   \exp_args:NNNo \str_set:Nn \l_stex_symname_post_str
2364     \l_stex_symname_post_str
2365 }
2366
2367 \NewDocumentCommand \symname { 0{} m }{
2368   \stex_symname_args:n { #1 }
2369   \stex_get_symbol:n { #2 }
2370   \str_set:Nx \l_tmpa_str {
2371     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
2372   }
2373   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
2374   \exp_args:NNx \use:nn
2375   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }![
2376     \l_tmpa_str \l_stex_symname_post_str
2377   ] }
2378 }

```

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

4.9 Notation Components

```

2379 <@@=stex_notationcomps>

```

`\stex_highlight_term:nn`

```

2380 \latexml_if:F {
2381   \scalatex_if:F{
2382     \RequirePackage{pdfcomment}
2383   }
2384 }
2385
2386 \str_new:N \l__stex_notationcomps_highlight_uri_str
2387 \cs_new_protected:Nn \stex_highlight_term:nn {
2388   \exp_args:Nnx
2389   \use:nn {
2390     \str_set:Nx \l__stex_notationcomps_highlight_uri_str { #1 }
2391     #2
2392   } {
2393     \str_set:Nx \exp_not:N \l__stex_notationcomps_highlight_uri_str
2394       { \l__stex_notationcomps_highlight_uri_str }
2395   }
2396 }
2397
2398 \cs_new_protected:Nn \stex_unhighlight_term:n {
2399   % \latexml_if:TF {
2400   %   #1
2401   % } {
2402   %   \scalatex_if:TF {
2403   %     #1

```

```

2404 %   } {
2405     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
2406 %   }
2407 % }
2408 }

```

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

```

\comp
\@comp
\@defemph
2409 \cs_new_protected:Npn \comp #1 {
2410   \str_if_empty:NF \l__stex_notationcomps_highlight_uri_str {
2411     \scalatex_if:TF {
2412       \stex_annotate:nnn { comp }{ \l__stex_notationcomps_highlight_uri_str }{ #1 }
2413     }{
2414       \exp_args:Nnx \@comp { #1 } { \l__stex_notationcomps_highlight_uri_str }
2415     }
2416   }
2417 }
2418
2419 \cs_new_protected:Npn \@comp #1 #2 {
2420   \pdftooltip {
2421     \textcolor{blue}{#1}
2422   } { #2 }
2423 }
2424
2425 \cs_new_protected:Npn \@defemph #1 #2 {
2426   \pdftooltip {
2427     \textbf{\textcolor{magenta}{#1}}
2428   } { #2 }
2429 }

```

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

\ellipses

```

2430 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix
\parrayline
\parraycell
2431 \bool_new:N \l_stex_inparray_bool
2432 \bool_set_false:N \l_stex_inparray_bool
2433 \NewDocumentCommand \parray { m m } {
2434   \begingroup
2435   \bool_set_true:N \l_stex_inparray_bool
2436   \begin{array}{#1}
2437     #2
2438   \end{array}
2439   \endgroup
2440 }
2441
2442 \NewDocumentCommand \prmatrix { m } {
2443   \begingroup
2444   \bool_set_true:N \l_stex_inparray_bool
2445   \begin{matrix}
2446     #1

```

```

2447 \end{matrix}
2448 \endgroup
2449 }
2450
2451 \def \parrayline #1 #2 {
2452   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
2453 }
2454
2455 \def \parraycell #1 {
2456   #1 \bool_if:NT \l_stex_inarray_bool {&}
2457 }

```

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

4.10 Structural Features

```

2458 \@@=stex_features

```

symboldoc

```

2459 \NewDocumentEnvironment{symboldoc}{m}{
2460   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
2461   \seq_clear:N \l_tmpb_seq
2462   \seq_map_inline:Nn \l_tmpa_seq {
2463     \stex_get_symbol:n { ##1 }
2464     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
2465       \l_stex_get_symbol_uri_str
2466     }
2467   }
2468   \par
2469   \exp_args:Nnnx
2470   \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
2471 }{
2472   \end{stex_annotate_env}
2473 }

```

STEXdefinition

```

2474
2475 \NewDocumentCommand \__stex_features_definiendum:w { 0{} m m } {
2476   \stex_get_symbol:n { #2 }
2477   \scalatex_if:TF {
2478     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { #3 }
2479   } {
2480     \exp_args:Nnx \@defemph { #3 } { \l_stex_get_symbol_uri_str }
2481   }
2482 }
2483 \NewDocumentCommand \__stex_features_definame:w { 0{} m } {
2484   % TODO: root
2485   \stex_get_symbol:n { #2 }
2486   \str_set:Nx \l_tmpa_str {
2487     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
2488   }
2489   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
2490   \scalatex_if:TF {
2491     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {

```



```

2492     \l_tmpa_str
2493   }
2494 } {
2495   \@defemph {
2496     \l_tmpa_str
2497   } { \l_stex_get_symbol_uri_str }
2498 }
2499 }
2500
2501 \cs_new_protected:Nn \__stex_features_defi_begin:n {
2502   \let\definiendum\__stex_features_definiendum:w
2503   \let\definame\__stex_features_definame:w
2504   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
2505   \seq_clear:N \l_tmpb_seq
2506   \seq_map_inline:Nn \l_tmpa_seq {
2507     \stex_get_symbol:n { ##1 }
2508     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
2509       \l_stex_get_symbol_uri_str
2510     }
2511   }
2512   \exp_args:Nnnx
2513   \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
2514 }
2515
2516 \cs_new_protected:Nn \__stex_features_defi_end: {
2517   \end{stex_annotate_env}
2518 }
2519
2520 \NewDocumentEnvironment{STEXdefinition}{ m }{
2521   \__stex_features_defi_begin:n { #1 }
2522 }{
2523   \__stex_features_defi_end:
2524 }

```

\setSTEXdefinition

```

2525 \cs_new_protected:Npn \setSTEXdefinition #1 {
2526   \AddToHook{env/#1/before}[stex]{\__stex_features_defi_begin:n{}}
2527   \AddToHook{env/#1/after}[stex]{\__stex_features_defi_end:}
2528 }

```

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

structural@feature

```

2529
2530 \NewDocumentEnvironment{structural@feature}{ m m m }{
2531   \stex_if_in_module:F {
2532     \msg_set:nnn{stex}{error/nomodule}{
2533       Structural~Feature~has~to~occur~in~a~module:\\
2534       Feature~#2~of~type~#1\\
2535       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
2536     }
2537     \msg_error:nn{stex}{error/nomodule}
2538   }
2539 }

```

```

2540 \str_set:Nx \l_stex_module_name_str {
2541   \prop_item:Nn \l_stex_current_module_prop
2542     { name } / #2 - feature
2543 }
2544
2545
2546 \str_clear:N \l_tmpa_str
2547 \seq_clear:N \l_tmpa_seq
2548 \tl_clear:N \l_tmpa_tl
2549 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
2550   origname = #2,
2551   name      = \l_stex_module_name_str ,
2552   ns        = \l_stex_module_ns_str ,
2553   imports   = \exp_not:o { \l_tmpa_seq } ,
2554   constants = \exp_not:o { \l_tmpa_seq } ,
2555   content   = \exp_not:o { \l_tmpa_tl } ,
2556   file      = \exp_not:o { \g_stex_currentfile_seq } ,
2557   lang      = \l_stex_module_lang_str ,
2558   sig       = \l_tmpa_str ,
2559   meta      = \l_tmpa_str ,
2560   feature   = #1 ,
2561 }
2562
2563 \stex_if_smsmode:TF {
2564   \stex_smsmode_set_codes:
2565 } {
2566   \begin{stex_annotate_env}{ feature:#1 }{}
2567   \stex_annotate_invisible:nnn{header}{}{ #3 }
2568 }
2569 }{
2570   \str_set:Nx \l_tmpa_str {
2571     c_stex_feature_
2572     \prop_item:Nn \l_stex_current_module_prop { ns } ?
2573     \prop_item:Nn \l_stex_current_module_prop { name }
2574     _prop
2575   }
2576   \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
2577   \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
2578   \stex_if_smsmode:TF {
2579     \exp_args:Nx \stex_addtosms:n {
2580       \prop_gset_from_keyval:cn {
2581         c_stex_feature_
2582         \prop_item:Nn \l_stex_current_module_prop { ns } ?
2583         \prop_item:Nn \l_stex_current_module_prop { name }
2584         _prop
2585       } {
2586         origname = #2,
2587         name      = \prop_item:cn { \l_tmpa_str } { name } ,
2588         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
2589         imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
2590         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
2591         content   = \prop_item:cn { \l_tmpa_str } { content } ,
2592         file      = \prop_item:cn { \l_tmpa_str } { file } ,
2593         lang      = \prop_item:cn { \l_tmpa_str } { lang } ,

```

```

2594         sig      = \prop_item:cn { \l_tmpa_str } { sig } ,
2595         meta      = \prop_item:cn { \l_tmpa_str } { meta } ,
2596         feature    = \prop_item:cn { \l_tmpa_str } { feature }
2597     }
2598 }
2599 } {
2600     \end{stex_annotate_env}
2601 }
2602 }
2603

```

structure

```

2604
2605 \prop_new:N \l_stex_all_structures_prop
2606
2607 \keys_define:nn { stex / features / structure } {
2608     name          .tl_set_x:N = \l__stex_features_structure_name_str ,
2609 }
2610
2611 \cs_new_protected:Nn \__stex_features_structure_args:n {
2612     \str_clear:N \l__stex_features_structure_name_str
2613     \keys_set:nn { stex / features / structure } { #1 }
2614     \exp_args:NNo \str_set:Nn \l__stex_features_structure_name_str
2615         \l__stex_features_structure_name_str
2616 }
2617
2618 %\stex_new_feature:nnnn { structure } { 0{ } m } {
2619 % \__stex_features_structure_args:n { ##1 }
2620 % \str_if_empty:NT \l__stex_features_structure_name_str {
2621 %     \str_set:Nx \l__stex_features_structure_name_str { ##2 }
2622 % }
2623 %} {
2624 %
2625 %}
2626
2627 \NewDocumentEnvironment{structure}{ 0{ } m }{
2628     \__stex_features_structure_args:n { #1 }
2629     \str_if_empty:NT \l__stex_features_structure_name_str {
2630         \str_set:Nx \l__stex_features_structure_name_str { #2 }
2631     }
2632     \exp_args:Nnnx
2633     \begin{structural@feature}{ structure }
2634         { \l__stex_features_structure_name_str }{}
2635         \seq_clear:N \l_tmpa_seq
2636         \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
2637     }{
2638         \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
2639         \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
2640         \str_set:Nx \l_tmpa_str {
2641             \prop_item:Nn \l_stex_current_module_prop { ns } ?
2642             \prop_item:Nn \l_stex_current_module_prop { name }
2643         }
2644     }
2645     \seq_map_inline:Nn \l_tmpa_seq {

```

```

2646 \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
2647 }
2648 \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
2649 \exp_args:NNx
2650 \AddToHookNext { env / structure / after }{
2651 \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}}{
2652 \_stex_term_math_oms:nnnn { \l_tmpa_str }{}{0}{}
2653 }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }}{ #2 }
2654 \STEXexport {
2655 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2656 {\prop_item:Nn \l_stex_current_module_prop { origname }}
2657 {\l_tmpa_str}
2658 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2659 {#2}{\l_tmpa_str}
2660 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
2661 % \prop_item:Nn \l_stex_current_module_prop { origname },
2662 % \l_tmpa_str
2663 % }
2664 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
2665 % #2,\l_tmpa_str
2666 % }
2667 % \tl_set:cx { #2 } {
2668 % \stex_invoke_structure:n { \l_tmpa_str }
2669 % }
2670 }
2671
2672 \end{structural@feature}
2673 % \g_stex_last_feature_prop
2674 }

```

\instantiate

```

2675 \seq_new:N \l__stex_features_structure_field_seq
2676 \str_new:N \l__stex_features_structure_field_str
2677 \str_new:N \l__stex_features_structure_def_tl
2678 \prop_new:N \l__stex_features_structure_prop
2679 \NewDocumentCommand \instantiate { m O{} m }{
2680 \stex_smsmode_set_codes:
2681 \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
2682 \prop_set_eq:Nc \l__stex_features_structure_prop {
2683 c_stex_feature_\l_tmpa_str _prop
2684 }
2685 \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
2686 \seq_map_inline:Nn \l__stex_features_structure_field_seq {
2687 \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
2688 \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
2689 \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
2690 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
2691 {!} \l_tmpa_tl
2692 \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
2693 \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
2694 \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
2695 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
2696 }{
2697 \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl

```

```

2698         \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
2699         \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
2700         \l_tmpa_tl
2701         \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
2702             \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
2703             \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
2704         }{
2705             \tl_clear:N \l_tmpb_tl
2706         }
2707     }
2708 }{
2709     \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
2710     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
2711         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
2712         \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
2713         \tl_clear:N \l_tmpa_tl
2714     }{
2715         % TODO throw error
2716     }
2717 }
2718 % \l_tmpa_str: name
2719 % \l_tmpa_tl: definiens
2720 % \l_tmpb_tl: notation
2721 \tl_if_empty:NT \l__stex_features_structure_field_str {
2722     % TODO throw error
2723 }
2724 \str_clear:N \l_tmpb_str
2725
2726 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
2727 \seq_map_inline:Nn \l_tmpa_seq {
2728     \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
2729     \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
2730     \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
2731         \seq_map_break:n {
2732             \str_set:Nn \l_tmpb_str { ####1 }
2733         }
2734     }
2735 }
2736 \prop_get:cnN { g_stex_symdecl_ \l_tmpb_str _prop } {args}
2737 \l_tmpb_str
2738
2739 \tl_if_empty:NNTF \l_tmpb_tl {
2740     \tl_if_empty:NF \l_tmpa_tl {
2741         \exp_args:Nx \use:n {
2742             \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fe
2743         }
2744     }
2745 }{
2746     \tl_if_empty:NNTF \l_tmpa_tl {
2747         \exp_args:Nx \use:n {
2748             \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\
2749         }
2750     }
2751 }{

```

```

2752         \exp_args:Nx \use:n {
2753             \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
2754             \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
2755         }
2756     }
2757 }
2758 % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
2759 % \prop_item:Nn \l_stex_current_module_prop {name} ?
2760 % #3/\l__stex_features_structure_field_str
2761 % \par
2762 % \expandafter\present\csname
2763 %     g_stex_symdecl_
2764 % \prop_item:Nn \l_stex_current_module_prop {ns} ?
2765 % \prop_item:Nn \l_stex_current_module_prop {name} ?
2766 % #3/\l__stex_features_structure_field_str
2767 % _prop
2768 % \endcsname
2769 }
2770
2771 \tl_clear:N \l__stex_features_structure_def_tl
2772
2773 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
2774 \seq_map_inline:Nn \l_tmpa_seq {
2775     \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
2776     \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
2777     \exp_args:Nx \use:n {
2778         \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
2779
2780         }
2781     }
2782
2783     \prop_if_exist:cF {
2784         g_stex_symdecl_
2785         \prop_item:Nn \l_stex_current_module_prop {ns} ?
2786         \prop_item:Nn \l_stex_current_module_prop {name} ?
2787         #3/\l_tmpa_str
2788         _prop
2789     }{
2790         \prop_get:cnN { g_stex_symdecl_ ##1 _prop } {args}
2791         \l_tmpb_str
2792         \exp_args:Nx \use:n {
2793             \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
2794         }
2795     }
2796 }
2797
2798 \symdecl*[type={\STEXsymbol{module-type}}{
2799     \_stex_term_math_oms:nnnn {
2800         \prop_item:Nn \l__stex_features_structure_prop {ns} ?
2801         \prop_item:Nn \l__stex_features_structure_prop {name}
2802     }{}{0}{}
2803 }{}]{#3}
2804
2805 % TODO: -> sms file

```

```

2806
2807 \tl_set:cx{ #3 }{
2808   \stex_invoke_structure:nnn {
2809     \prop_item:Nn \l_stex_current_module_prop {ns} ?
2810     \prop_item:Nn \l_stex_current_module_prop {name} ? #3
2811   } {
2812     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
2813     \prop_item:Nn \l__stex_features_structure_prop {name}
2814   }
2815 }
2816
2817 }

```

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

\stex_invoke_structure:nnn

```

2818 % #1: URI of the instance
2819 % #2: URI of the instantiated module
2820 \cs_new_protected:Nn \stex_invoke_structure:nnn {
2821   \tl_if_empty:nTF{ #3 }{
2822     \prop_set_eq:Nc \l__stex_features_structure_prop {
2823       c_stex_feature_ #2 _prop
2824     }
2825     \tl_clear:N \l_tmpa_tl
2826     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
2827     \seq_map_inline:Nn \l_tmpa_seq {
2828       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
2829       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
2830       \cs_if_exist:cT {
2831         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
2832       }{
2833         \tl_if_empty:NF \l_tmpa_tl {
2834           \tl_put_right:Nn \l_tmpa_tl {,}
2835         }
2836         \tl_put_right:Nx \l_tmpa_tl {
2837           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
2838         }
2839       }
2840     }
2841     \scalatexBREAK
2842     \exp_args:No \mathstrut \l_tmpa_tl
2843   }{
2844     \stex_invoke_symbol:n{#1/#3}
2845   }
2846 }

```

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

4.11 Put these somewhere

\MSC

```

2847 \NewDocumentCommand \MSC {m} {
2848   % TODO
2849 }

```

```

2850 \@ifpackageloaded{tikzinput}{
2851   \RequirePackage{stex-tikzinput}
2852 }{}
2853
2854 \AddToHook{begindocument}{
2855   \input{stex-metatheory}
2856 }
2857 \end{package}

```

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

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

88


```

2890 %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
2891 %\notation[lambda]{mapto}{\comp\lambda #1 \comp.\; #2}{#1 \comp, #2}
2892 %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp.\; #2}{#1 \comp, #2}
2893
2894 % function/operator application
2895 \symdecl[args=ia]{apply}
2896 \notation[prec=0;0x\neginfprec,parens]{apply}{#1 \comp( #2 \comp)}{#1 \comp, #2}
2897 \notation[prec=0;0x\neginfprec,lambda]{apply}{#1 \; #2 }{#1 \; #2}
2898
2899 % ‘‘type’’ of all collections (sets, classes, types, kinds)
2900 \symdecl{collection}
2901 \notation[U]{collection}{\comp{\mathcal{U}}}
2902 \notation[set]{collection}{\comp{\textsf{Set}}}
2903
2904 % sequences
2905 \symdecl[args=1]{seqtype}
2906 \notation[kleene]{seqtype}{#1^{\comp\ast}}
2907
2908 \symdef[args=2,li]{sequence-index}{#1_{#2}}
2909 \notation[ui]{sequence-index}{#1^{#2}}
2910
2911 %\symdef[args=3,li]{sequence-from-to}{#1_{#2}\comp{\,\ellipses},#1_{#3}}
2912 %\notation[ui]{sequence-from-to}{#1^{#2}\comp{\,\ellipses},#1^{#3}}
2913 % ^ superceded by \aseqfromto and \livar/\uivar
2914
2915 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{\,\ellipses}}{#1\comp,#2}
2916 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{\,\ellipses\comp,}#2 }{#1\comp,#2}
2917
2918 % letin (‘‘let’’, local definitions, variable substitution)
2919 \symdecl[args=bii]{letin}
2920 \notation[let]{letin}{\comp{\rm let}}\;#1\comp{=}\;#2\; \comp{\rm in}}\;#3}
2921 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
2922 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
2923
2924 % structures
2925 \symdecl*[args=1]{module-type}
2926 \notation{module-type}{\mathtt{MOD} #1}
2927 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
2928 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{#1 \comp, #2}
2929
2930 \STEXexport{
2931   \let\nappa\apply
2932   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
2933   \def\livar{\csname sequence-index\endcsname[li]}
2934   \def\uivar{\csname sequence-index\endcsname[ui]}
2935   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
2936   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
2937 }
2938
2939 \end{@module}
2940 \ExplSyntaxOff
2941 </metatheory>

```

4.13 Auxiliary Packages

4.13.1 tikzinput

```
2942 <*tikzinput>
2943 <@@=tikzinput>
2944 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
2945 \RequirePackage{l3keys2e}
2946
2947 \keys_define:nn { tikzinput } {
2948   image .bool_set:N = \c_tikzinput_image_bool
2949 }
2950
2951 \ProcessKeysOptions { tikzinput }
2952
2953 \bool_if:NTF \c_tikzinput_image_bool {
2954   \RequirePackage{graphicx}
2955
2956   \providecommand\usetikzlibrary[]{}
2957   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
2958 }{
2959   \RequirePackage{tikz}
2960   \RequirePackage{standalone}
2961
2962   \newcommand \tikzinput [2] [] {
2963     \setkeys{Gin}{#1}
2964     \ifx \Gin@width \Gin@exclamation
2965       \ifx \Gin@height \Gin@exclamation
2966         \input { #2 }
2967       \else
2968         \resizebox{!}{ \Gin@height }{
2969           \input { #2 }
2970         }
2971       \fi
2972     \else
2973       \ifx \Gin@height \Gin@exclamation
2974         \resizebox{ \Gin@width }{!}{
2975           \input { #2 }
2976         }
2977       \else
2978         \resizebox{ \Gin@width }{ \Gin@height }{
2979           \input { #2 }
2980         }
2981       \fi
2982     \fi
2983   }
2984 }
2985
2986 \newcommand \ctikzinput [2] [] {
2987   \begin{center}
2988     \tikzinput [#1] {#2}
2989   \end{center}
2990 }
2991
2992 \@ifpackageloaded{stex}{
```

```

2993 \RequirePackage{stex-tikzinput}
2994 }{}
2995 </tikzinput>
2996 <*stex-tikzinput>
2997 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
2998 \RequirePackage{stex}
2999 \RequirePackage{tikzinput}
3000
3001 % TODO
3002
3003 </stex-tikzinput>

```

4.13.2 sTeX1 Compatibility

```

3004 <*smglom>
3005 \RequirePackage{expl3,l3keys2e}
3006 \ProvidesExplClass{smglom}{2021/08/01}{1.9}{sTeX1 compatibility}
3007 \LoadClass[border=1px,varwidth]{standalone}
3008 \setlength\textwidth{15cm}
3009 %\g@addto@macro{\@parboxrestore}{\setlength\parskip{\baselineskip}}
3010 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
3011 \ProcessOptions
3012
3013 \RequirePackage{stex-compatibility}
3014 </smglom>
3015
3016 <*compat>
3017 <@@=stex_deprec>
3018 \ProvidesExplPackage{stex-compatibility}{2021/08/01}{1.9}{bla}
3019 \RequirePackage[lang={de,en,ro,tr,fr}]{stex}
3020
3021 \NewDocumentEnvironment { mhmodnl } { 0{} m m } {
3022   \msg_set:nnn{stex}{warning/deprecated}{
3023     \\\
3024     Environment~mhmodnl~is~deprected! \\\
3025     Please~update~module~#2~in~file~
3026     \stex_path_to_string:N \g_stex_currentfile_seq!
3027     \\\ \\\
3028   }
3029   \msg_warning:nn{stex}{warning/deprecated}
3030
3031   \begin{module}[#1,lang=#3]{#2}
3032     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
3033     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
3034     \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
3035     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
3036     \input { \stex_path_to_string:N \l_tmpa_seq / \l_tmpa_str.tex }
3037   } {
3038     \end{module}
3039   }
3040
3041 \NewDocumentEnvironment { modsig } { 0{} m } {
3042   \stex_if_in_module:TF {
3043     \prop_get:NnN \l_stex_current_module_prop {name} \l_tmpa_str
3044     \str_set:Nn \l_tmpb_str { #2 }

```

```

3045 \str_if_eq:NNTF \l_tmpa_str \l_tmpb_str {
3046 \prop_set_eq:NN \l_modsig_old_module_prop \l_stex_current_module_prop
3047 \begin{@module}{modsig-#2}
3048 % \prop_set_eq:NN \l_stex_current_module_prop \l_modsig_old_module_prop
3049 } {
3050 \begin{@module}{#2}
3051 }
3052 } {
3053 \begin{@module}{#2}
3054 }
3055 }{
3056 \end{@module}
3057 \AddToHookNext { env / modsig / after }{
3058 \stex_if_in_module:T {
3059 \prop_get:NnN \l_stex_current_module_prop {name} \l_tmpa_str
3060 \str_set:Nn \l_tmpb_str { #2 }
3061 \str_if_eq:NNT \l_tmpa_str \l_tmpb_str {
3062 % \xdef \g_stex_module_after_group_tl {
3063 \stex_if_smsmode:TF {
3064 \exp_args:Nx
3065 \stex_add_to_current_module:n {
3066 \stex_debug:n{Activating~signature~of~#2}
3067 \exp_not:N \prop_item:cn { c_stex_module_
3068 \prop_item:Nn \l_stex_current_module_prop {ns} ?
3069 \prop_item:Nn \l_stex_current_module_prop {name}
3070 / modsig-#2_prop } { content }
3071 }
3072 }
3073 {
3074 \gdef \g_stex_modsig_after_group_tl {
3075 \stex_activate_module:n {
3076 \prop_item:Nn \l_stex_current_module_prop {ns} ?
3077 \prop_item:Nn \l_stex_current_module_prop {name}
3078 / modsig-#2
3079 }
3080
3081 \exp_args:Nx
3082 \stex_add_to_current_module:n {
3083 \stex_activate_module:n {
3084 \prop_item:Nn \l_stex_current_module_prop {ns} ?
3085 \prop_item:Nn \l_stex_current_module_prop {name}
3086 / modsig-#2
3087 }
3088 }
3089 }
3090 \aftergroup \g_stex_modsig_after_group_tl
3091 }
3092 }
3093 }
3094 }
3095 }
3096
3097 \cs_new_protected:Npn \gimport {
3098 \peek_charcode_remove:NTF * {

```

```

3099     \gimport_do:
3100   } {
3101     \gimport_do:
3102   }
3103 }
3104
3105 \NewDocumentCommand \gimport_do: { 0{} m } {
3106   \msg_set:nnn{stex}{warning/deprecated}{
3107     \\\
3108     \c_backslash_str gimport~is~deprecated! \\\
3109     Please~use~\c_backslash_str importmodule[#1]{#2}~instead!~(in~file~
3110     \stex_path_to_string:N \g_stex_currentfile_seq)
3111     \\\ \\\
3112   }
3113   \msg_warning:nn{stex}{warning/deprecated}
3114   \importmodule[#1]{#2}
3115 }
3116
3117 \cs_new_protected:Npn \guse {
3118   \peek_charcode_remove:NTF * {
3119     \guse_do:
3120   } {
3121     \guse_do:
3122   }
3123 }
3124
3125 \NewDocumentCommand \guse_do: { 0{} m } {
3126   \msg_set:nnn{stex}{warning/deprecated}{
3127     \\\
3128     \c_backslash_str guse~is~deprecated! \\\
3129     Please~use~\c_backslash_str usemodule[#1]{#2}~instead!~(in~file~
3130     \stex_path_to_string:N \g_stex_currentfile_seq)
3131     \\\ \\\
3132   }
3133   \msg_warning:nn{stex}{warning/deprecated}
3134   \usemodule[#1]{#2}
3135 }
3136
3137 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3138
3139 \cs_new_protected:Npn \symi {
3140   \peek_charcode_remove:NTF * {
3141     \symi_do:
3142   } {
3143     \symi_do:
3144   }
3145 }
3146
3147 \NewDocumentCommand \symi_do: { 0{} m } {
3148   \msg_set:nnn{stex}{warning/deprecated}{
3149     \\\
3150     \c_backslash_str symi~is~deprecated! \\\
3151     Please~use~\c_backslash_str symdecl[#1]{#2}~instead!~(in~file~
3152     \stex_path_to_string:N \g_stex_currentfile_seq)

```

```

3153     \\\ \\\
3154 }
3155 \msg_warning:nn{stex}{warning/deprecated}
3156 \symdecl*{#1}{#2}
3157 }
3158
3159 \cs_new_protected:Npn \symii {
3160   \peek_charcode_remove:NTF * {
3161     \symii_do:
3162   } {
3163     \symii_do:
3164   }
3165 }
3166
3167 \NewDocumentCommand \symii_do: { 0{} m m } {
3168   \msg_set:nnn{stex}{warning/deprecated}{
3169     \\\
3170     \c_backslash_str symii~is~deprecated! \\\
3171     Please-use~\c_backslash_str symdecl{#1}{#2-#3}~instead!~(in-file~
3172     \stex_path_to_string:N \g_stex_currentfile_seq)
3173     \\\ \\\
3174   }
3175   \msg_warning:nn{stex}{warning/deprecated}
3176   \symdecl*{#1}{#2-#3}
3177 }
3178
3179 \cs_new_protected:Npn \symiii {
3180   \peek_charcode_remove:NTF * {
3181     \symiii_do:
3182   } {
3183     \symiii_do:
3184   }
3185 }
3186
3187 \NewDocumentCommand \symiii_do: { 0{} m m m } {
3188   \msg_set:nnn{stex}{warning/deprecated}{
3189     \\\
3190     \c_backslash_str symiii~is~deprecated! \\\
3191     Please-use~\c_backslash_str symdecl{#1}{#2-#3-#4}~instead!~(in-file~
3192     \stex_path_to_string:N \g_stex_currentfile_seq)
3193     \\\ \\\
3194   }
3195   \msg_warning:nn{stex}{warning/deprecated}
3196   \symdecl*{#1}{#2-#3-#4}
3197 }
3198
3199 \keys_define:nn { stex / deprec / defi } {
3200   name .tl_set_x:N = \l_tmpa_str
3201 }
3202
3203 \cs_new_protected:Npn \defi {
3204   \peek_charcode_remove:NTF * {
3205     \defi_do:
3206   } {

```

```

3207     \defi_do:
3208   }
3209 }
3210
3211 \NewDocumentCommand \defi_do: { 0{ } m } {
3212   \str_clear:N \l_tmpa_str
3213   \keys_set:nn { stex / deprec / defi } { #1 }
3214
3215   \str_if_empty:NTF \l_tmpa_str {
3216     \msg_set:nnn{stex}{warning/deprecated}{
3217       \\\
3218       \c_backslash_str defi-is~deprecated! \\\
3219       Please~use~\c_backslash_str STEXsymbol{#2}![#2]~instead!~(in~file~
3220       \stex_path_to_string:N \g_stex_currentfile_seq)
3221       \\\ \\\
3222     }
3223     \msg_warning:nn{stex}{warning/deprecated}
3224     \STEXsymbol { #2 }![ \comp{#2} ]
3225   } {
3226     \msg_set:nnn{stex}{warning/deprecated}{
3227       \\\
3228       \c_backslash_str defi-is~deprecated! \\\
3229       Please~use~\c_backslash_str STEXsymbol { \l_tmpa_str }[ #2 ]~instead!~(in~file~
3230       \stex_path_to_string:N \g_stex_currentfile_seq)
3231       \\\ \\\
3232     }
3233     \msg_warning:nn{stex}{warning/deprecated}
3234     \exp_args:No \STEXsymbol { \l_tmpa_str }![ \comp{#2} ]
3235   }
3236 }
3237
3238
3239 \cs_new_protected:Npn \Defi {
3240   \peek_charcode_remove:NTF * {
3241     \Defi_do:
3242   } {
3243     \Defi_do:
3244   }
3245 }
3246
3247 \NewDocumentCommand \Defi_do: { 0{ } m } {
3248   \str_clear:N \l_tmpa_str
3249   \keys_set:nn { stex / deprec / defi } { #1 }
3250
3251   \str_if_empty:NTF \l_tmpa_str {
3252     \msg_set:nnn{stex}{warning/deprecated}{
3253       \\\
3254       \c_backslash_str Defi-is~deprecated! \\\
3255       Please~use~\c_backslash_str STEXsymbol{#2}![\exp_after:wN \stex_capitalize:n #2]~inste
3256       \stex_path_to_string:N \g_stex_currentfile_seq)
3257       \\\ \\\
3258     }
3259     \msg_warning:nn{stex}{warning/deprecated}
3260     \STEXsymbol { #2 }![ \comp{\exp_after:wN \stex_capitalize:n #2} ]

```

```

3261 } {
3262   \msg_set:nnn{stex}{warning/deprecated}{
3263     \\\
3264     \c_backslash_str Defi~is~deprecated! \\\
3265     Please~use~\c_backslash_str STEXsymbol { \l_tmpa_str }[ \exp_after:wN \stex_capitalize
3266     \stex_path_to_string:N \g_stex_currentfile_seq)
3267     \\\ \\\
3268   }
3269   \msg_warning:nn{stex}{warning/deprecated}
3270   \exp_args:No \STEXsymbol { \l_tmpa_str }![ \comp{\exp_after:wN \stex_capitalize:n #2} ]
3271 }
3272 }
3273
3274 \cs_new_protected:Npn \adefi {
3275   \peek_charcode_remove:NTF * {
3276     \adefi_do:
3277   } {
3278     \adefi_do:
3279   }
3280 }
3281
3282 \NewDocumentCommand \adefi_do: { 0{} m m } {
3283   \str_clear:N \l_tmpa_str
3284   \keys_set:nn { stex / deprec / defi } { #1 }
3285
3286   \str_if_empty:NTF \l_tmpa_str {
3287     \msg_set:nnn{stex}{warning/deprecated}{
3288       \\\
3289       \c_backslash_str adefi~is~deprecated! \\\
3290       Please~use~\c_backslash_str STEXsymbol{#3}![#2]~instead!~(in~file~
3291       \stex_path_to_string:N \g_stex_currentfile_seq)
3292       \\\ \\\
3293     }
3294     \msg_warning:nn{stex}{warning/deprecated}
3295     \STEXsymbol { #3 }![ \comp{#2} ]
3296   } {
3297     \msg_set:nnn{stex}{warning/deprecated}{
3298       \\\
3299       \c_backslash_str adefi~is~deprecated! \\\
3300       Please~use~\c_backslash_str STEXsymbol { \l_tmpa_str }[ #2 ]~instead!~(in~file~
3301       \stex_path_to_string:N \g_stex_currentfile_seq)
3302       \\\ \\\
3303     }
3304     \msg_warning:nn{stex}{warning/deprecated}
3305     \exp_args:No \STEXsymbol { \l_tmpa_str }![ \comp{#2} ]
3306   }
3307 }
3308
3309 \cs_new_protected:Npn \defis {
3310   \peek_charcode_remove:NTF * {
3311     \defis_do:
3312   } {
3313     \defis_do:
3314   }

```



```

3315 }
3316
3317 \NewDocumentCommand \defis_do: { 0{} m } {
3318   \str_clear:N \l_tmpa_str
3319   \keys_set:nn { stex / deprec / defi } { #1 }
3320
3321   \str_if_empty:NTF \l_tmpa_str {
3322     \msg_set:nnn{stex}{warning/deprecated}{
3323       \\\
3324       \c_backslash_str defis-is-deprecated! \\\
3325       Please~use~\c_backslash_str STExsymbol{#2}![#2s]~instead!~(in~file~
3326       \stex_path_to_string:N \g_stex_currentfile_seq)
3327       \\\ \\\
3328     }
3329     \msg_warning:nn{stex}{warning/deprecated}
3330     \STExsymbol { #2 }![ \comp{#2s} ]
3331   } {
3332     \msg_set:nnn{stex}{warning/deprecated}{
3333       \\\
3334       \c_backslash_str defis-is-deprecated! \\\
3335       Please~use~\c_backslash_str STExsymbol { \l_tmpa_str }[ #2s ]~instead!~(in~file~
3336       \stex_path_to_string:N \g_stex_currentfile_seq)
3337       \\\ \\\
3338     }
3339     \msg_warning:nn{stex}{warning/deprecated}
3340     \exp_args:No \STExsymbol { \l_tmpa_str }![ \comp{#2s} ]
3341   }
3342 }
3343
3344 \cs_new_protected:Npn \defii {
3345   \peek_charcode_remove:NTF * {
3346     \defii_do:
3347   } {
3348     \defii_do:
3349   }
3350 }
3351
3352 \NewDocumentCommand \defii_do: { 0{} m m } {
3353   \str_clear:N \l_tmpa_str
3354   \keys_set:nn { stex / deprec / defi } { #1 }
3355   \str_if_empty:NTF \l_tmpa_str {
3356     \msg_set:nnn{stex}{warning/deprecated}{
3357       \\\
3358       \c_backslash_str defii-is-deprecated! \\\
3359       Please~use~\c_backslash_str STExsymbol{#2-#3}![#2~#3]~instead!~(in~file~
3360       \stex_path_to_string:N \g_stex_currentfile_seq)
3361       \\\ \\\
3362     }
3363     \msg_warning:nn{stex}{warning/deprecated}
3364     \STExsymbol { #2-#3 }![ \comp{#2~#3} ]
3365   } {
3366     \msg_set:nnn{stex}{warning/deprecated}{
3367       \\\
3368       \c_backslash_str defii-is-deprecated! \\\

```

```

3369     Please~use~\c_backslash_str STExsymbol { \l_tmpa_str }[ #2~#3 ]~instead!~(in~file~
3370     \stex_path_to_string:N \g_stex_currentfile_seq)
3371     \\\ \\\
3372   }
3373   \msg_warning:nn{stex}{warning/deprecated}
3374   \exp_args:No \STExsymbol { \l_tmpa_str }![ \comp{#2~#3} ]
3375 }
3376 }
3377
3378
3379 \cs_new_protected:Npn \defiis {
3380   \peek_charcode_remove:NTF * {
3381     \defiis_do:
3382   } {
3383     \defiis_do:
3384   }
3385 }
3386
3387 \NewDocumentCommand \defiis_do: { O{} m m } {
3388   \str_clear:N \l_tmpa_str
3389   \keys_set:nn { stex / deprec / defi } { #1 }
3390   \str_if_empty:NTF \l_tmpa_str {
3391     \msg_set:nnn{stex}{warning/deprecated}{
3392       \\\
3393       \c_backslash_str defiis~is~deprecated! \\\
3394       Please~use~\c_backslash_str STExsymbol{#2~#3}![#2~#3s]~instead!~(in~file~
3395       \stex_path_to_string:N \g_stex_currentfile_seq)
3396       \\\ \\\
3397     }
3398     \msg_warning:nn{stex}{warning/deprecated}
3399     \STExsymbol { #2~#3 }![ \comp{#2~#3s} ]
3400   } {
3401     \msg_set:nnn{stex}{warning/deprecated}{
3402       \\\
3403       \c_backslash_str defiis~is~deprecated! \\\
3404       Please~use~\c_backslash_str STExsymbol { \l_tmpa_str }[ #2~#3s ]~instead!~(in~file~
3405       \stex_path_to_string:N \g_stex_currentfile_seq)
3406       \\\ \\\
3407     }
3408     \msg_warning:nn{stex}{warning/deprecated}
3409     \exp_args:No \STExsymbol { \l_tmpa_str }![ \comp{#2~#3s} ]
3410   }
3411 }
3412
3413
3414 \cs_new_protected:Npn \defiii {
3415   \peek_charcode_remove:NTF * {
3416     \defiii_do:
3417   } {
3418     \defiii_do:
3419   }
3420 }
3421
3422 \NewDocumentCommand \defiii_do: { O{} m m m } {

```

```

3423 \str_clear:N \l_tmpa_str
3424 \keys_set:nn { stex / deprec / defi } { #1 }
3425 \str_if_empty:NTF \l_tmpa_str {
3426   \msg_set:nnn{stex}{warning/deprecated}{
3427     \\\
3428     \c_backslash_str defiii~is-deprecated! \\\
3429     Please~use~\c_backslash_str STExsymbol{#2~#3~#4}![#2~#3~#4]~instead!~(in~file~
3430     \stex_path_to_string:N \g_stex_currentfile_seq)
3431     \\\ \\\
3432   }
3433   \msg_warning:nn{stex}{warning/deprecated}
3434   \STExsymbol { #2~#3~#4 }![ \comp{#2~#3~#4} ]
3435 } {
3436   \msg_set:nnn{stex}{warning/deprecated}{
3437     \\\
3438     \c_backslash_str defiii~is-deprecated! \\\
3439     Please~use~\c_backslash_str STExsymbol { \l_tmpa_str }[ #2~#3~#4 ]~instead!~(in~file~
3440     \stex_path_to_string:N \g_stex_currentfile_seq)
3441     \\\ \\\
3442   }
3443   \msg_warning:nn{stex}{warning/deprecated}
3444   \exp_args:No \STExsymbol { \l_tmpa_str }![ \comp{#2~#3~#4} ]
3445 }
3446 }
3447
3448 %\RequirePackage[hyperref]{ntheorem}
3449 %\theoremstyle{plain}
3450 %\RequirePackage{amsthm}
3451
3452 \NewDocumentEnvironment {definition} { 0{} } {
3453   \begin{STExdefinition}{}
3454 }{
3455   \end{STExdefinition}
3456 }
3457 \keys_define:nn { stex / omtex } {
3458   title .tl_set_x:N = \l_stex_omtext_title_str
3459 }
3460 \cs_new_protected:Nn \stex_omtext_args:n {
3461   \str_clear:N \l_stex_omtext_title_str
3462   \keys_set:nn { stex / omtex } { #1 }
3463   \exp_args:NNo \str_set:Nn \l_stex_omtext_title_str
3464     \l_stex_omtext_title_str
3465 }
3466 \NewDocumentEnvironment {omtext} { 0{} } {
3467   \stex_omtext_args:n { #1 }
3468   \paragraph{\l_stex_omtext_title_str}
3469 }{
3470
3471 }
3472 \NewDocumentEnvironment {assertion} { 0{} } {
3473
3474 }{
3475
3476 }

```

```

3477
3478 \NewDocumentCommand \inlinedef { m } {
3479   \begingroup
3480   \let\definiendum\_stex_deprec_definiendum:w
3481   \let\definame\_stex_deprec_definame:w
3482   #1
3483   \endgroup
3484 }
3485
3486 \NewDocumentCommand \inlineass { m } { #1 }
3487
3488 \NewDocumentCommand \trefi { O{} m } {
3489   \str_set:Nn \l_tmpa_str { #1 }
3490   \str_if_empty:NTF \l_tmpa_str {
3491     \msg_set:nnn{stex}{warning/deprecated}{
3492       \\\
3493       \c_backslash_str trefi-is-deprecated! \\\
3494       Please~use~\c_backslash_str STEXsymbol{#2}![#2]~instead!~(in~file~
3495       \stex_path_to_string:N \g_stex_currentfile_seq)
3496       \\\
3497     }
3498     \msg_warning:nn{stex}{warning/deprecated}
3499     \STEXsymbol { #2 }![ \comp{#2} ]
3500   } {
3501     \msg_set:nnn{stex}{warning/deprecated}{
3502       \\\
3503       \c_backslash_str trefi-is-deprecated! \\\
3504       Please~use~\c_backslash_str STEXsymbol { #1?#2 }[ #2 ]~instead!~(in~file~
3505       \stex_path_to_string:N \g_stex_currentfile_seq)
3506       \\\
3507     }
3508     \msg_warning:nn{stex}{warning/deprecated}
3509     \STEXsymbol { #1 }![ \comp{#2} ]
3510   }
3511 }
3512
3513
3514 \NewDocumentCommand \Trefi { O{} m } {
3515   \str_set:Nn \l_tmpa_str { #1 }
3516   \str_if_empty:NTF \l_tmpa_str {
3517     \msg_set:nnn{stex}{warning/deprecated}{
3518       \\\
3519       \c_backslash_str Trefi-is-deprecated! \\\
3520       Please~use~\c_backslash_str STEXsymbol{#2}![\exp_after:wN \stex_capitalize:n #2]~inste
3521       \stex_path_to_string:N \g_stex_currentfile_seq)
3522       \\\
3523     }
3524     \msg_warning:nn{stex}{warning/deprecated}
3525     \STEXsymbol { #2 }![ \comp{\exp_after:wN \stex_capitalize:n #2} ]
3526   } {
3527     \msg_set:nnn{stex}{warning/deprecated}{
3528       \\\
3529       \c_backslash_str Trefi-is-deprecated! \\\
3530       Please~use~\c_backslash_str STEXsymbol { #1 }[ \exp_after:wN \stex_capitalize:n #2 ]~i

```

```

3531     \stex_path_to_string:N \g_stex_currentfile_seq)
3532     \\\ \\\
3533   }
3534   \msg_warning:nn{stex}{warning/deprecated}
3535   \STEXsymbol { #1 }![ \comp{\exp_after:wN \stex_capitalize:n #2} ]
3536 }
3537 }
3538
3539 \NewDocumentCommand \trefis { O{} m } {
3540   \str_set:Nn \l_tmpa_str { #1 }
3541   \str_if_empty:NTF \l_tmpa_str {
3542     \msg_set:nnn{stex}{warning/deprecated}{
3543       \\\
3544       \c_backslash_str trefi-is-deprecated! \\\
3545       Please~use~\c_backslash_str STEXsymbol{#2}![#2s]~instead!~(in~file~
3546       \stex_path_to_string:N \g_stex_currentfile_seq)
3547       \\\ \\\
3548     }
3549     \msg_warning:nn{stex}{warning/deprecated}
3550     \STEXsymbol { #2 }![ \comp{#2s} ]
3551   } {
3552     \msg_set:nnn{stex}{warning/deprecated}{
3553       \\\
3554       \c_backslash_str trefi-is-deprecated! \\\
3555       Please~use~\c_backslash_str STEXsymbol { #1 }[ #2s ]~instead!~(in~file~
3556       \stex_path_to_string:N \g_stex_currentfile_seq)
3557       \\\ \\\
3558     }
3559     \msg_warning:nn{stex}{warning/deprecated}
3560     \STEXsymbol { #1 }![ \comp{#2s} ]
3561   }
3562 }
3563
3564
3565 \NewDocumentCommand \Trefis { O{} m } {
3566   \str_set:Nn \l_tmpa_str { #1 }
3567   \str_if_empty:NTF \l_tmpa_str {
3568     \msg_set:nnn{stex}{warning/deprecated}{
3569       \\\
3570       \c_backslash_str Trefis-is-deprecated! \\\
3571       Please~use~\c_backslash_str STEXsymbol{#2}![ \exp_after:wN \stex_capitalize:n #2s ]~inst
3572       \stex_path_to_string:N \g_stex_currentfile_seq)
3573       \\\ \\\
3574     }
3575     \msg_warning:nn{stex}{warning/deprecated}
3576     \STEXsymbol { #2 }![ \comp{\exp_after:wN \stex_capitalize:n #2s} ]
3577   } {
3578     \msg_set:nnn{stex}{warning/deprecated}{
3579       \\\
3580       \c_backslash_str Trefis-is-deprecated! \\\
3581       Please~use~\c_backslash_str STEXsymbol { #1 }[ \exp_after:wN \stex_capitalize:n #2s ]~
3582       \stex_path_to_string:N \g_stex_currentfile_seq)
3583       \\\ \\\
3584     }

```

```

3585 \msg_warning:nn{stex}{warning/deprecated}
3586 \STEXsymbol { #1 }![ \comp{\exp_after:wN \stex_capitalize:n #2s} ]
3587 }
3588 }
3589
3590 \NewDocumentCommand \trefii { 0{} m m } {
3591 \str_set:Nn \l_tmpa_str { #1 }
3592 \str_if_empty:NTF \l_tmpa_str {
3593 \msg_set:nnn{stex}{warning/deprecated}{
3594 \\\
3595 \c_backslash_str trefii~is-deprecated! \\\
3596 Please~use~\c_backslash_str STEXsymbol{#2~#3}![#2~#3]~instead!~(in~file~
3597 \stex_path_to_string:N \g_stex_currentfile_seq)
3598 \\\ \\\
3599 }
3600 \msg_warning:nn{stex}{warning/deprecated}
3601 \STEXsymbol { #2~#3 }![ \comp{#2~#3} ]
3602 } {
3603 \msg_set:nnn{stex}{warning/deprecated}{
3604 \\\
3605 \c_backslash_str trefii~is-deprecated! \\\
3606 Please~use~\c_backslash_str STEXsymbol { #1 }[ #2~#3 ]~instead!~(in~file~
3607 \stex_path_to_string:N \g_stex_currentfile_seq)
3608 \\\ \\\
3609 }
3610 \msg_warning:nn{stex}{warning/deprecated}
3611 \STEXsymbol { #1 }![ \comp{#2~#3} ]
3612 }
3613 }
3614
3615 \NewDocumentCommand \trefiii { 0{} m m m } {
3616 \str_set:Nn \l_tmpa_str { #1 }
3617 \str_if_empty:NTF \l_tmpa_str {
3618 \msg_set:nnn{stex}{warning/deprecated}{
3619 \\\
3620 \c_backslash_str trefiii~is-deprecated! \\\
3621 Please~use~\c_backslash_str STEXsymbol{#2~#3~#4}![#2~#3~#4]~instead!~(in~file~
3622 \stex_path_to_string:N \g_stex_currentfile_seq)
3623 \\\ \\\
3624 }
3625 \msg_warning:nn{stex}{warning/deprecated}
3626 \STEXsymbol { #2~#3~#4 }![ \comp{#2~#3~#4} ]
3627 } {
3628 \msg_set:nnn{stex}{warning/deprecated}{
3629 \\\
3630 \c_backslash_str trefiii~is-deprecated! \\\
3631 Please~use~\c_backslash_str STEXsymbol { #1 }[ #2~#3~#4 ]~instead!~(in~file~
3632 \stex_path_to_string:N \g_stex_currentfile_seq)
3633 \\\ \\\
3634 }
3635 \msg_warning:nn{stex}{warning/deprecated}
3636 \STEXsymbol { #1 }![ \comp{#2~#3~#4} ]
3637 }
3638 }

```

```

3639
3640
3641 \NewDocumentCommand \treffiis { 0{} m m } {
3642   \str_set:Nn \l_tmpa_str { #1 }
3643   \str_if_empty:NTF \l_tmpa_str {
3644     \msg_set:nnn{stex}{warning/deprecated}{
3645       \\
3646       \c_backslash_str treffiis~is-deprecated! \\
3647       Please~use~\c_backslash_str STExsymbol{#2~#3}![#2~#3s]~instead!~(in~file~
3648       \stex_path_to_string:N \g_stex_currentfile_seq)
3649       \\ \\
3650     }
3651     \msg_warning:nn{stex}{warning/deprecated}
3652     \STExsymbol { #2~#3 }![ \comp{#2~#3s} ]
3653   } {
3654     \msg_set:nnn{stex}{warning/deprecated}{
3655       \\
3656       \c_backslash_str treffiis~is-deprecated! \\
3657       Please~use~\c_backslash_str STExsymbol { #1 }[ #2~#3s ]~instead!~(in~file~
3658       \stex_path_to_string:N \g_stex_currentfile_seq)
3659       \\ \\
3660     }
3661     \msg_warning:nn{stex}{warning/deprecated}
3662     \STExsymbol { #1 }![ \comp{#2~#3s} ]
3663   }
3664 }
3665
3666 \NewDocumentCommand \symvariant { 0{} m 0{0} m m } {
3667   \msg_set:nnn{stex}{warning/deprecated}{
3668     \\
3669     \c_backslash_str symvariant~is-deprecated! \\
3670     Please~use~\c_backslash_str notation[#4]{ #2 }~instead!~(in~file~
3671     \stex_path_to_string:N \g_stex_currentfile_seq)
3672     \\ \\
3673   }
3674   \msg_warning:nn{stex}{warning/deprecated}
3675
3676   \notation[variant=#4]{#2}{#5}
3677 }
3678
3679 \NewDocumentCommand \mixfixi { 0{} m m m } {
3680   \msg_set:nnn{stex}{warning/deprecated}{
3681     \c_backslash_str mixfixi~is~fatally~deprecated!\\
3682     Symbol:~\l__stex_term_highlight_uri_str\\
3683     Current~file:~\stex_path_to_string:N \g_stex_currentfile_seq
3684   }
3685   \msg_error:nn{stex}{warning/deprecated}
3686 }
3687
3688
3689 \NewDocumentCommand \infix {} {
3690   \msg_set:nnn{stex}{warning/deprecated}{
3691     \c_backslash_str infix~is~fatally~deprecated!\\
3692     Symbol:~\l__stex_term_highlight_uri_str\\

```

```

3693     Current~file:~\stex_path_to_string:N \g_stex_currentfile_seq
3694 }
3695 \msg_error:nn{stex}{warning/deprecated}
3696 }
3697
3698 \let\iprec\infpref
3699
3700 \NewDocumentCommand \inlineex { m } {
3701   \msg_set:nnn{stex}{warning/deprecated}{
3702     \c_backslash_str inlineex~is~deprecated!\\
3703     No~replacement~exists~yet.\\
3704     Current~file:~\stex_path_to_string:N \g_stex_currentfile_seq
3705   }
3706   \msg_warning:nn{stex}{warning/deprecated}
3707   #1
3708 }
3709
3710
3711 \NewDocumentCommand \term { m } {
3712   \msg_set:nnn{stex}{warning/deprecated}{
3713     \c_backslash_str term~is~deprecated!\\
3714     No~replacement~exists~yet.\\
3715     Current~file:~\stex_path_to_string:N \g_stex_currentfile_seq
3716   }
3717   \msg_warning:nn{stex}{warning/deprecated}
3718   #1
3719 }
3720
3721
3722 \NewDocumentCommand \Definame { O{} m } {
3723   \stex_get_symbol:n { #2 }
3724   \str_set:Nx \l_tmpa_str {
3725     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3726   }
3727   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3728   \scalatex_if:TF {
3729     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3730       \l_tmpa_str
3731     }
3732   } {
3733     \@defemph {
3734       \exp_after:wN \stex_capitalize:n \l_tmpa_str
3735     } { \l_stex_get_symbol_uri_str }
3736   }
3737 }
3738
3739 \NewDocumentCommand \Definiendum { O{} m m } {
3740   \stex_get_symbol:n { #2 }
3741   \str_set:Nx \l_tmpa_str {
3742     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3743   }
3744   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3745   \scalatex_if:TF {
3746     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {

```



```

3747     \l_tmpa_str
3748   }
3749 } {
3750   \@defemph {
3751     \exp_after:wN \stex_capitalize:n \l_tmpa_str
3752   } { \l_stex_get_symbol_uri_str }
3753 }
3754 }
3755
3756 \NewDocumentCommand \Symname { 0{} m }{
3757   \stex_symname_args:n { #1 }
3758   \stex_get_symbol:n { #2 }
3759   \str_set:Nx \l_tmpa_str {
3760     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3761   }
3762   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3763   \exp_args:NNx \use:nn
3764   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }![
3765     \exp_after:wN \stex_capitalize:n \l_tmpa_str
3766     \l_stex_symname_post_str
3767   ] }
3768 }
3769
3770
3771 \seq_gput_right:Nx \g_stex_smsmode_allowedenvs_seq { \tl_to_str:n { mhmodnl } }
3772 \seq_gput_right:Nx \g_stex_smsmode_allowedenvs_seq { \tl_to_str:n { modsig } }
3773 \tl_gput_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\gimport\syml\symii\symiii\symiv\
3774
3775 % omtex:
3776 \cs_new_protected:Npn \lec #1 {
3777   \strut\hfil\strut\hfill(#1)
3778 }
3779 \cs_new_protected:Npn \nlex #1 {
3780   \textcolor{green}{\s1 #1}}
3781 }
3782
3783
3784 </compat>

```