

stex.sty: \TeX 2.0*

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

December 2, 2021

Abstract

TODO

1 Introduction

TODO

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

Contents

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

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

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

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

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

Example 6

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

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

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

Example 7

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

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

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

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

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

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

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

2.2.1 Other Argument Types

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

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

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

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

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

Example 8

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

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

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

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

Example 9

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

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

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

2 3

2.2.2 Precedences

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

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

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

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

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

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

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

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

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

For example:

Example 10

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

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

2.3 Archives and Imports

2.3.1 Namespaces

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

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

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

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

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

2.3.2 Paths in Import-Statements

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

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

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

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

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

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

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

3 Documentation

3.1 Utils

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

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

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

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

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

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

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

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

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

Annotates the HTML generated by `⟨content⟩` with

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

3.1.2 Languages

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

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

3.2 Files, Paths, URIs

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

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

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

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

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

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

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

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

`\g_stex_currentfile_seq`

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

Test 1

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

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

3.3 MathHub Archives

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

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

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

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

`\l_stex_current_repository_prop`

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

`\stex_set_current_repository:n`

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

`\stex_require_repository:n`

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

`\libinput`

`\libinput{<filename>}`

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

Test 2

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

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

3.4 The Module System

`\l_stex_current_module_prop`

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

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

Additionally, it stores:

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

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

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

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

`\stex_add_to_current_module:n`
`\STEXexport`

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

`\stex_add_constant_to_current_module:n`

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

`\stex_add_import_to_current_module:n`

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

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

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

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

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

Computes the current namespace

Test 3

```

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

```

```

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

```

3.4.1 The module-environment

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

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

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

Test 4

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

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

Test 5

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

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

\l_stex_all_modules_seq

Stores full URIs for all modules currently in scope.

\STEXModule

\STEXModule {*<fragment>*}

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

`\stex_invoke_module:n`

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

Test 6

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

Module 3.2[STEXModuleTest1]

Module 3.3[STEXModuleTest2]

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

3.4.2 SMS Mode

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

`\g_stex_smsmode_allowedmacros_tl`

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

`\g_stex_smsmode_allowedmacros_escape_tl`

Macros that are executed with the category codes restored.

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

`\g_stex_smsmode_allowedenvs_seq`

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

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

Tests whether SMS mode is currently active.

`\stex_smsmode_set_codes:`

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

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

`\stex_in_smsmode:nn`

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

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

Test 7

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

3.4.3 Imports and Inheritance

`\importmodule`

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

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

Test 8

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

Module 3.5[Foo]

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

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

Module 3.6[Importtest]

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

Module 3.7[Importtest2]

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

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

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

Test 9

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

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

Module 3.8[UseTest1]

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

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

Test 10

Circular dependencies:

```

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

```

Circular dependencies:

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

| | |
|---|---|
| <hr/> <hr/> | <hr/> |
| <code>\stex_import_module_uri:nn</code> | <code>\stex_import_module_uri:nn {<archive-ID>} {<module-path>}</code> |
| | Determines the URI of a module by splitting <code><module-path></code> into <code><path>?<name></code> . If <code><module-path></code> does <i>not</i> contain a <code>?</code> -character, we consider it to be the <code><name></code> , and <code><path></code> to be empty. If <code><archive-ID></code> is empty, it is automatically set to the ID of the current archive (if one exists). |
| | 1. If <code><archive-ID></code> is empty: |
| | (a) If <code><path></code> is empty, then <code><name></code> must have been declared earlier in the same file and retrievable from <code>\g_stex_modules_in_file_seq</code> , or a file with name <code><name>.<lang>.tex</code> must exist in the same folder, containing a module <code><name></code> . That module should have the same namespace as the current one. |
| | (b) If <code><path></code> is not empty, it must point to the relative path of the containing file as well as the namespace. |
| | 2. Otherwise: |
| | (a) If <code><path></code> is empty, then <code><name></code> must have been declared earlier in the same file and retrievable from <code>\g_stex_modules_in_file_seq</code> , or a file with name <code><name>.<lang>.tex</code> must exist in the top <code>source</code> folder of the archive, containing a module <code><name></code> . That module should lie directly in the namespace of the archive. |
| | (b) If <code><path></code> is not empty, it must point to the path of the containing file as well as the namespace, relative to the namespace of the archive. If a module by that namespace exists, it is returned. Otherwise, we call <code>\stex_require_module:nn</code> on the <code>source</code> directory of the archive to find the file. |

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

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

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

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

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

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

3.5 Symbols and Terms

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

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

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

`\stex_symdecl_do:n`

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

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

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

Test 11

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

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

`\l_stex_all_symbols_seq`

Stores full URIs for all modules currently in scope.

`\stex_get_symbol:n`

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

`\STEXsymbol`

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

`\symref`

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

`\stex_invoke_symbol:n`

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

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

`\notation`

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

`\stex_notation_do:nn`

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

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

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

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

Test 12

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

Module 3.13[NotationTest]

`\symdef`

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

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

Test 13

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

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

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

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

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


```

Module 3.16[MathTest2]
  <a|[b,c,d,e,f]g> and <a|[b,c]g> and <a|[b]c>
  a+(b·c) and a· $\frac{a}{b}+\frac{a}{c}$ 
  a+(b·c) and a· $\frac{a}{b}+\frac{a}{c}$ 

```

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

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

Test 16

```

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

\bar[some ]a[ and some ]b[ and also some ]c[ here].
 $\bar*[\text{some }]a[\text{ and some }]b[\text{ and also some }]c[\text{ here}]\$.$ 
 $\bar![\mathtt{bar}]\$$ 
\bar{*a}*{b}[or just some ]c
\bar![bar]
\bar[or first ]*[2]{b}[ , then ]*[3]{c}[ , and finally ]a
\end{module}

```

```

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

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

```

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

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

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

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

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

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

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

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

3.6 Structural Features

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

3.6.1 Structures

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

Test 17

```

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

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

\present\magma

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

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

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 \TeX document class

```

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

```

```

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

```

4.2 Preliminaries

```

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

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

```

\sTeX The sTeX logo:

```

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

```

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

Messages

```

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

```

\stex_debug:n Debug mode

```

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

```

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

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

(End definition for `\c__stex_sms_iow`.)

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

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

4.2.1 L^AT_EX_ML and S^CA_LT_EX

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

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

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

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

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

4.2.2 HTML Annotations

77 `<@=stex_annotate>`

`\l__stex_annotate_arg_tl`
`\c__stex_annotate_emptyarg_tl`

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

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

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

`__stex_annotate_checkempty:n`

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

(End definition for `__stex_annotate_checkempty:n`.)

`\stex_annotate:nnw`

`\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^Lat_Ex).

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

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

```

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

```

```

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

```

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

4.2.3 Languages

```

173 <@@=stex_language>

```

`\c_stex_languages_prop`

`\c_stex_language_abbrevs_prop`

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

```

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

```

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

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

```

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

```

4.3 Files, Paths and URIs

214 `<@@=stex_path>`

4.3.1 Generic Path Handling

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

```

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

```

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

```

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

```

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

```

\c__stex_path_dot_str . 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_tmpa_str_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 \stex_in_repository:nn {
486   \str_set:Nx \l_tmpa_str { #1 }
487   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
488   \str_if_empty:NTF \l_tmpa_str {
489     \exp_args:Ne \l_tmpa_cs{
490       \prop_item:Nn \l_stex_current_repository_prop { id }
491     }
492   }{
493     \stex_require_repository:n \l_tmpa_str
494     \str_set:Nx \l_tmpa_str { #1 }
495     \exp_args:Nne \use:nn {

```

```

496     \stex_set_current_repository:n \l_tmpa_str
497     \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
498   }{
499     \stex_set_current_repository:n {
500       \prop_item:Nn \l_stex_current_repository_prop { id }
501     }
502   }
503 }
504 }
505
506 \cs_new_protected:Nn \inputref:nn {
507   \stex_in_repository:nn {#1} {
508     \ifinputref
509       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
510     \else
511       \inputreftrue
512       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
513     \inputreffalse
514   \fi
515 }
516 }
517 \NewDocumentCommand \inputref { 0{} m}{
518   \inputref:nn{ #1 }{ #2 }
519 }

```

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

`\mhpath`

```

520 \def \mhpath #1 #2 {
521   \exp_args:Ne \str_if_eq:nnTF{#1}{}{
522     \c_stex_mathhub_str /
523     \prop_item:Nn \l_stex_current_repository_prop { id }
524     / source / #2
525   }{
526     \c_stex_mathhub_str / #1 / source / #2
527   }
528 }

```

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

`\libinput`

```

529 \cs_new_protected:Npn \libinput #1 {
530   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
531     \msg_set:nnn{stex}{error/norepository}{
532       \c_backslash_str libinput-needs-to-be-called-in-an-archive
533     }
534     \msg_error:nn{stex}{error/norepository}
535   }
536   \bool_set_false:N \l_tmpa_bool
537   \tl_clear:N \l_tmpa_tl
538   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
539   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
540   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str
541   \seq_pop_left:NNT \l_tmpb_seq \l_tmpb_str {
542     \seq_put_right:No \l_tmpa_seq \l_tmpb_str

```

```

543 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
544 / meta-inf / lib / #1.tex}{
545 \bool_set_true:N \l_tmpa_bool
546 \tl_put_right:Nx \l_tmpa_tl {
547 \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
548 / meta-inf / lib / #1.tex}
549 }
550 }{}
551 }
552 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
553 / \l_tmpa_str / lib / #1.tex
554 }{
555 \bool_set_true:N \l_tmpa_bool
556 \tl_put_right:Nx \l_tmpa_tl {
557 \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
558 / \l_tmpa_str / lib / #1.tex}
559 }
560 }{}
561 \bool_if:NF \l_tmpa_bool {
562 \msg_set:nnn{stex}{error/nofile}{
563 \c_backslash_str libinput~no~file~#1.tex~found!
564 }
565 \msg_error:nn{stex}{error/nofile}
566 }
567 \l_tmpa_tl
568 }

```

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

4.5 Module System

```

569 <@@=stex_module>

```

`\l_stex_current_module_prop`

```

570 \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`

```

571 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
572 \prop_if_empty:NTF \l_stex_current_module_prop
573 \prg_return_false: \prg_return_true:
574 }

```

(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`

```

575 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
576 \prop_if_exist:cTF { c_stex_module_#1_prop }
577 \prg_return_true: \prg_return_false:
578 }

```

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

`\stex_add_to_current_module:n`
`\STEXexport`

```

579 \cs_new_protected:Nn \stex_add_to_current_module:n {
580   \prop_get:NnN \l_stex_current_module_prop { content } \l_tmpa_tl
581   \tl_put_right:Nn \l_tmpa_tl { #1 }
582   \prop_put:Nno \l_stex_current_module_prop { content } { \l_tmpa_tl }
583 }
584 \cs_new_protected:Npn \STEXexport #1 {
585   \stex_smsmode_set_codes:
586   \stex_add_to_current_module:n { #1 }
587   #1
588 }

```

(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`

```

589 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
590   \str_set:Nx \l_tmpa_str { #1 }
591   \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
592   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
593   \prop_put:Nno \l_stex_current_module_prop { constants } \l_tmpa_seq
594 }

```

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

`\stex_add_import_to_current_module:n`

```

595 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
596   \str_set:Nx \l_tmpa_str { #1 }
597   \prop_get:NnN \l_stex_current_module_prop { imports } \l_tmpa_seq
598   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
599   \prop_put:Nno \l_stex_current_module_prop { imports } \l_tmpa_seq
600 }

```

(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`

```

601 \str_new:N \l_stex_modules_ns_str
602 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
603   \str_set:Nx \l_tmpa_str { #1 }
604   \seq_set_eq:NN \l_tmpa_seq #2
605   % split off file extension
606   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
607   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
608   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
609   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
610
611   \bool_set_true:N \l_tmpa_bool
612   \bool_while_do:Nn \l_tmpa_bool {
613     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
614     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
615       {source} { \bool_set_false:N \l_tmpa_bool }

```



```

616     }{}{
617         \seq_if_empty:NT \l_tmpa_seq {
618             \bool_set_false:N \l_tmpa_bool
619         }
620     }
621 }
622
623 \seq_if_empty:NTF \l_tmpa_seq {
624     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
625 }{
626     \str_set:Nx \l_stex_modules_ns_str {
627         \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
628     }
629 }
630 }

```

(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:`

```

631 \cs_new_protected:Nn \stex_modules_current_namespace: {
632     \prop_get:NnNTF \l_stex_current_repository_prop { ns } \l_tmpa_str {
633         \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
634     }{
635         % split off file extension
636         \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
637         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
638         \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
639         \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
640         \seq_put_right:No \l_tmpa_seq \l_tmpb_str
641         \str_set:Nx \l_stex_modules_ns_str {
642             file:/\stex_path_to_string:N \l_tmpa_seq
643         }
644     }
645 }

```

(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

```

646 \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`

```

647 \NewDocumentCommand \STEXModule { m } {
648     \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
649     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
650     \tl_set:Nn \l_tmpa_tl {
651         \msg_set:nnn{stex}{error/unknownmodule}{
652             No~module~#1~found!
653         }
654         \msg_error:nn{stex}{error/unknownmodule}
655     }

```

```

656 \seq_map_inline:Nn \l_stex_all_modules_seq {
657   \str_set:Nn \l_tmpb_str { ##1 }
658   \str_if_eq:eeT { \l_tmpa_str } {
659     \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
660   } {
661     \seq_map_break:n {
662       \tl_set:Nn \l_tmpa_tl {
663         \stex_invoke_module:n { ##1 }
664       }
665     }
666   }
667 }
668 \l_tmpa_tl
669 }
670
671 \cs_new_protected:Nn \stex_invoke_module:n {
672   \stex_debug:n{Invoking~module~#1}
673   \peek_charcode_remove:NTF ! {
674     \__stex_module_invoke_uri:nN { #1 }
675   } {
676     \peek_charcode_remove:NTF ? {
677       \__stex_module_invoke_symbol:nn { #1 }
678     } {
679       \msg_set:nnn{stex}{error/syntax}{
680         Syntax~error:~?~or~!~expected~after~
681         \c_backslash_str STEXModule{#1}
682       }
683       \msg_error:nn{stex}{error/syntax}
684     }
685   }
686 }
687
688 \cs_new_protected:Nn \__stex_module_invoke_uri:nN {
689   \str_set:Nn #2 { #1 }
690 }
691
692 \cs_new_protected:Nn \__stex_module_invoke_symbol:nn {
693   \stex_invoke_symbol:n{#1?#2}
694 }

```

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

module module arguments:

```

695 \keys_define:nn { stex / module } {
696   title      .tl_set_x:N = \l_stex_module_title_str ,
697   ns         .tl_set_x:N = \l_stex_module_ns_str ,
698   lang       .tl_set_x:N = \l_stex_module_lang_str ,
699   sig        .tl_set_x:N = \l_stex_module_sig_str ,
700   creators   .tl_set_x:N = \l_stex_module_creators_str ,
701   contributors .tl_set_x:N = \l_stex_module_contributors_str ,
702   meta       .tl_set_x:N = \l_stex_module_meta_str
703 }
704

```

```

705 % module parameters here? In the body?
706
707 \cs_new_protected:Nn \__stex_module_args:n {
708   \str_clear:N \l_stex_module_title_str
709   \str_clear:N \l_stex_module_ns_str
710   \str_clear:N \l_stex_module_lang_str
711   \str_clear:N \l_stex_module_sig_str
712   \str_clear:N \l_stex_module_creators_str
713   \str_clear:N \l_stex_module_contributors_str
714   \str_clear:N \l_stex_module_meta_str
715   \keys_set:nn { stex / module } { #1 }
716   \exp_args:NNo \str_set:Nn \l_stex_module_title_str
717     \l_stex_module_title_str
718   \exp_args:NNo \str_set:Nn \l_stex_module_ns_str
719     \l_stex_module_ns_str
720   \exp_args:NNo \str_set:Nn \l_stex_module_lang_str
721     \l_stex_module_lang_str
722   \exp_args:NNo \str_set:Nn \l_stex_module_sig_str
723     \l_stex_module_sig_str
724   \exp_args:NNo \str_set:Nn \l_stex_module_meta_str
725     \l_stex_module_meta_str
726   \exp_args:NNo \str_set:Nn \l_stex_module_creators_str
727     \l_stex_module_creators_str
728   \exp_args:NNo \str_set:Nn \l_stex_module_contributors_str
729     \l_stex_module_contributors_str
730 }

```

__stex_module_begin_module: implements \begin{module}

```

731 \cs_new_protected:Nn \__stex_module_begin_module: {
732   % Nested module?
733   \stex_if_in_module:TF {
734     % Nested module
735     \prop_get:NnN \l_stex_current_module_prop
736       { ns } \l_stex_module_ns_str
737     \str_set:Nx \l_stex_module_name_str {
738       \prop_item:Nn \l_stex_current_module_prop
739         { name } / \l_stex_module_name_str
740     }
741   }{
742     % not nested:
743     \str_if_empty:NT \l_stex_module_ns_str {
744       \stex_modules_current_namespace:
745       \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
746       \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
747         / {\l_stex_module_ns_str}
748       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
749       \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
750         \str_set:Nx \l_stex_module_ns_str {
751           \stex_path_to_string:N \l_tmpa_seq
752         }
753       }
754     }
755   }
756

```

```

757 % language
758 \str_if_empty:NT \l_stex_module_lang_str {
759   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
760   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
761   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
762   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
763   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
764     \stex_debug:n {Language~\l_stex_module_lang_str~
765       inferred~from~file~name}
766     \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
767   }
768 }
769
770 \str_if_empty:NF \l_stex_module_lang_str {
771   \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
772   \l_tmpa_str {
773     \ltx@ifpackageloaded{babel}{
774       \exp_args:Nx \selectlanguage { \l_tmpa_str }
775     }{}
776   } {
777     \msg_set:nnn{stex}{error/unknownlanguage}{
778       Unknown~language~\l_tmpa_str
779     }
780     \msg_error:nn{stex}{error/unknownlanguage}
781   }
782 }
783
784 % signature
785 \str_if_empty:NTF \l_stex_module_sig_str {
786   \str_clear:N \l_tmpa_str
787   \seq_clear:N \l_tmpa_seq
788   \tl_clear:N \l_tmpa_tl
789   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
790     name      = \l_stex_module_name_str ,
791     ns        = \l_stex_module_ns_str ,
792     imports   = \exp_not:o { \l_tmpa_seq } ,
793     constants = \exp_not:o { \l_tmpa_seq } ,
794     content   = \exp_not:o { \l_tmpa_tl } ,
795     file      = \exp_not:o { \g_stex_currentfile_seq } ,
796     lang      = \l_stex_module_lang_str ,
797     sig       = \l_stex_module_sig_str ,
798     meta      = \l_stex_module_meta_str
799   }
800 }{
801   \str_if_empty:NT \l_stex_module_lang_str {
802     \msg_set:nnn{stex}{error/siglanguage}{
803       Module~\l_stex_module_ns_str?\l_stex_module_name_str~
804       declares~signature~\l_stex_module_sig_str,~but~does~not~
805       declare~its~language
806     }
807     \msg_error:nn{stex}{error/siglanguage}
808   }
809
810   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq

```

```

811 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
812 \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
813 \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
814 \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
815 \str_set:Nx \l_tmpa_str {
816   \stex_path_to_string:N \l_tmpa_seq /
817   \l_tmpa_str . \l_stex_module_sig_str .tex
818 }
819 \IfFileExists \l_tmpa_str {
820   \exp_args:No \stex_in_smsmode:nn { \l_tmpa_str } {
821     \seq_clear:N \l_stex_all_modules_seq
822     \prop_clear:N \l_stex_current_module_prop
823     \stex_debug:n{Loading~signature~\l_tmpa_str}
824     \input { \l_tmpa_str }
825   }
826 }{
827   \msg_set:nnn{stex}{error/modulemissing}{
828     No~file~for~signature~module~\l_tmpa_str~found
829   }
830   \msg_error:nn{stex}{error/modulemissing}
831 }
832 \stex_activate_module:n {
833   \l_stex_module_ns_str ? \l_stex_module_name_str
834 }
835 \prop_set_eq:Nc \l_stex_current_module_prop {
836   c_stex_module_
837   \l_stex_module_ns_str ?
838   \l_stex_module_name_str
839   _prop
840 }
841 }
842
843 % metatheory
844 \str_if_empty:NT \l_stex_module_meta_str {
845   \str_set:Nx \l_stex_module_meta_str {
846     \c_stex_metatheory_ns_str ? Metatheory
847   }
848 }
849
850
851 \stex_debug:n{
852   New~module:\\
853   Namespace:~\l_stex_module_ns_str\\
854   Name:~\l_stex_module_name_str\\
855   Language:~\l_stex_module_lang_str\\
856   Signature:~\l_stex_module_sig_str\\
857   Metatheory:~\l_stex_module_meta_str\\
858   File:~\stex_path_to_string:N \g_stex_currentfile_seq
859 }
860
861 \seq_put_right:Nx \l_stex_all_modules_seq {
862   \l_stex_module_ns_str ? \l_stex_module_name_str
863 }
864

```

```

865 \seq_gput_right:Nx \g_stex_modules_in_file_seq
866   { \l_stex_module_ns_str ? \l_stex_module_name_str }
867
868 \stex_if_smsmode:TF {
869   \stex_smsmode_set_codes:
870 } {
871   \begin{stex_annotate_env} {theory} {
872     \l_stex_module_ns_str ? \l_stex_module_name_str
873   }
874
875   \stex_annotate_invisible:nnn{header}{} {
876     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
877     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
878     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
879       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
880     }
881   }
882 }
883
884 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
885   \exp_args:Nx \STEXexport{
886     \stex_activate_module:n { \l_stex_module_meta_str }
887   }
888 }
889 % TODO: Inherit metatheory for nested modules?
890 }
891 \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}

```

892 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
893 \cs_new_protected:Nn \_stex_module_end_module: {
894   \str_set:Nx \l_tmpa_str {
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   %^^A \prop_new:c { \l_tmpa_str }
901   \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
902   \stex_debug:n{Closing-module~\prop_item:Nn \l_stex_current_module_prop { name }}
903   \stex_if_smsmode:TF {
904     \exp_args:Nx \stex_addtosms:n {
905       \prop_gset_from_keyval:cn {
906         c_stex_module_
907         \prop_item:Nn \l_stex_current_module_prop { ns } ?
908         \prop_item:Nn \l_stex_current_module_prop { name }
909         _prop
910       } {
911         name      = \prop_item:cn { \l_tmpa_str } { name } ,
912         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
913         imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
914         constants = \prop_item:cn { \l_tmpa_str } { constants } ,

```

```

915         content = \prop_item:cn { \l_tmpa_str } { content } ,
916         file     = \prop_item:cn { \l_tmpa_str } { file } ,
917         lang     = \prop_item:cn { \l_tmpa_str } { lang } ,
918         sig      = \prop_item:cn { \l_tmpa_str } { sig } ,
919         meta     = \prop_item:cn { \l_tmpa_str } { meta }
920     }
921 }
922 }{
923     \end{stex_annotate_env}
924 }
925 }

```

(End definition for `__stex_module_end_module:.`)

@module The core environment, with no header

```

926 \NewDocumentEnvironment { @module } { 0{} m } {
927     \str_set:Nx \l_stex_module_name_str { #2 }
928     \par
929     \__stex_module_args:n { #1 }
930     \__stex_module_begin_module:
931 } {
932     \__stex_module_end_module:
933 }

```

\stex_modules_heading: Code for document headers

```

934 \cs_if_exist:NTF \thesection {
935     \newcounter{module}[section]
936 }{
937     \newcounter{module}
938 }
939
940 \bool_if:NT \c_stex_showmods_bool {
941     \latexml_if:F { \RequirePackage{mdframed} }
942 }
943
944 \cs_new_protected:Nn \stex_modules_heading: {
945     \stepcounter{module}
946     \par
947     \bool_if:NT \c_stex_showmods_bool {
948         \noindent{\textbf{Module} ~
949             \cs_if_exist:NT \thesection {\thesection.}
950             \themodule ~ [\l_stex_module_name_str]
951         }
952         % TODO references
953         % \sref@label@id{Module \thesection.\themodule [\module@name]]%
954         \str_if_empty:NTF \l_stex_module_title_str {
955             }{
956                 \quad(\l_stex_module_title_str)\hfill
957             }\par
958         }
959     }

```

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

Finally:

```

960 \NewDocumentEnvironment { module } { 0{} m } {
961   \bool_if:NT \c_stex_showmods_bool {
962     \begin{mdframed}
963   }
964   \begin{@module}[#1]{#2}
965   \stex_modules_heading:
966 }{
967   \end{@module}
968   \bool_if:NT \c_stex_showmods_bool {
969     \end{mdframed}
970   }
971 }

```

4.5.2 SMS Mode

```

972 <@=stex_smsmode>

```

```

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

```

```

973 \tl_new:N \g_stex_smsmode_allowedmacros_tl
974 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
975 \seq_new:N \g_stex_smsmode_allowedenvs_seq
976
977 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
978   \makeatletter
979   \makeatother
980   \ExplSyntaxOn
981   \ExplSyntaxOff
982 }
983
984 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
985   \symdef
986   \importmodule
987   \notation
988   \symdecl
989   \STEXexport
990 }
991
992 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
993   \tl_to_str:n {
994     module,
995     @module
996   }
997 }

```

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

```

\stex_if_smsmode_p:
\stex_if_smsmode:TF

```

```

998 \bool_new:N \g__stex_smsmode_bool
999 \bool_set_false:N \g__stex_smsmode_bool
1000 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1001   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1002 }

```

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


```

\stex_smsmode_if_catcodes_p: Checks whether the SMS mode category code scheme is active.
__stex_smsmode_if_catcodes:TF
1003 \bool_new:N \g__stex_smsmode_catcode_bool
1004 \bool_set_false:N \g__stex_smsmode_catcode_bool
1005 \prg_new_conditional:Nnn \__stex_smsmode_if_catcodes: { p, T, F, TF } {
1006   \bool_if:NTF \g__stex_smsmode_catcode_bool
1007   \prg_return_true: \prg_return_false:
1008 }

```

(End definition for __stex_smsmode_if_catcodes:TF.)

\stex_smsmode_set_codes:

```

1009 \cs_new_protected:Nn \stex_smsmode_set_codes: {
1010   \stex_if_smsmode:T {
1011     \__stex_smsmode_if_catcodes:F {
1012       \bool_gset_true:N \g__stex_smsmode_catcode_bool
1013       \exp_after:wN \char_gset_active_eq:NN
1014       \c_backslash_str \__stex_smsmode_cs:
1015       \tex_global:D \char_set_catcode_active:N \
1016       \tex_global:D \char_set_catcode_other:N $
1017       \tex_global:D \char_set_catcode_other:N ^
1018       \tex_global:D \char_set_catcode_other:N _
1019       \tex_global:D \char_set_catcode_other:N &
1020       \tex_global:D \char_set_catcode_other:N ##
1021     }
1022   }
1023 } \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.

```

1024 \cs_new_protected:Nn \__stex_smsmode_unset_codes: {
1025   \__stex_smsmode_if_catcodes:T {
1026     \bool_gset_false:N \g__stex_smsmode_catcode_bool
1027     \exp_after:wN \tex_global:D \exp_after:wN
1028     \char_set_catcode_escape:N \c_backslash_str
1029     \tex_global:D \char_set_catcode_math_toggle:N $
1030     \tex_global:D \char_set_catcode_math_superscript:N ^
1031     \tex_global:D \char_set_catcode_math_subscript:N _
1032     \tex_global:D \char_set_catcode_alignment:N &
1033     \tex_global:D \char_set_catcode_parameter:N ##
1034   }
1035 } \iffalse $ \fi % to make syntax highlighting work again

```

(End definition for __stex_smsmode_unset_codes:.)

\stex_in_smsmode:nn

```

1036 \cs_new_protected:Nn \stex_in_smsmode:nn {
1037   \vbox_set:Nn \l_tmpa_box {
1038     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1039     \bool_gset_true:N \g__stex_smsmode_bool
1040     \stex_smsmode_set_codes:
1041     #2
1042     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1043     \stex_if_smsmode:F {

```

```

1044     \__stex_smsmode_unset_codes:
1045   }
1046 }
1047 \box_clear:N \l_tmpa_box
1048 }

```

(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:

```

1049 \cs_new_protected:Nn \__stex_smsmode_cs: {
1050   \str_clear:N \l_tmpa_str
1051   \peek_analysis_map_inline:n {
1052     % #1: token (one expansion)
1053     % #2: charcode
1054     % #3 catcode
1055     \token_if_eq_charcode:NNTF ##3 B {
1056       % token is a letter
1057       \exp_args:NNNo \str_put_right:Nn \l_tmpa_str { ##1 }
1058     } {
1059       \str_if_empty:NNTF \l_tmpa_str {
1060         % we don't allow (or need) single non-letter CSs
1061         % for now
1062         \peek_analysis_map_break:
1063       } {
1064         \str_if_eq:ontf \l_tmpa_str { begin } {
1065           \peek_analysis_map_break:n {
1066             \exp_after:wN \__stex_smsmode_checkbegin:n ##1
1067           }
1068         } {
1069           \str_if_eq:ontf \l_tmpa_str { end } {
1070             \peek_analysis_map_break:n {
1071               \exp_after:wN \__stex_smsmode_checkend:n ##1
1072             }
1073           } {
1074             \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
1075             \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1076               \g_stex_smsmode_allowedmacros_tl
1077               { \use:c{\l_tmpa_str} } {
1078               \stex_debug:n{Executing~1:~\l_tmpa_str}
1079               \peek_analysis_map_break:n {
1080                 \exp_after:wN \l_tmpa_tl ##1
1081               }
1082             } {
1083               \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1084               \g_stex_smsmode_allowedmacros_escape_tl
1085               { \use:c{\l_tmpa_str} } {
1086               \stex_debug:n{Executing~2:~\l_tmpa_str}
1087               % TODO \__stex_smsmode_rescan_cs:
1088               \exp_after:wN \exp_after:wN \exp_after:wN
1089               \token_if_eq_charcode:NNTF \exp_after:wN \c_backslash_str ##1 {
1090               \peek_analysis_map_break:n {
1091                 \__stex_smsmode_unset_codes:
1092               \__stex_smsmode_rescan_cs:

```

```

1093 %           }
1094 %       } {
1095 %           \peek_analysis_map_break:n {
1096 %               \__stex_smsmode_unset_codes:
1097 %               \exp_after:wN \l_tmpa_tl ##1
1098 %           }
1099 %       }
1100 %   } {
1101 %       \peek_analysis_map_break:n { ##1 }
1102 %   }
1103 % }
1104 % }
1105 % }
1106 % }
1107 % }
1108 % }
1109 % }

```

(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:

```

1110 \cs_new_protected:Nn \__stex_smsmode_rescan_cs: {
1111   \str_clear:N \l_tmpb_str
1112   \peek_analysis_map_inline:n {
1113     \token_if_eq_charcode:NNTF ##3 B {
1114       % token is a letter
1115       \exp_args:NNo \str_put_right:Nn \l_tmpb_str { ##1 }
1116     } {
1117       \peek_analysis_map_break:n {
1118         \exp_after:wN \use:c \exp_after:wN {
1119           \exp_after:wN \l_tmpa_str\exp_after:wN
1120         } \use:c { \l_tmpb_str \exp_after:wN } ##1
1121       }
1122     }
1123   }
1124 }

```

(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.

```

1125 \cs_new_protected:Nn \__stex_smsmode_checkbegin:n {
1126   \str_set:Nn \l_tmpa_str { #1 }
1127   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1128     \__stex_smsmode_unset_codes:
1129     \begin{#1}
1130   }
1131 }

```

(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.

```

1132 \cs_new_protected:Nn \__stex_smsmode_checkend:n {
1133   \str_set:Nn \l_tmpa_str { #1 }
1134   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {

```

```

1135     \end{#1}
1136   }
1137 }

```

(End definition for `_stex_smsmode_checkend:n`.)

4.5.3 Inheritance

```

1138 <@=stex_importmodule>

```

`\stex_import_module_uri:nn`

```

1139 \cs_new_protected:Nn \stex_import_module_uri:nn {
1140   \str_set:Nx \l__stex_importmodule_archive_str { #1 }
1141   \str_set:Nn \l__stex_importmodule_path_str { #2 }
1142   \str_if_empty:NT \l__stex_importmodule_archive_str {
1143     \prop_if_empty:NF \l_stex_current_repository_prop {
1144       \prop_get:NnN \l_stex_current_repository_prop { id } \l__stex_importmodule_archive_str
1145     }
1146   }
1147
1148   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l__stex_importmodule_path_str }
1149   \seq_pop_right:NN \l_tmpb_seq \l__stex_importmodule_name_str
1150   \str_set:Nx \l__stex_importmodule_path_str { \seq_use:Nn \l_tmpb_seq ? }
1151
1152   \str_if_empty:NTF \l__stex_importmodule_archive_str {
1153     \stex_modules_current_namespace:
1154     \str_if_empty:NF \l__stex_importmodule_path_str {
1155       \str_set:Nx \l_stex_module_ns_str {
1156         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1157       }
1158     }
1159   }{
1160     \stex_require_repository:n \l__stex_importmodule_archive_str
1161     \prop_get:cnN { c_stex_mathhub_\l__stex_importmodule_archive_str _manifest_prop } { ns }
1162     \l_stex_module_ns_str
1163     \str_if_empty:NF \l__stex_importmodule_path_str {
1164       \str_set:Nx \l_stex_module_ns_str {
1165         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1166       }
1167     }
1168   }
1169 }

```

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

| | |
|---|--|
| <code>\l_stex_importmodule_name_str</code> | Store the return values of <code>\stex_import_module_uri:nn</code> . |
| <code>\l_stex_importmodule_archive_str</code> | 1170 <code>\str_new:N \l__stex_importmodule_name_str</code> |
| <code>\l_stex_importmodule_path_str</code> | 1171 <code>\str_new:N \l__stex_importmodule_archive_str</code> |
| <code>\l_stex_importmodule_file_str</code> | 1172 <code>\str_new:N \l__stex_importmodule_path_str</code> |
| | 1173 <code>\str_new:N \g__stex_importmodule_file_str</code> |

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

```

\stex_import_require_module:nnnn      {\<ns>} {\<archive-ID>} {\<path>} {\<name>}}
1174 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1175   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1176     % \stex_debug:n{Arguments: #1, #2, #3, #4}
1177
1178     % archive
1179     \str_set:Nx \l_tmpa_str { #2 }
1180     \str_if_empty:NTF \l_tmpa_str {
1181       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1182     } {
1183       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1184       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1185       \seq_put_right:Nn \l_tmpa_seq { source }
1186     }
1187
1188     % path
1189     \str_set:Nx \l_tmpb_str { #3 }
1190     \str_if_empty:NTF \l_tmpb_str {
1191       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1192
1193       \ltx@ifpackageloaded{babel} {
1194         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1195           { \language } \l_tmpb_str {
1196           \msg_set:nnn{stex}{error/unknownlanguage}{
1197             Unknown~language~\language
1198           }
1199           \msg_error:nn{stex}{error/unknownlanguage}
1200         }
1201       } {
1202         \str_clear:N \l_tmpb_str
1203       }
1204
1205       \stex_debug:n{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1206       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1207         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1208       }{
1209         \stex_debug:n{Checking~\l_tmpa_str.tex}
1210         \IfFileExists{ \l_tmpa_str.tex }{
1211           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1212         }{
1213           % try english as default
1214           \stex_debug:n{Checking~\l_tmpa_str.en.tex}
1215           \IfFileExists{ \l_tmpa_str.en.tex }{
1216             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1217           }{
1218             \msg_set:nnn{stex}{error/modulemissing}{
1219               No~file~for~module~#1?#4~found
1220             }
1221             \msg_error:nn{stex}{error/modulemissing}
1222           }
1223         }
1224       }
1225     } {
1226

```

```

1227 \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1228 \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1229
1230 \ltx@ifpackageloaded{babel} {
1231   \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1232     { \language } \l_tmpb_str {
1233     \msg_set:nnn{stex}{error/unknownlanguage}{
1234       Unknown~language~\language
1235     }
1236     \msg_error:nn{stex}{error/unknownlanguage}
1237   }
1238 } {
1239   \str_clear:N \l_tmpb_str
1240 }
1241
1242 \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1243
1244 \stex_debug:n{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1245 \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1246   \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1247 }{
1248   \stex_debug:n{Checking~\l_tmpa_str/#4.tex}
1249   \IfFileExists{ \l_tmpa_str/#4.tex }{
1250     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1251   }{
1252     % try english as default
1253     \stex_debug:n{Checking~\l_tmpa_str/#4.en.tex}
1254     \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1255       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1256     }{
1257       \stex_debug:n{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1258       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1259         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1260       }{
1261         \stex_debug:n{Checking~\l_tmpa_str.tex}
1262         \IfFileExists{ \l_tmpa_str.tex }{
1263           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1264         }{
1265           % try english as default
1266           \stex_debug:n{Checking~\l_tmpa_str.en.tex}
1267           \IfFileExists{ \l_tmpa_str.en.tex }{
1268             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1269           }{
1270             \msg_set:nnn{stex}{error/modulemissing}{
1271               No~file~for~module~#1?#4~found
1272             }
1273             \msg_error:nn{stex}{error/modulemissing}
1274           }
1275         }
1276       }
1277     }
1278   }
1279 }
1280 }

```

```

1281
1282 \seq_set_eq:NN \l_tmpa_seq \g_stex_modules_in_file_seq
1283 \seq_clear:N \g_stex_modules_in_file_seq
1284 % \exp_args:Nnx \use:nn {
1285 \exp_args:No \stex_in_smsmode:nn { \g__stex_importmodule_file_str } {
1286 \seq_clear:N \l_stex_all_modules_seq
1287 \prop_clear:N \l_stex_current_module_prop
1288 \str_set:Nx \l_tmpb_str { #2 }
1289 \str_if_empty:NF \l_tmpb_str {
1290 \stex_set_current_repository:n { #2 }
1291 }
1292 \stex_debug:n{Loading~\g__stex_importmodule_file_str}
1293 \input { \g__stex_importmodule_file_str }
1294 }
1295 % }{
1296
1297 % }
1298 \prop_gput:Noo \g_stex_module_files_prop
1299 \g__stex_importmodule_file_str \g_stex_modules_in_file_seq
1300 \seq_set_eq:NN \g_stex_modules_in_file_seq \l_tmpa_seq
1301
1302 \stex_if_module_exists:nF { #1 ? #4 } {
1303 \msg_set:nnn{stex}{error/modulemissing}{
1304 Module~#1?#4~not~found~in~file~\g__stex_importmodule_file_str
1305 }
1306 \msg_error:nn{stex}{error/modulemissing}
1307 }
1308 }
1309 \stex_activate_module:n { #1 ? #4 }
1310 }

```

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

`\stex_activate_module:n`

```

1311 \cs_new_protected:Nn \stex_activate_module:n {
1312 \stex_debug:n{Activating~module~#1}
1313 \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1314 \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1315 \prop_item:cn { c_stex_module_#1_prop } { content }
1316 }
1317 }

```

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

`\importmodule`

```

1318 \NewDocumentCommand \importmodule { 0{} m } {
1319 \stex_import_module_uri:nn { #1 } { #2 }
1320 \stex_debug:n{Importing~module:~
1321 \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1322 }
1323 \stex_if_smsmode:F {
1324 \stex_import_require_module:nnnn
1325 { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1326 { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1327 \stex_annotate_invisible:nnn

```

```

1328     {import} {\l_stex_module_ns_str ? \l__stex_importmodule_name_str} {}
1329 }
1330 \exp_args:Nx \stex_add_to_current_module:n {
1331   \stex_import_require_module:nnnn
1332   { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1333   { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1334 }
1335 \exp_args:Nx \stex_add_import_to_current_module:n {
1336   \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1337 }
1338 \stex_smsmode_set_codes:
1339 }

```

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

`\usemodule`

```

1340 \NewDocumentCommand \usemodule { 0{} m } {
1341   \stex_if_smsmode:F {
1342     \stex_import_module_uri:nn { #1 } { #2 }
1343     \stex_import_require_module:nnnn
1344     { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1345     { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1346     \stex_annotate_invisible:nnn
1347     {usemodule} {\l_stex_module_ns_str ? \l__stex_importmodule_name_str} {}
1348   }
1349   \stex_smsmode_set_codes:
1350 }

```

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

`\g_stex_modules_in_file_seq` `\g_stex_module_files_prop`

```

1351 \seq_new:N \g_stex_modules_in_file_seq
1352 \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

```

1353 <@@=stex_symdecl>

```

`\l_stex_all_symbols_seq` Stores all available symbols

```

1354 \seq_new:N \l_stex_all_symbols_seq

```

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

`\STEXsymbol`

```

1355 \NewDocumentCommand \STEXsymbol { m } {
1356   \stex_get_symbol:n { #1 }
1357   \exp_args:No
1358   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1359 }

```


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

`symdecl` arguments:

```

1360 \keys_define:nn { stex / symdecl } {
1361   name      .tl_set:x:N = \l_stex_symdecl_name_str ,
1362   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1363   args      .tl_set:x:N = \l_stex_symdecl_args_str ,
1364   type      .tl_set:N   = \l_stex_symdecl_type_tl ,
1365   align     .tl_set:N   = \l_stex_symdecl_align_str , % TODO(?)
1366   gfc       .tl_set:N   = \l_stex_symdecl_gfc_str , % TODO(?)
1367   specializes .tl_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
1368   def       .tl_set:N   = \l_stex_symdecl_definiens_tl
1369 }
1370
1371 \bool_new:N \l_stex_symdecl_make_macro_bool
1372
1373 \cs_new_protected:Nn \__stex_symdecl_args:n {
1374   \str_clear:N \l_stex_symdecl_name_str
1375   \str_clear:N \l_stex_symdecl_args_str
1376   \bool_set_false:N \l_stex_symdecl_local_bool
1377   \tl_clear:N \l_stex_symdecl_type_tl
1378   \tl_clear:N \l_stex_symdecl_definiens_tl
1379 }
1380
1381 \keys_set:nn { stex / symdecl } { #1 }
1382
1383 \exp_args:NNo \str_set:Nn \l_stex_symdecl_name_str
1384   \l_stex_symdecl_name_str
1385 \exp_args:NNo \str_set:Nn \l_stex_symdecl_args_str
1386   \l_stex_symdecl_args_str

```

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

```

1387
1388 \NewDocumentCommand \symdecl { s O{} m } {
1389   \__stex_symdecl_args:n { #2 }
1390   \IfBooleanTF #1 {
1391     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1392   } {
1393     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1394   }
1395   \stex_symdecl_do:n { #3 }
1396   \stex_smsmode_set_codes:
1397 }

```

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

`\stex_symdecl_do:n`

```

1398 \cs_new_protected:Nn \stex_symdecl_do:n {
1399   \stex_if_in_module:F {
1400     % TODO throw error? some default namespace?
1401   }
1402
1403   \str_if_empty:NT \l_stex_symdecl_name_str {

```

```

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

```

```

1458     }
1459 }
1460 \bool_if:NTF \l_tmpa_bool {
1461   % possibly numeric
1462   \str_if_empty:NTF \l_stex_symdecl_args_str {
1463     \prop_put:Nnn \l_tmpa_prop { args } {}
1464     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1465   }{
1466     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1467     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1468     \str_clear:N \l_tmpa_str
1469     \int_step_inline:nn \l_tmpa_int {
1470       \str_put_right:Nn \l_tmpa_str i
1471     }
1472     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1473   }
1474 } {
1475   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1476   \prop_put:Nnx \l_tmpa_prop { arity }
1477   { \str_count:N \l_stex_symdecl_args_str }
1478 }
1479 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
1480
1481
1482 % semantic macro
1483
1484 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1485   \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1486     \prop_item:Nn \l_tmpa_prop { module } ?
1487     \prop_item:Nn \l_tmpa_prop { name }
1488   } }
1489
1490   \bool_if:NF \l_stex_symdecl_local_bool {
1491     \exp_args:Nx \stex_add_to_current_module:n {
1492       \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1493         \prop_item:Nn \l_tmpa_prop { module } ?
1494         \prop_item:Nn \l_tmpa_prop { name }
1495       } }
1496     }
1497   }
1498 }
1499
1500 % add to all symbols
1501
1502 \bool_if:NF \l_stex_symdecl_local_bool {
1503   \exp_args:Nx \stex_add_to_current_module:n {
1504     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1505       \prop_item:Nn \l_tmpa_prop { module } ?
1506       \prop_item:Nn \l_tmpa_prop { name }
1507     }
1508   }
1509 }
1510
1511 \stex_debug:n{New~symbol:~

```

```

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

```

```

1566   } {
1567     \stex_annotate_invisible:nnn{type}{-}{\l_stex_symdecl_type_tl$}
1568     \stex_annotate_invisible:nnn{args}{-}{
1569       \prop_item:Nn \l_tmpa_prop { args }
1570     }
1571     \stex_annotate_invisible:nnn{macroname}{-}{#1}
1572     \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
1573       \stex_annotate_invisible:nnn{definiens}{-}{
1574         {\l_stex_symdecl_definiens_tl$}
1575       }
1576     }
1577   }
1578 }

```

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

`\stex_get_symbol:n`

```

1579 \str_new:N \l_stex_get_symbol_uri_str
1580
1581 \cs_new_protected:Nn \stex_get_symbol:n {
1582   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
1583     \__stex_symdecl_get_symbol_from_cs:n { #1 }
1584   }{
1585     % argument is a string
1586     % is it a command name?
1587     \cs_if_exist:cTF { #1 }{
1588       \cs_set_eq:Nc \l_tmpa_tl { #1 }
1589       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
1590       \str_if_empty:NNTF \l_tmpa_str {
1591         \exp_args:Nx \cs_if_eq:NNTF {
1592           \tl_head:N \l_tmpa_tl
1593         } \stex_invoke_symbol:n {
1594           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
1595         }{
1596           \__stex_symdecl_get_symbol_from_string:n { #1 }
1597         }
1598       } {
1599         \__stex_symdecl_get_symbol_from_string:n { #1 }
1600       }
1601     }{
1602       % argument is not a command name
1603       \__stex_symdecl_get_symbol_from_string:n { #1 }
1604       % \l_stex_all_symbols_seq
1605     }
1606   }
1607 }
1608
1609 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
1610   \bool_set_false:N \l_tmpa_bool
1611   \stex_if_in_module:T {
1612     \prop_get:NnN \l_stex_current_module_prop
1613     { constants } \l_tmpa_seq
1614     \exp_args:NNo \seq_if_in:NnT \l_tmpa_seq { \l_tmpa_str } {
1615       \bool_set_true:N \l_tmpa_bool

```

```

1616     \str_set:Nx \l_stex_get_symbol_uri_str {
1617       \prop_item:Nn \l_stex_current_module_prop { ns } ?
1618       \prop_item:Nn \l_stex_current_module_prop { name } ? #1
1619     }
1620   }
1621 }
1622 \bool_if:NF \l_tmpa_bool {
1623   \tl_set:Nn \l_tmpa_tl {
1624     \msg_set:nnn{stex}{error/unknownsymbol}{
1625       No~symbol~#1~found!
1626     }
1627     \msg_error:nn{stex}{error/unknownsymbol}
1628   }
1629   \str_set:Nn \l_tmpa_str { #1 }
1630   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1631   \seq_map_inline:Nn \l_stex_all_symbols_seq {
1632     \str_set:Nn \l_tmpb_str { ##1 }
1633     \str_if_eq:eeT { \l_tmpa_str } {
1634       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1635     } {
1636       \seq_map_break:n {
1637         \tl_set:Nn \l_tmpa_tl {
1638           \str_set:Nn \l_stex_get_symbol_uri_str {
1639             ##1
1640           }
1641         }
1642       }
1643     }
1644   }
1645   \l_tmpa_tl
1646 }
1647 }
1648
1649 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
1650   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
1651   { \tl_tail:N \l_tmpa_tl }
1652   \tl_if_single:NTF \l_tmpa_tl {
1653     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
1654       \exp_after:wN \str_set:Nn \exp_after:wN
1655       \l_stex_get_symbol_uri_str \l_tmpa_tl
1656     }{
1657       % TODO
1658       % tail is not a single group
1659     }
1660   }{
1661     % TODO
1662     % tail is not a single group
1663   }
1664 }

```

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

4.7 Notations

```

1665 <@@=stex_notation>

notation arguments:
1666 \keys_define:nn { stex / notation } {
1667   lang .tl_set_x:N = \l__stex_notation_lang_str ,
1668   variant .tl_set_x:N = \l__stex_notation_variant_str ,
1669   prec .tl_set_x:N = \l__stex_notation_prec_str ,
1670   op .tl_set:N = \l__stex_notation_op_tl ,
1671   unknown .code:n = \str_set:Nx
1672     \l__stex_notation_variant_str \l_keys_key_str
1673 }
1674
1675 \cs_new_protected:Nn \__stex_notation_args:n {
1676   \str_clear:N \l__stex_notation_lang_str
1677   \str_clear:N \l__stex_notation_variant_str
1678   \str_clear:N \l__stex_notation_prec_str
1679   \tl_clear:N \l__stex_notation_op_tl
1680
1681   \keys_set:nn { stex / notation } { #1 }
1682
1683   \str_set:Nx \l__stex_notation_lang_str \l__stex_notation_lang_str
1684   \str_set:Nx \l__stex_notation_variant_str \l__stex_notation_variant_str
1685   \str_set:Nx \l__stex_notation_prec_str \l__stex_notation_prec_str
1686 }

```

\notation

```

1687 \NewDocumentCommand \notation { 0{} m } {
1688   \__stex_notation_args:n { #1 }
1689   \tl_clear:N \l_stex_symdecl_definiens_tl
1690   \stex_get_symbol:n { #2 }
1691   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
1692 }

```

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

\stex_notation_do:nn

```

1693 \cs_new_protected:Nn \stex_notation_do:nn {
1694   \prop_set_eq:Nc \l_tmpa_prop {
1695     g_stex_symdecl_ #1 _prop
1696   }
1697
1698   \prop_clear:N \l_tmpb_prop
1699   \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
1700   \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
1701   \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
1702
1703   % precedences
1704   \seq_clear:N \l_tmpb_seq
1705   \exp_args:NNno
1706   \str_if_empty:NTF \l__stex_notation_prec_str {
1707     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1708     \int_compare:nNnTF \l_tmpa_str = 0 {
1709       \exp_args:NNnx
1710       \prop_put:Nno \l_tmpb_prop { opprec }
1711       { \neginfprec }

```

```

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

```



```

1766     \stex_term_math_oms:nnnn { #1 }
1767     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1768     { \prop_item:Nn \l_tmpb_prop { opprec } }
1769     { \exp_not:n { #2 } }
1770 }
1771 \__stex_notation_final:
1772 }{
1773   \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
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       \str_if_in:NnTF \l_tmpb_str B {
1786         \exp_args:Nne \use:nn
1787         {
1788           \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1789           \cs_set:Npn \l_tmpa_str } { {
1790             \stex_term_math_oma:nnnn { #1 }
1791             { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1792             { \prop_item:Nn \l_tmpb_prop { opprec } }
1793             { \exp_not:n { #2 } }
1794           } }
1795         }{
1796           \exp_args:Nne \use:nn
1797           {
1798             \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1799             \cs_set:Npn \l_tmpa_str } { {
1800               \stex_term_math_oma:nnnn { #1 }
1801               { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1802               { \prop_item:Nn \l_tmpb_prop { opprec } }
1803               { \exp_not:n { #2 } }
1804             } }
1805           }
1806         }
1807       }
1808       \int_zero:N \l_tmpa_int
1809       \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
1810       \prop_get:NnN \l_tmpb_prop { argprecs } \l_tmpa_seq
1811       \__stex_notation_arguments:
1812     }
1813   }

```

(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

```

1814 \cs_new_protected:Nn \__stex_notation_arguments: {
1815   \int_incr:N \l_tmpa_int

```

```

1816 \str_if_empty:NTF \l_tmpa_str {
1817   \__stex_notation_final:
1818 }{
1819   \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
1820   \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
1821   \str_if_eq:VnTF \l_tmpb_str a {
1822     \__stex_notation_argument_assoc:n
1823   }{
1824     \str_if_eq:VnTF \l_tmpb_str B {
1825       \__stex_notation_argument_assoc:n
1826     }{
1827       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1828       \tl_put_right:Nx \l_tmpa_tl {
1829         { \stex_term_math_arg:nnn
1830           { \int_use:N \l_tmpa_int }
1831           { \l_tmpb_str }
1832           { ####\int_use:N \l_tmpa_int }
1833         }
1834       }
1835       \__stex_notation_arguments:
1836     }
1837   }
1838 }
1839 }

```

(End definition for __stex_notation_arguments:.)

__stex_notation_argument_assoc:n

```

1840 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
1841   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1842   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
1843   \tl_put_right:Nx \l_tmpa_tl {
1844     { \stex_term_math_assoc_arg:nnnn
1845       { \int_use:N \l_tmpa_int }
1846       { \l_tmpb_str }
1847       \exp_args:No \exp_not:n
1848       {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
1849       { ####\int_use:N \l_tmpa_int }
1850     }
1851   }
1852   \__stex_notation_arguments:
1853 }

```

(End definition for __stex_notation_argument_assoc:n.)

__stex_notation_final: Called after processing all notation arguments

```

1854 \cs_new_protected:Nn \__stex_notation_final: {
1855   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
1856   \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
1857   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
1858   \exp_args:Nne \use:nn
1859   {
1860     \cs_generate_from_arg_count:cNnn {
1861       stex_notation_ \l_tmpa_str \c_hash_str

```

```

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

```

```

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

```

```

1970     } }
1971   }{
1972     \str_if_eq:VnTF \l_tmpb_str B {
1973       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
1974         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
1975         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
1976       } }
1977     }{
1978       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
1979         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
1980       } }
1981     }
1982   }
1983 }
1984 \stex_annotate_invisible:nnn { notationcomp }{}{
1985   $ \exp_args:Nno \use:nn { \use:c {
1986     stex_notation_ \prop_item:Nn \l_tmpb_prop { symbol }
1987     \c_hash_str \l__stex_notation_variant_str
1988     \c_hash_str \l__stex_notation_lang_str _cs
1989   } } { \l_tmpa_tl } $
1990 }
1991 }
1992 }
1993 }

```

(End definition for _stex_notation_final:.)

\symdef

```

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

```

```

2020
2021 \exp_args:NNo \str_set:Nn \l_stex_symdecl_name_str
2022   \l_stex_symdecl_name_str
2023 \exp_args:NNo \str_set:Nn \l_stex_symdecl_args_str
2024   \l_stex_symdecl_args_str
2025 \exp_args:NNo \str_set:Nn \l__stex_notation_lang_str
2026   \l__stex_notation_lang_str
2027 \exp_args:NNo \str_set:Nn \l__stex_notation_variant_str
2028   \l__stex_notation_variant_str
2029 \exp_args:NNo \str_set:Nn \l__stex_notation_prec_str
2030   \l__stex_notation_prec_str
2031 }
2032
2033 \NewDocumentCommand \symdef { 0{} m } {
2034   \__stex_notation_symdef_args:n { #1 }
2035   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2036   \stex_symdecl_do:n { #2 }
2037   \exp_args:Nx \stex_notation_do:nn {
2038     \prop_item:Nn \l_tmpa_prop { module } ?
2039     \prop_item:Nn \l_tmpa_prop { name }
2040   }
2041 }

```

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

`\stex_invoke_symbol:n` Invokes a semantic macro

```

2042 %\cs_new_protected:Nn \stex_invoke_symbol:n {
2043 %   \peek_charcode_remove:NTF ! {
2044 %     \stex_term_custom:nn { #1 } { }
2045 %   } {
2046 %     \if_mode_math:
2047 %       \exp_after:wN \__stex_notation_invoke_math:n
2048 %     \else:
2049 %       \exp_after:wN \__stex_notation_invoke_text:n
2050 %     \fi: { #1 }
2051 %   }
2052 %}
2053
2054 \cs_new_protected:Nn \stex_invoke_symbol:n {
2055   \if_mode_math:
2056     \exp_after:wN \__stex_notation_invoke_math:n
2057   \else:
2058     \exp_after:wN \__stex_notation_invoke_text:n
2059   \fi: { #1 }
2060 }

```

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

`__stex_notation_invoke_math:n`

```

2061 \cs_new_protected:Nn \__stex_notation_invoke_math:n {
2062   \peek_charcode_remove:NTF ! {
2063     \peek_charcode:NTF [ {
2064       \__stex_notation_invoke_op:nw { #1 }
2065     }{

```

```

2066     \__stex_notation_invoke_op:nw { #1 } []
2067   }
2068 }{
2069   \peek_charcode_remove:NTF * {
2070     \__stex_notation_invoke_text:n { #1 }
2071   }{
2072     \peek_charcode:NTF [ {
2073       \__stex_notation_invoke_math:nw { #1 }
2074     }{
2075       \__stex_notation_invoke_math:nw { #1 } []
2076     }
2077   }
2078 }
2079 }

```

(End definition for __stex_notation_invoke_math:n.)

__stex_notation_invoke_op:nw

```

2080 \cs_new_protected:Npn \__stex_notation_invoke_op:nw #1 [#2] {
2081   \__stex_notation_args:n { #2 }
2082   \cs_if_exist:cTF {
2083     stex_op_notation_ #1 \c_hash_str
2084     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str _cs
2085   }{
2086     \csname stex_op_notation_ #1 \c_hash_str
2087       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str _cs
2088     \endcsname
2089   }{
2090     % TODO throw error
2091   }
2092 }

```

(End definition for __stex_notation_invoke_op:nw.)

__stex_notation_invoke_math:nw

```

2093 \cs_new_protected:Npn \__stex_notation_invoke_math:nw #1 [#2] {
2094   \__stex_notation_args:n { #2 }
2095   \prop_set_eq:Nc \l_tmpa_prop {
2096     g_stex_symdecl_ #1 _prop
2097   }
2098   \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
2099   \seq_if_empty:NTF \l_tmpa_seq {
2100     \msg_set:nnn{stex}{error/nonotations}{
2101       Symbol~#1~used,~but~has~no~notations!
2102     }
2103     \msg_error:nn{stex}{error/nonotations}
2104   } {
2105     \seq_if_in:NxTF \l_tmpa_seq
2106       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2107       \use:c{
2108         stex_notation_ #1 \c_hash_str
2109         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2110         _cs
2111       }
2112     }{

```

```

2113 \str_if_empty:NTF \l__stex_notation_variant_str {
2114 \str_if_empty:NTF \l__stex_notation_lang_str {
2115 \seq_get_left:NN \l_tmpa_seq \l_tmpa_str
2116 \use:c{
2117 stex_notation_ #1 \c_hash_str \l_tmpa_str
2118 _cs
2119 }
2120 }{
2121 \msg_set:nnn{stex}{error/wrongnotation}{
2122 Symbol~#1~has~no~notation~
2123 \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2124 }
2125 \msg_error:nn{stex}{error/wrongnotation}
2126 }
2127 }{
2128 \msg_set:nnn{stex}{error/wrongnotation}{
2129 Symbol~#1~has~no~notation~
2130 \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2131 }
2132 \msg_error:nn{stex}{error/wrongnotation}
2133 }
2134 }
2135 }
2136 }

```

(End definition for `__stex_notation_invoke_math:nw`.)

`__stex_notation_invoke_text:n`

```

2137 \cs_new_protected:Nn \__stex_notation_invoke_text:n {
2138 \peek_charcode_remove:NTF ! {
2139 \stex_term_custom:nn { #1 } { }
2140 }{
2141 \prop_set_eq:Nc \l_tmpa_prop {
2142 g_stex_symdecl_ #1 _prop
2143 }
2144 \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2145 \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2146 }
2147 }

```

(End definition for `__stex_notation_invoke_text:n`.)

4.8 Terms

2148 `<@@=stex_term>`

Precedences:

```

\infprec
\neginfprec
\l__stex_term_downprec
2149 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2150 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2151 \int_new:N \l__stex_term_downprec
2152 \int_set_eq:NN \l__stex_term_downprec \infprec

```

(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
2153 \tl_set:Nn \l__stex_term_left_bracket_str (
2154 \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

2155 \cs_new_protected:Nn \__stex_term_maybe_brackets:nn {
2156   \int_compare:nNnTF { #1 } > \l__stex_term_downprec {
2157     \bool_if:NTF \l_stex_inarray_bool { #2 }{
2158       \dobrackets { #2 }
2159     }
2160   }{ #2 }
2161 }

(End definition for \__stex_term_maybe_brackets:nn.)

```

```

\dobrackets

2162 %\RequirePackage{scalerel}
2163 \cs_new_protected:Npn \dobrackets #1 {
2164   %\ThisStyle{\if D\m@switch
2165   %   \exp_args:Nnx \use:nn
2166   %   { \exp_after:wN \left\l__stex_term_left_bracket_str #1 }
2167   %   { \exp_not:N\right\l__stex_term_right_bracket_str }
2168   %   \else
2169   %   \exp_args:Nnx \use:nn
2170   %   { \l__stex_term_left_bracket_str #1 }
2171   %   { \l__stex_term_right_bracket_str }
2172   %\fi}
2173 }

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

```

```

\withbrackets

2174 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2175   \exp_args:Nnx \use:nn
2176   {
2177     \tl_set:Nx \l__stex_term_left_bracket_str { #1 }
2178     \tl_set:Nx \l__stex_term_right_bracket_str { #2 }
2179     #3
2180   }
2181   {
2182     \tl_set:Nn \exp_not:N \l__stex_term_left_bracket_str
2183     {\l__stex_term_left_bracket_str}
2184     \tl_set:Nn \exp_not:N \l__stex_term_right_bracket_str
2185     {\l__stex_term_right_bracket_str}
2186   }
2187 }

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

```

```

\STEXinvisible

2188 \cs_new_protected:Npn \STEXinvisible #1 {
2189   \stex_annotate_invisible:n { #1 }
2190 }

```

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

OMDOC terms:

`_stex_term_math_oms:nnnn`

```

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

```

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

`_stex_term_math_oma:nnnn`

```

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

```

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

`_stex_term_math_omb:nnnn`

```

2213 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2214   \stex_annotate:nnn{ OMBIND }{ #2 }{
2215     \stex_highlight_term:nn { #1 } { #3 }
2216   }
2217 }
2218
2219 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2220   \_stex_term_maybe_brackets:nn { #3 }{
2221     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2222   }
2223 }

```

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

`_stex_term_math_arg:nnn`

```

2224 \cs_new_protected:Nn \_stex_term_arg:nn {
2225   \stex_unhighlight_term:n {
2226     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2227   }
2228 }

```

```

2229 \cs_new_protected:Nn \stex_term_math_arg:nnn {
2230   \exp_args:Nnx \use:nn
2231     { \int_set:Nn \l__stex_term_downprec { #2 }
2232       \stex_term_arg:nn { #1 }{ #3 }
2233     }
2234     { \int_set:Nn \exp_not:N \l__stex_term_downprec { \int_use:N \l__stex_term_downprec } }
2235 }

```

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

`\stex_term_math_assoc_arg:nnnn`

```

2236 \cs_new_protected:Nn \stex_term_math_assoc_arg:nnnn {
2237   \seq_set_split:Nnn \l_tmpa_seq , { #4 }
2238   \int_compare:nNnTF { \seq_count:N \l_tmpa_seq } < 2 {
2239     \tl_set:Nn \l_tmpa_tl { #4 }
2240   }{
2241     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
2242     \seq_reverse:N \l_tmpa_seq
2243     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_tl
2244     \tl_set:No \l_tmpa_tl { \l_tmpb_tl }
2245
2246     \seq_map_inline:Nn \l_tmpa_seq {
2247       \exp_args:NNo \tl_set:No \l_tmpa_tl {
2248         \exp_args:Nno
2249           \l_tmpa_cs { ##1 } \l_tmpa_tl
2250       }
2251     }
2252
2253   }
2254   \exp_args:Nnno
2255   \stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2256 }

```

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

`\stex_term_custom:nn`

```

2257 \cs_new_protected:Nn \stex_term_custom:nn {
2258   \str_set:Nn \l__stex_term_custom_uri { #1 }
2259   \str_set:Nn \l_tmpa_str { #2 }
2260   \tl_clear:N \l_tmpa_tl
2261   \int_zero:N \l_tmpa_int
2262   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2263   \__stex_term_custom_loop:
2264 }

```

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

`__stex_term_custom_loop:`

```

2265 \cs_new_protected:Nn \__stex_term_custom_loop: {
2266   \bool_set_false:N \l_tmpa_bool
2267   \bool_while_do:nn {
2268     \str_if_eq_p:ee X {
2269       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2270     }
2271   }{

```

```

2272 \int_incr:N \l_tmpa_int
2273 }
2274
2275 \peek_charcode:NTF [ {
2276   % notation/text component
2277   \__stex_term_custom_component:w
2278 } {
2279   \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2280     % all arguments read => finish
2281     \__stex_term_custom_final:
2282   } {
2283     % arguments missing
2284     \peek_charcode_remove:NTF * {
2285       % invisible, specific argument position or both
2286       \peek_charcode:NTF [ {
2287         % visible specific argument position
2288         \__stex_term_custom_arg:wn
2289       } {
2290         % invisible
2291         \peek_charcode_remove:NTF * {
2292           % invisible specific argument position
2293           \__stex_term_custom_arg_inv:wn
2294         } {
2295           % invisible next argument
2296           \__stex_term_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2297         }
2298       }
2299     } {
2300       % next normal argument
2301       \__stex_term_custom_arg:wn [ \l_tmpa_int + 1 ]
2302     }
2303   }
2304 }
2305 }

```

(End definition for __stex_term_custom_loop:.)

__stex_term_custom_arg_inv:wn

```

2306 \cs_new_protected:Npn \__stex_term_custom_arg_inv:wn [ #1 ] #2 {
2307   \bool_set_true:N \l_tmpa_bool
2308   \__stex_term_custom_arg:wn [ #1 ] { #2 }
2309 }

```

(End definition for __stex_term_custom_arg_inv:wn.)

__stex_term_custom_arg:wn

```

2310 \cs_new_protected:Npn \__stex_term_custom_arg:wn [ #1 ] #2 {
2311   \str_set:Nx \l_tmpb_str {
2312     \str_item:Nn \l_tmpa_str { #1 }
2313   }
2314   \str_case:VnTF \l_tmpb_str {
2315     { X } { } % TODO throw error ?
2316     { i } { \__stex_term_custom_set_X:n { #1 } }
2317     { b } { \__stex_term_custom_set_X:n { #1 } }
2318     { a } { \__stex_term_custom_set_X:n { #1 } } % TODO ?

```

```

2319     { B } { \_stex_term_custom_set_X:n { #1 } } % TODO ?
2320 }{}{
2321   % TODO throw error
2322 }
2323
2324 \bool_if:nTF \l_tmpa_bool {
2325   \tl_put_right:Nx \l_tmpa_tl {
2326     \stex_annotate_invisible:n {
2327       \_stex_term_arg:nn { \int_eval:n { #1 } }
2328       \exp_not:n { { #2 } }
2329     }
2330   }
2331 } {
2332   \tl_put_right:Nx \l_tmpa_tl {
2333     \_stex_term_arg:nn { \int_eval:n { #1 } }
2334     \exp_not:n { { #2 } }
2335   }
2336 }
2337
2338 \_stex_term_custom_loop:
2339 }

```

(End definition for _stex_term_custom_arg:wn.)

_stex_term_custom_set_X:n

```

2340 \cs_new_protected:Nn \_stex_term_custom_set_X:n {
2341   \str_set:Nx \l_tmpa_str {
2342     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
2343     X
2344     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
2345   }
2346 }

```

(End definition for _stex_term_custom_set_X:n.)

_stex_term_custom_component:

```

2347 \cs_new_protected:Npn \_stex_term_custom_component:w [ #1 ] {
2348   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
2349   \_stex_term_custom_loop:
2350 }

```

(End definition for _stex_term_custom_component:.)

_stex_term_custom_final:

```

2351 \cs_new_protected:Nn \_stex_term_custom_final: {
2352   \int_compare:nNnTF \l_tmpb_int = 0 {
2353     \exp_args:Nnno \_stex_term_oms:nnn
2354   }{
2355     \str_if_in:NnTF \l_tmpa_str {b} {
2356       \exp_args:Nnno \_stex_term_ombind:nnn
2357     } {
2358       \exp_args:Nnno \_stex_term_oma:nnn
2359     }
2360   }
2361   { \l__stex_term_custom_uri } { \l__stex_term_custom_uri } { \l_tmpa_tl }
2362 }

```

(End definition for `_stex_term_custom_final:`)

```

\symref
\symname 2363 \NewDocumentCommand \symref { m m }{
2364   \STEXsymbol{#1}![#2]
2365 }
2366
2367 \keys_define:nn { stex / symname } {
2368   post      .tl_set_x:N    = \l_stex_symname_post_str
2369 }
2370
2371 \cs_new_protected:Nn \stex_symname_args:n {
2372   \str_clear:N \l_stex_symname_post_str
2373   \keys_set:nn { stex / symname } { #1 }
2374   \exp_args:NNno \str_set:Nn \l_stex_symname_post_str
2375     \l_stex_symname_post_str
2376 }
2377
2378 \NewDocumentCommand \symname { 0{} m }{
2379   \stex_symname_args:n { #1 }
2380   \stex_get_symbol:n { #2 }
2381   \str_set:Nx \l_tmpa_str {
2382     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
2383   }
2384   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
2385   \exp_args:NNx \use:nn
2386   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }![
2387     \l_tmpa_str \l_stex_symname_post_str
2388   ] }
2389 }

```

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

4.9 Notation Components

```

2390 <@@=stex_notationcomps>

```

`\stex_highlight_term:nn`

```

2391 \latexml_if:F {
2392   \scalatex_if:F{
2393     % \RequirePackage{pdfcomment}
2394   }
2395 }
2396
2397 \str_new:N \l__stex_notationcomps_highlight_uri_str
2398 \cs_new_protected:Nn \stex_highlight_term:nn {
2399   \exp_args:Nnx
2400   \use:nn {
2401     \str_set:Nx \l__stex_notationcomps_highlight_uri_str { #1 }
2402     #2
2403   } {
2404     \str_set:Nx \exp_not:N \l__stex_notationcomps_highlight_uri_str
2405       { \l__stex_notationcomps_highlight_uri_str }
2406   }

```

```

2407 }
2408
2409 \cs_new_protected:Nn \stex_unhighlight_term:n {
2410 % \latexml_if:TF {
2411 % #1
2412 % } {
2413 % \scalatex_if:TF {
2414 % #1
2415 % } {
2416 #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
2417 % }
2418 % }
2419 }

```

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

```

\comp
\@comp
\@defemph
2420 \cs_new_protected:Npn \comp #1 {
2421 \str_if_empty:NF \l__stex_notationcomps_highlight_uri_str {
2422 \scalatex_if:TF {
2423 \stex_annotate:nnn { comp }{ \l__stex_notationcomps_highlight_uri_str }{ #1 }
2424 }{
2425 \exp_args:Nnx \@comp { #1 } { \l__stex_notationcomps_highlight_uri_str }
2426 }
2427 }
2428 }
2429
2430 \cs_new_protected:Npn \@comp #1 #2 {
2431 % \pdftooltip {
2432 \textcolor{blue}{#1}
2433 % } { #2 }
2434 }
2435
2436 \cs_new_protected:Npn \@defemph #1 #2 {
2437 % \pdftooltip {
2438 \textbf{\textcolor{magenta}{#1}}
2439 % } { #2 }
2440 }

```

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

`\ellipses`

```

2441 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix
\parrayline
\parraycell
2442 \bool_new:N \l_stex_inparray_bool
2443 \bool_set_false:N \l_stex_inparray_bool
2444 \NewDocumentCommand \parray { m m } {
2445 \begingroup
2446 \bool_set_true:N \l_stex_inparray_bool
2447 \begin{array}{#1}
2448 #2
2449 \end{array}

```

```

2450 \endgroup
2451 }
2452
2453 \NewDocumentCommand \prmatrix { m } {
2454 \begin{group}
2455 \bool_set_true:N \l_stex_inarray_bool
2456 \begin{matrix}
2457 #1
2458 \end{matrix}
2459 \end{group}
2460 }
2461
2462 \def \parrayline #1 #2 {
2463 #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
2464 }
2465
2466 \def \parraycell #1 {
2467 #1 \bool_if:NT \l_stex_inarray_bool {\&}
2468 }

```

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

4.10 Structural Features

```

2469 <@@=stex_features>

```

symboldoc

```

2470 \NewDocumentEnvironment{symboldoc}{ m }{
2471 \seq_set_split:Nnn \l_tmpa_seq , { #1 }
2472 \seq_clear:N \l_tmpb_seq
2473 \seq_map_inline:Nn \l_tmpa_seq {
2474 \stex_get_symbol:n { ##1 }
2475 \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
2476 \l_stex_get_symbol_uri_str
2477 }
2478 }
2479 \par
2480 \exp_args:Nnnx
2481 \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
2482 }{
2483 \end{stex_annotate_env}
2484 }

```

STEXdefinition

```

2485
2486 \NewDocumentCommand \stex_definiendum:w { 0{} m m } {
2487 \stex_get_symbol:n { #2 }
2488 \scalatex_if:TF {
2489 \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { #3 }
2490 } {
2491 \exp_args:Nnx \@defemph { #3 } { \l_stex_get_symbol_uri_str }
2492 }
2493 }
2494 \NewDocumentCommand \stex_definame:w { 0{} m } {

```



```

2495 % TODO: root
2496 \stex_get_symbol:n { #2 }
2497 \str_set:Nx \l_tmpa_str {
2498   \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
2499 }
2500 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
2501 \scalatex_if:TF {
2502   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
2503     \l_tmpa_str
2504   }
2505 } {
2506   \@defemph {
2507     \l_tmpa_str
2508   } { \l_stex_get_symbol_uri_str }
2509 }
2510 }
2511
2512 \cs_new_protected:Nn \__stex_features_defi_begin:n {
2513   \let\definiendum\stex_definiendum:w
2514   \let\definame\stex_definame:w
2515   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
2516   \seq_clear:N \l_tmpb_seq
2517   \seq_map_inline:Nn \l_tmpa_seq {
2518     \stex_get_symbol:n { ##1 }
2519     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
2520       \l_stex_get_symbol_uri_str
2521     }
2522   }
2523   \exp_args:Nnnx
2524   \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
2525 }
2526
2527 \cs_new_protected:Nn \__stex_features_defi_end: {
2528   \end{stex_annotate_env}
2529 }
2530
2531 \NewDocumentEnvironment{STEXdefinition}{ m }{
2532   \__stex_features_defi_begin:n { #1 }
2533 }{
2534   \__stex_features_defi_end:
2535 }

```

\setSTEXdefinition

```

2536 \cs_new_protected:Npn \setSTEXdefinition #1 {
2537   \AddToHook{env/#1/before}[stex]{\__stex_features_defi_begin:n{}}
2538   \AddToHook{env/#1/after}[stex]{\__stex_features_defi_end:}
2539 }

```

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

structural@feature

```

2540
2541 \NewDocumentEnvironment{structural@feature}{ m m m }{
2542   \stex_if_in_module:F {

```

```

2543 \msg_set:nnn{stex}{error/nomodule}{
2544   Structural~Feature~has~to~occur~in~a~module:\\
2545   Feature~#2~of~type~#1\\
2546   In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
2547 }
2548 \msg_error:nn{stex}{error/nomodule}
2549 }
2550
2551 \str_set:Nx \l_stex_module_name_str {
2552   \prop_item:Nn \l_stex_current_module_prop
2553   { name } / #2 - feature
2554 }
2555
2556
2557 \str_clear:N \l_tmpa_str
2558 \seq_clear:N \l_tmpa_seq
2559 \tl_clear:N \l_tmpa_tl
2560 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
2561   origname = #2,
2562   name      = \l_stex_module_name_str ,
2563   ns        = \l_stex_module_ns_str ,
2564   imports   = \exp_not:o { \l_tmpa_seq } ,
2565   constants = \exp_not:o { \l_tmpa_seq } ,
2566   content   = \exp_not:o { \l_tmpa_tl } ,
2567   file      = \exp_not:o { \g_stex_currentfile_seq } ,
2568   lang      = \l_stex_module_lang_str ,
2569   sig       = \l_tmpa_str ,
2570   meta      = \l_tmpa_str ,
2571   feature   = #1 ,
2572 }
2573
2574 \stex_if_smsmode:TF {
2575   \stex_smsmode_set_codes:
2576 } {
2577   \begin{stex_annotate_env}{ feature:#1 }{}
2578   \stex_annotate_invisible:nnn{header}{}{ #3 }
2579 }
2580 }{
2581   \str_set:Nx \l_tmpa_str {
2582     c_stex_feature_
2583     \prop_item:Nn \l_stex_current_module_prop { ns } ?
2584     \prop_item:Nn \l_stex_current_module_prop { name }
2585     _prop
2586   }
2587   \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
2588   \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
2589   \stex_if_smsmode:TF {
2590     \exp_args:Nx \stex_addtosms:n {
2591       \prop_gset_from_keyval:cn {
2592         c_stex_feature_
2593         \prop_item:Nn \l_stex_current_module_prop { ns } ?
2594         \prop_item:Nn \l_stex_current_module_prop { name }
2595         _prop
2596       } {

```

```

2597         origname = #2,
2598         name      = \prop_item:cn { \l_tmpa_str } { name } ,
2599         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
2600         imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
2601         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
2602         content   = \prop_item:cn { \l_tmpa_str } { content } ,
2603         file      = \prop_item:cn { \l_tmpa_str } { file } ,
2604         lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
2605         sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
2606         meta      = \prop_item:cn { \l_tmpa_str } { meta } ,
2607         feature   = \prop_item:cn { \l_tmpa_str } { feature }
2608     }
2609 }
2610 { {
2611     \end{stex_annotate_env}
2612 }
2613 }
2614

```

structure

```

2615
2616 \prop_new:N \l_stex_all_structures_prop
2617
2618 \keys_define:nn { stex / features / structure } {
2619     name .tl_set_x:N = \l__stex_features_structure_name_str ,
2620 }
2621
2622 \cs_new_protected:Nn \__stex_features_structure_args:n {
2623     \str_clear:N \l__stex_features_structure_name_str
2624     \keys_set:nn { stex / features / structure } { #1 }
2625     \exp_args:NNo \str_set:Nn \l__stex_features_structure_name_str
2626         \l__stex_features_structure_name_str
2627 }
2628
2629 %\stex_new_feature:nnnn { structure } { 0{ } m } {
2630 % \__stex_features_structure_args:n { ##1 }
2631 % \str_if_empty:NT \l__stex_features_structure_name_str {
2632 % \str_set:Nx \l__stex_features_structure_name_str { ##2 }
2633 % }
2634 %} {
2635 %
2636 %}
2637
2638 \NewDocumentEnvironment{structure}{ 0{ } m }{
2639     \__stex_features_structure_args:n { #1 }
2640     \str_if_empty:NT \l__stex_features_structure_name_str {
2641         \str_set:Nx \l__stex_features_structure_name_str { #2 }
2642     }
2643     \exp_args:Nnnx
2644     \begin{structural@feature}{ structure }
2645         { \l__stex_features_structure_name_str }{}
2646         \seq_clear:N \l_tmpa_seq
2647         \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
2648

```

```

2649 }{
2650   \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
2651   \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
2652   \str_set:Nx \l_tmpa_str {
2653     \prop_item:Nn \l_stex_current_module_prop { ns } ?
2654     \prop_item:Nn \l_stex_current_module_prop { name }
2655   }
2656   \seq_map_inline:Nn \l_tmpa_seq {
2657     \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
2658   }
2659   \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
2660   \exp_args:Nnx
2661   \AddToHookNext { env / structure / after }{
2662     \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}}{
2663       \_stex_term_math_oms:nnnn { \l_tmpa_str }{}{0}{}
2664     }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
2665     \STEXexport {
2666       \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2667       {\prop_item:Nn \l_stex_current_module_prop { origname }}
2668       {\l_tmpa_str}
2669       \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2670       {#2}{\l_tmpa_str}
2671       % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
2672       %   \prop_item:Nn \l_stex_current_module_prop { origname },
2673       %   \l_tmpa_str
2674       % }
2675       % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
2676       %   #2,\l_tmpa_str
2677       % }
2678       % \tl_set:cx { #2 } {
2679       %   \stex_invoke_structure:n { \l_tmpa_str }
2680     }
2681   }
2682
2683   \end{structural@feature}
2684   % \g_stex_last_feature_prop
2685 }

```

\instantiate

```

2686 \seq_new:N \l__stex_features_structure_field_seq
2687 \str_new:N \l__stex_features_structure_field_str
2688 \str_new:N \l__stex_features_structure_def_tl
2689 \prop_new:N \l__stex_features_structure_prop
2690 \NewDocumentCommand \instantiate { m O{} m }{
2691   \stex_smsmode_set_codes:
2692   \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
2693   \prop_set_eq:Nc \l__stex_features_structure_prop {
2694     c_stex_feature_\l_tmpa_str _prop
2695   }
2696   \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
2697   \seq_map_inline:Nn \l__stex_features_structure_field_seq {
2698     \seq_set_split:Nnn \l_tmpa_seq[=]{ ##1 }
2699     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
2700       \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl

```

```

2701 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
2702 {!} \l_tmpa_tl
2703 \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
2704 \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
2705 \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
2706 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
2707 }{
2708 \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
2709 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
2710 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
2711 \l_tmpa_tl
2712 \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
2713 \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
2714 \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
2715 }{
2716 \tl_clear:N \l_tmpb_tl
2717 }
2718 }
2719 }{
2720 \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
2721 \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
2722 \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
2723 \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
2724 \tl_clear:N \l_tmpa_tl
2725 }{
2726 % TODO throw error
2727 }
2728 }
2729 % \l_tmpa_str: name
2730 % \l_tmpa_tl: definiens
2731 % \l_tmpb_tl: notation
2732 \tl_if_empty:NT \l__stex_features_structure_field_str {
2733 % TODO throw error
2734 }
2735 \str_clear:N \l_tmpb_str
2736
2737 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
2738 \seq_map_inline:Nn \l_tmpa_seq {
2739 \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
2740 \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
2741 \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
2742 \seq_map_break:n {
2743 \str_set:Nn \l_tmpb_str { ####1 }
2744 }
2745 }
2746 }
2747 \prop_get:cnN { g_stex_symdecl_ \l_tmpb_str _prop } {args}
2748 \l_tmpb_str
2749
2750 \tl_if_empty:NNT \l_tmpb_tl {
2751 \tl_if_empty:NF \l_tmpa_tl {
2752 \exp_args:Nx \use:n {
2753 \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fe
2754 }

```

```

2755     }
2756   }{
2757     \tl_if_empty:NTF \l_tmpa_tl {
2758       \exp_args:Nx \use:n {
2759         \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\exp_after:wN\exp_after:wN
2760       }
2761     }
2762   }{
2763     \exp_args:Nx \use:n {
2764       \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_features_structure_field_str}
2765       \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
2766     }
2767   }
2768 }
2769 % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
2770 % \prop_item:Nn \l_stex_current_module_prop {name} ?
2771 % #3/\l__stex_features_structure_field_str
2772 % \par
2773 % \expandafter\present\csname
2774 %   g_stex_symdecl_
2775 %   \prop_item:Nn \l_stex_current_module_prop {ns} ?
2776 %   \prop_item:Nn \l_stex_current_module_prop {name} ?
2777 %   #3/\l__stex_features_structure_field_str
2778 %   _prop
2779 % \endcsname
2780 }
2781
2782 \tl_clear:N \l__stex_features_structure_def_tl
2783
2784 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
2785 \seq_map_inline:Nn \l_tmpa_seq {
2786   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
2787   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
2788   \exp_args:Nx \use:n {
2789     \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
2790
2791     }
2792   }
2793 }
2794
2795 \prop_if_exist:cF {
2796   g_stex_symdecl_
2797   \prop_item:Nn \l_stex_current_module_prop {ns} ?
2798   \prop_item:Nn \l_stex_current_module_prop {name} ?
2799   #3/\l_tmpa_str
2800   _prop
2801 }{
2802   \prop_get:cnN { g_stex_symdecl_ ##1 _prop } {args}
2803   \l_tmpb_str
2804   \exp_args:Nx \use:n {
2805     \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
2806   }
2807 }
2808

```

```

2809 \symdecl*[type={\STEXsymbol{module-type}}{
2810   \_stex_term_math_oms:nnnn {
2811     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
2812     \prop_item:Nn \l__stex_features_structure_prop {name}
2813     }{}{0}{}
2814   }{}{#3}
2815
2816 % TODO: -> sms file
2817
2818 \tl_set:cx{ #3 }{
2819   \stex_invoke_structure:nnn {
2820     \prop_item:Nn \l_stex_current_module_prop {ns} ?
2821     \prop_item:Nn \l_stex_current_module_prop {name} ? #3
2822   } {
2823     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
2824     \prop_item:Nn \l__stex_features_structure_prop {name}
2825   }
2826 }
2827
2828 }

```

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

\stex_invoke_structure:nnn

```

2829 % #1: URI of the instance
2830 % #2: URI of the instantiated module
2831 \cs_new_protected:Nn \stex_invoke_structure:nnn {
2832   \tl_if_empty:nTF{ #3 }{
2833     \prop_set_eq:Nc \l__stex_features_structure_prop {
2834       c_stex_feature_ #2 _prop
2835     }
2836     \tl_clear:N \l_tmpa_tl
2837     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
2838     \seq_map_inline:Nn \l_tmpa_seq {
2839       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
2840       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
2841       \cs_if_exist:cT {
2842         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
2843       }{
2844         \tl_if_empty:NF \l_tmpa_tl {
2845           \tl_put_right:Nn \l_tmpa_tl {,}
2846         }
2847         \tl_put_right:Nx \l_tmpa_tl {
2848           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
2849         }
2850       }
2851     }
2852     \exp_args:No \mathstruct \l_tmpa_tl
2853   }{
2854     \stex_invoke_symbol:n{#1/#3}
2855   }
2856 }

```

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

4.11 Put these somewhere

\MSC

```
2857 \NewDocumentCommand \MSC {m} {
2858   % TODO
2859 }
```

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

```

2860 \@ifpackageloaded{tikzinput}{
2861   \RequirePackage{stex-tikzinput}
2862 }{}
2863
2864 \AddToHook{begindocument}{
2865   \input{stex-metatheory}
2866 }
2867 \end{package}

```

4.12 Metatheory

The default meta theory for an \SIX 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.

```

2868 \begin{document}
2869 \ExplSyntaxOn
2870 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
2871 \begin{@module}[ns=\c_stex_metatheory_ns_str,meta=NONE]{Metatheory}
2872 \ExplSyntaxOff
2873
2874 % is-a (a:A, a \in A, a is an A, etc.)
2875 \symdecl[args=ai]{isa}
2876 \notation[typed]{isa}{#1 \comp: #2}{#1 \comp, #2}
2877 \notation[in]{isa}{#1 \comp\in #2}{#1 \comp, #2}
2878 \notation[pred]{isa}{#2\comp(#1 \comp)}{#1 \comp, #2}
2879
2880 % bind (\forall, \Pi, \lambda etc.)
2881 \symdecl[args=Bi]{bind}
2882 \notation[forall]{bind}{\comp\forall #1.;#2}{#1 \comp, #2}
2883 \notation[\Pi]{bind}{\comp\prod_{#1}#2}{#1 \comp, #2}
2884 \notation[deffun]{bind}{\comp( #1 \comp{} \; \; \to \; ) #2}{#1 \comp, #2}
2885
2886 % dummy variable
2887 \symdecl{dummyvar}
2888 \notation[underscore]{dummyvar}{\comp\_}
2889 \notation[dot]{dummyvar}{\comp\cdot}
2890 \notation[dot]{dummyvar}{\comp\cdot}
2891 \notation[dash]{dummyvar}{\comp{\rm --}}
2892

```



```

2893 %fromto (function space, Hom-set, implication etc.)
2894 \symdecl[args=ai]{fromto}
2895 \notation[xarrow]{fromto}{#1 \comp\to #2}{#1 \comp\times #2}
2896 \notation[arrow]{fromto}{#1 \comp\to #2}{#1 \comp\to #2}
2897
2898 % mapto (lambda etc.)
2899 %\symdecl[args=Bi]{mapto}
2900 %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
2901 %\notation[lambda]{mapto}{\comp\lambda #1 \comp.; #2}{#1 \comp, #2}
2902 %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp.; #2}{#1 \comp, #2}
2903
2904 % function/operator application
2905 \symdecl[args=ia]{apply}
2906 \notation[prec=0;0x\neginfprec,parens]{apply}{#1 \comp( #2 \comp)}{#1 \comp, #2}
2907 \notation[prec=0;0x\neginfprec,lambda]{apply}{#1 \; #2 }{#1 \; #2}
2908
2909 % ‘‘type’’ of all collections (sets, classes, types, kinds)
2910 \symdecl{collection}
2911 \notation[U]{collection}{\comp{\mathcal{U}}}
2912 \notation[set]{collection}{\comp{\textsf{Set}}}
2913
2914 % sequences
2915 \symdecl[args=1]{seqtype}
2916 \notation[kleene]{seqtype}{#1^{\comp\ast}}
2917
2918 \symdef[args=2,li]{sequence-index}{#1_{#2}}
2919 \notation[ui]{sequence-index}{#1^{#2}}
2920
2921 %\symdef[args=3,li]{sequence-from-to}{#1_{#2}\comp{\,\ellipses,}#1_{#3}}
2922 %\notation[ui]{sequence-from-to}{#1^{#2}\comp{\,\ellipses,}#1^{#3}}
2923 % ^ superceded by \aseqfromto and \livar/\uivar
2924
2925 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{\,\ellipses}}{#1\comp,#2}
2926 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{\,\ellipses\comp,}#2 }{#1\comp,#2}
2927 \symdef[args=aai,prec=nobrackets]{aseqfromtovia}{#1\comp{\,\ellipses\comp,}#2\comp{\,\ellipses}
2928
2929 % letin (‘‘let’’, local definitions, variable substitution)
2930 \symdecl[args=bii]{letin}
2931 \notation[let]{letin}{\comp{\{\rm let\}}\;#1\comp{=}\#2\; \comp{\{\rm in\}}\;#3}
2932 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
2933 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
2934
2935 % structures
2936 \symdecl*[args=1]{module-type}
2937 \notation{module-type}{\mathtt{MOD} #1}
2938 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
2939 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{#1 \comp, #2}
2940
2941 \STEXexport{
2942   \let\nappa\apply
2943   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
2944   \def\livar{\csname sequence-index\endcsname[li]}
2945   \def\uivar{\csname sequence-index\endcsname[ui]}
2946   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}

```

```

2947 \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
2948 }
2949
2950 \end{@module}
2951 \ExplSyntaxOff
2952 </metatheory>

```

4.13 Auxiliary Packages

4.13.1 tikzinput

```

2953 <*tikzinput>
2954 <@@=tikzinput>
2955 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
2956 \RequirePackage{l3keys2e}
2957
2958 \keys_define:nn { tikzinput } {
2959   image .bool_set:N = \c_tikzinput_image_bool,
2960   image .default:n = false ,
2961 }
2962
2963 \ProcessKeysOptions { tikzinput }
2964
2965 \bool_if:NTF \c_tikzinput_image_bool {
2966   \RequirePackage{graphicx}
2967
2968   \providecommand\usetikzlibrary[]{}
2969   \newcommand\tikzinput[2][]{\includegraphics[#1]{#2}}
2970 }{
2971   \RequirePackage{tikz}
2972   \RequirePackage{standalone}
2973
2974   \newcommand\tikzinput[2][] {
2975     \setkeys{Gin}{#1}
2976     \ifx \Gin@ewidth \Gin@exclamation
2977       \ifx \Gin@eheight \Gin@exclamation
2978         \input { #2 }
2979       \else
2980         \resizebox{!}{ \Gin@eheight }{
2981           \input { #2 }
2982         }
2983       \fi
2984     \else
2985       \ifx \Gin@eheight \Gin@exclamation
2986         \resizebox{ \Gin@ewidth }{!}{
2987           \input { #2 }
2988         }
2989       \else
2990         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
2991           \input { #2 }
2992         }
2993       \fi
2994     \fi
2995   }

```

```

2996 }
2997
2998 \newcommand \ctikzinput [2] [] {
2999   \begin{center}
3000     \tikzinput [#1] {#2}
3001   \end{center}
3002 }
3003
3004 \@ifpackageloaded{stex}{
3005   \RequirePackage{stex-tikzinput}
3006 }{}
3007 </tikzinput>
3008 <*stex-tikzinput>
3009 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
3010 \RequirePackage{stex}
3011 \RequirePackage{tikzinput}
3012
3013 % TODO
3014
3015 </stex-tikzinput>

```

4.13.2 sTeX1 Compatibility

```

3016 <*smglom>
3017 \RequirePackage{expl3,l3keys2e}
3018 \ProvidesExplClass{smglom}{2021/08/01}{1.9}{sTeX1 compatibility}
3019 \LoadClass[border=1px,varwidth]{standalone}
3020 \setlength\textwidth{15cm}
3021 %\g@addto@macro{\@parboxrestore}{\setlength\parskip{\baselineskip}}
3022 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
3023 \ProcessOptions
3024
3025 \RequirePackage{stex-compatibility}
3026 </smglom>
3027
3028 <*compat>
3029 <@@=stex_deprec>
3030 \ProvidesExplPackage{stex-compatibility}{2021/08/01}{1.9}{bla}
3031 %\RequirePackage[lang={de,en,ro,tr,fr}]{stex}
3032 \RequirePackage[lang=en]{stex}
3033
3034 \NewDocumentEnvironment { mhmodnl } { 0{} m m } {
3035   \msg_set:nnn{stex}{warning/deprecated}{
3036     \\\
3037     Environment~mhmodnl~is~deprected! \\\
3038     Please~update~module~#2~in~file~
3039     \stex_path_to_string:N \g_stex_currentfile_seq!
3040     \\\ \\\
3041   }
3042   \msg_warning:nn{stex}{warning/deprecated}
3043
3044   \begin{module}[#1,lang=#3]{#2}
3045     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
3046     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
3047     \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str

```

```

3048 \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
3049 \input { \stex_path_to_string:N \l_tmpa_seq / \l_tmpa_str.tex }
3050 } {
3051 \end{module}
3052 }
3053
3054 \NewDocumentEnvironment { modsig } { 0{ } m } {
3055 \stex_if_in_module:TF {
3056 \prop_get:NnN \l_stex_current_module_prop {name} \l_tmpa_str
3057 \str_set:Nn \l_tmpb_str { #2 }
3058 \str_if_eq:NNTF \l_tmpa_str \l_tmpb_str {
3059 \prop_set_eq:NN \l_modsig_old_module_prop \l_stex_current_module_prop
3060 \begin{@module}{modsig-#2}
3061 % \prop_set_eq:NN \l_stex_current_module_prop \l_modsig_old_module_prop
3062 } {
3063 \begin{@module}{#2}
3064 }
3065 } {
3066 \begin{@module}{#2}
3067 }
3068 }{
3069 \end{@module}
3070 \AddToHookNext { env / modsig / after }{
3071 \stex_if_in_module:T {
3072 \prop_get:NnN \l_stex_current_module_prop {name} \l_tmpa_str
3073 \str_set:Nn \l_tmpb_str { #2 }
3074 \str_if_eq:NNT \l_tmpa_str \l_tmpb_str {
3075 % \xdef \g_stex_module_after_group_tl {
3076 \stex_if_smsmode:TF {
3077 \exp_args:Nx
3078 \stex_add_to_current_module:n {
3079 \stex_debug:n{Activating-signature-of-#2}
3080 \exp_not:N \prop_item:cn { c_stex_module_
3081 \prop_item:Nn \l_stex_current_module_prop {ns} ?
3082 \prop_item:Nn \l_stex_current_module_prop {name}
3083 / modsig-#2_prop } { content }
3084 }
3085 }
3086 {
3087 \gdef \g_stex_modsig_after_group_tl {
3088 \stex_activate_module:n {
3089 \prop_item:Nn \l_stex_current_module_prop {ns} ?
3090 \prop_item:Nn \l_stex_current_module_prop {name}
3091 / modsig-#2
3092 }
3093
3094 \exp_args:Nx
3095 \stex_add_to_current_module:n {
3096 \stex_activate_module:n {
3097 \prop_item:Nn \l_stex_current_module_prop {ns} ?
3098 \prop_item:Nn \l_stex_current_module_prop {name}
3099 / modsig-#2
3100 }
3101 }

```

```

3102         }
3103         \aftergroup \g_stex_modsig_after_group_tl
3104     }
3105 }
3106 }
3107 }
3108 }
3109
3110 \cs_new_protected:Npn \gimport {
3111     \peek_charcode_remove:NTF * {
3112         \gimport_do:
3113     } {
3114         \gimport_do:
3115     }
3116 }
3117
3118 \NewDocumentCommand \gimport_do: { 0{ } m } {
3119     \msg_set:nnn{stex}{warning/deprecated}{
3120         \
3121         \c_backslash_str gimport-is-deprecated! \
3122         Please~use~\c_backslash_str importmodule[#1]{#2}~instead!~(in~file~
3123         \stex_path_to_string:N \g_stex_currentfile_seq)
3124         \
3125     }
3126     \msg_warning:nn{stex}{warning/deprecated}
3127     \importmodule[#1]{#2}
3128 }
3129
3130 \cs_new_protected:Npn \guse {
3131     \peek_charcode_remove:NTF * {
3132         \guse_do:
3133     } {
3134         \guse_do:
3135     }
3136 }
3137
3138 \NewDocumentCommand \guse_do: { 0{ } m } {
3139     \msg_set:nnn{stex}{warning/deprecated}{
3140         \
3141         \c_backslash_str guse-is-deprecated! \
3142         Please~use~\c_backslash_str usemodule[#1]{#2}~instead!~(in~file~
3143         \stex_path_to_string:N \g_stex_currentfile_seq)
3144         \
3145     }
3146     \msg_warning:nn{stex}{warning/deprecated}
3147     \usemodule[#1]{#2}
3148 }
3149
3150 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3151
3152 \cs_new_protected:Npn \symi {
3153     \peek_charcode_remove:NTF * {
3154         \symi_do:
3155     } {

```

```

3156     \syml_do:
3157   }
3158 }
3159
3160 \NewDocumentCommand \syml_do: { 0{} m } {
3161   \msg_set:nnn{stex}{warning/deprecated}{
3162     \\\
3163     \c_backslash_str syml~is-deprecated! \\\
3164     Please~use~\c_backslash_str symdecl[#1]{#2}~instead!~(in~file~
3165     \stex_path_to_string:N \g_stex_currentfile_seq)
3166     \\\ \\\
3167   }
3168   \msg_warning:nn{stex}{warning/deprecated}
3169   \symdecl*{#1}{#2}
3170 }
3171
3172 \cs_new_protected:Npn \syml {
3173   \peek_charcode_remove:NTF * {
3174     \syml_do:
3175   } {
3176     \syml_do:
3177   }
3178 }
3179
3180 \NewDocumentCommand \syml_do: { 0{} m m } {
3181   \msg_set:nnn{stex}{warning/deprecated}{
3182     \\\
3183     \c_backslash_str syml~is-deprecated! \\\
3184     Please~use~\c_backslash_str symdecl[#1]{#2-#3}~instead!~(in~file~
3185     \stex_path_to_string:N \g_stex_currentfile_seq)
3186     \\\ \\\
3187   }
3188   \msg_warning:nn{stex}{warning/deprecated}
3189   \symdecl*{#1}{#2-#3}
3190 }
3191
3192 \cs_new_protected:Npn \syml {
3193   \peek_charcode_remove:NTF * {
3194     \syml_do:
3195   } {
3196     \syml_do:
3197   }
3198 }
3199
3200 \NewDocumentCommand \syml_do: { 0{} m m m } {
3201   \msg_set:nnn{stex}{warning/deprecated}{
3202     \\\
3203     \c_backslash_str syml~is-deprecated! \\\
3204     Please~use~\c_backslash_str symdecl[#1]{#2-#3-#4}~instead!~(in~file~
3205     \stex_path_to_string:N \g_stex_currentfile_seq)
3206     \\\ \\\
3207   }
3208   \msg_warning:nn{stex}{warning/deprecated}
3209   \symdecl*{#1}{#2-#3-#4}

```

```

3210 }
3211
3212 \keys_define:nn { stex / deprec / defi } {
3213   name .tl_set_x:N = \l_tmpa_str
3214 }
3215
3216 \cs_new_protected:Npn \defi {
3217   \peek_charcode_remove:NTF * {
3218     \defi_do:
3219   } {
3220     \defi_do:
3221   }
3222 }
3223
3224 \NewDocumentCommand \defi_do: { 0{} m } {
3225   \str_clear:N \l_tmpa_str
3226   \keys_set:nn { stex / deprec / defi } { #1 }
3227
3228   \str_if_empty:NTF \l_tmpa_str {
3229     \msg_set:nnn{stex}{warning/deprecated}{
3230       \\\
3231       \c_backslash_str defi~is~deprecated! \\\
3232       Please~use~\c_backslash_str STExsymbol{#2}![#2]~instead!~(in~file~
3233       \stex_path_to_string:N \g_stex_currentfile_seq)
3234       \\\ \\\
3235     }
3236     \msg_warning:nn{stex}{warning/deprecated}
3237     \STExsymbol { #2 }![ \comp{#2} ]
3238   } {
3239     \msg_set:nnn{stex}{warning/deprecated}{
3240       \\\
3241       \c_backslash_str defi~is~deprecated! \\\
3242       Please~use~\c_backslash_str STExsymbol { \l_tmpa_str }[ #2 ]~instead!~(in~file~
3243       \stex_path_to_string:N \g_stex_currentfile_seq)
3244       \\\ \\\
3245     }
3246     \msg_warning:nn{stex}{warning/deprecated}
3247     \exp_args:No \STExsymbol { \l_tmpa_str }![ \comp{#2} ]
3248   }
3249 }
3250
3251
3252 \cs_new_protected:Npn \Defi {
3253   \peek_charcode_remove:NTF * {
3254     \Defi_do:
3255   } {
3256     \Defi_do:
3257   }
3258 }
3259
3260 \NewDocumentCommand \Defi_do: { 0{} m } {
3261   \str_clear:N \l_tmpa_str
3262   \keys_set:nn { stex / deprec / defi } { #1 }
3263

```

```

3264 \str_if_empty:NTF \l_tmpa_str {
3265   \msg_set:nnn{stex}{warning/deprecated}{
3266     \\\
3267     \c_backslash_str Defi~is~deprecated! \\\
3268     Please~use~\c_backslash_str STEXsymbol{#2}![\exp_after:wN \stex_capitalize:n #2]~inste
3269     \stex_path_to_string:N \g_stex_currentfile_seq)
3270   \\\
3271 }
3272 \msg_warning:nn{stex}{warning/deprecated}
3273 \STEXsymbol { #2 }![ \comp{\exp_after:wN \stex_capitalize:n #2} ]
3274 } {
3275   \msg_set:nnn{stex}{warning/deprecated}{
3276     \\\
3277     \c_backslash_str Defi~is~deprecated! \\\
3278     Please~use~\c_backslash_str STEXsymbol { \l_tmpa_str }[ \exp_after:wN \stex_capitalize
3279     \stex_path_to_string:N \g_stex_currentfile_seq)
3280     \\\
3281   }
3282   \msg_warning:nn{stex}{warning/deprecated}
3283   \exp_args:No \STEXsymbol { \l_tmpa_str }![ \comp{\exp_after:wN \stex_capitalize:n #2} ]
3284 }
3285 }
3286
3287 \cs_new_protected:Npn \adefi {
3288   \peek_charcode_remove:NTF * {
3289     \adefi_do:
3290   } {
3291     \adefi_do:
3292   }
3293 }
3294
3295 \NewDocumentCommand \adefi_do: { 0{} m m } {
3296   \str_clear:N \l_tmpa_str
3297   \keys_set:nn { stex / deprec / defi } { #1 }
3298
3299   \str_if_empty:NTF \l_tmpa_str {
3300     \msg_set:nnn{stex}{warning/deprecated}{
3301       \\\
3302       \c_backslash_str adefi~is~deprecated! \\\
3303       Please~use~\c_backslash_str STEXsymbol{#3}![#2]~instead!~(in~file~
3304       \stex_path_to_string:N \g_stex_currentfile_seq)
3305       \\\
3306     }
3307     \msg_warning:nn{stex}{warning/deprecated}
3308     \STEXsymbol { #3 }![ \comp{#2} ]
3309   } {
3310     \msg_set:nnn{stex}{warning/deprecated}{
3311       \\\
3312       \c_backslash_str adefi~is~deprecated! \\\
3313       Please~use~\c_backslash_str STEXsymbol { \l_tmpa_str }[ #2 ]~instead!~(in~file~
3314       \stex_path_to_string:N \g_stex_currentfile_seq)
3315       \\\
3316     }
3317     \msg_warning:nn{stex}{warning/deprecated}

```



```

3318     \exp_args:No \STEXsymbol { \l_tmpa_str }![ \comp{#2} ]
3319   }
3320 }
3321
3322 \cs_new_protected:Npn \defis {
3323   \peek_charcode_remove:NTF * {
3324     \defis_do:
3325   } {
3326     \defis_do:
3327   }
3328 }
3329
3330 \NewDocumentCommand \defis_do: { 0{ } m } {
3331   \str_clear:N \l_tmpa_str
3332   \keys_set:nn { stex / deprec / defi } { #1 }
3333
3334   \str_if_empty:NTF \l_tmpa_str {
3335     \msg_set:nnn{stex}{warning/deprecated}{
3336       \\\
3337       \c_backslash_str defis-is-deprecated! \\\
3338       Please~use~\c_backslash_str STEXsymbol{#2}![#2s]~instead!~(in~file~
3339       \stex_path_to_string:N \g_stex_currentfile_seq)
3340       \\\ \\\
3341     }
3342     \msg_warning:nn{stex}{warning/deprecated}
3343     \STEXsymbol { #2 }![ \comp{#2s} ]
3344   } {
3345     \msg_set:nnn{stex}{warning/deprecated}{
3346       \\\
3347       \c_backslash_str defis-is-deprecated! \\\
3348       Please~use~\c_backslash_str STEXsymbol { \l_tmpa_str }[ #2s ]~instead!~(in~file~
3349       \stex_path_to_string:N \g_stex_currentfile_seq)
3350       \\\ \\\
3351     }
3352     \msg_warning:nn{stex}{warning/deprecated}
3353     \exp_args:No \STEXsymbol { \l_tmpa_str }![ \comp{#2s} ]
3354   }
3355 }
3356
3357 \cs_new_protected:Npn \defii {
3358   \peek_charcode_remove:NTF * {
3359     \defii_do:
3360   } {
3361     \defii_do:
3362   }
3363 }
3364
3365 \NewDocumentCommand \defii_do: { 0{ } m m } {
3366   \str_clear:N \l_tmpa_str
3367   \keys_set:nn { stex / deprec / defi } { #1 }
3368   \str_if_empty:NTF \l_tmpa_str {
3369     \msg_set:nnn{stex}{warning/deprecated}{
3370       \\\
3371       \c_backslash_str defii-is-deprecated! \\\

```

```

3372     Please~use~\c_backslash_str STEXsymbol{#2~#3}![#2~#3]~instead!~(in~file~
3373     \stex_path_to_string:N \g_stex_currentfile_seq)
3374     \\\ \\\
3375   }
3376   \msg_warning:nn{stex}{warning/deprecated}
3377   \STEXsymbol { #2~#3 }![ \comp{#2~#3} ]
3378 } {
3379   \msg_set:nnn{stex}{warning/deprecated}{
3380     \\\
3381     \c_backslash_str defii~is~deprecated! \\\
3382     Please~use~\c_backslash_str STEXsymbol { \l_tmpa_str }[ #2~#3 ]~instead!~(in~file~
3383     \stex_path_to_string:N \g_stex_currentfile_seq)
3384     \\\ \\\
3385   }
3386   \msg_warning:nn{stex}{warning/deprecated}
3387   \exp_args:No \STEXsymbol { \l_tmpa_str }![ \comp{#2~#3} ]
3388 }
3389 }
3390
3391
3392 \cs_new_protected:Npn \defiis {
3393   \peek_charcode_remove:NTF * {
3394     \defiis_do:
3395   } {
3396     \defiis_do:
3397   }
3398 }
3399
3400 \NewDocumentCommand \defiis_do: { 0{} m m } {
3401   \str_clear:N \l_tmpa_str
3402   \keys_set:nn { stex / deprec / defi } { #1 }
3403   \str_if_empty:NTF \l_tmpa_str {
3404     \msg_set:nnn{stex}{warning/deprecated}{
3405       \\\
3406       \c_backslash_str defiis~is~deprecated! \\\
3407       Please~use~\c_backslash_str STEXsymbol{#2~#3}![#2~#3s]~instead!~(in~file~
3408       \stex_path_to_string:N \g_stex_currentfile_seq)
3409       \\\ \\\
3410     }
3411     \msg_warning:nn{stex}{warning/deprecated}
3412     \STEXsymbol { #2~#3 }![ \comp{#2~#3s} ]
3413   } {
3414     \msg_set:nnn{stex}{warning/deprecated}{
3415       \\\
3416       \c_backslash_str defiis~is~deprecated! \\\
3417       Please~use~\c_backslash_str STEXsymbol { \l_tmpa_str }[ #2~#3s ]~instead!~(in~file~
3418       \stex_path_to_string:N \g_stex_currentfile_seq)
3419       \\\ \\\
3420     }
3421     \msg_warning:nn{stex}{warning/deprecated}
3422     \exp_args:No \STEXsymbol { \l_tmpa_str }![ \comp{#2~#3s} ]
3423   }
3424 }
3425

```

```

3426
3427 \cs_new_protected:Npn \defiii {
3428   \peek_charcode_remove:NTF * {
3429     \defiii_do:
3430   } {
3431     \defiii_do:
3432   }
3433 }
3434
3435 \NewDocumentCommand \defiii_do: { 0{} m m m } {
3436   \str_clear:N \l_tmpa_str
3437   \keys_set:nn { stex / deprec / defi } { #1 }
3438   \str_if_empty:NTF \l_tmpa_str {
3439     \msg_set:nnn{stex}{warning/deprecated}{
3440       \\\
3441       \c_backslash_str defiii~is-deprecated! \\\
3442       Please~use~\c_backslash_str STEXsymbol{#2~#3~#4}![#2~#3~#4]~instead!~(in~file~
3443       \stex_path_to_string:N \g_stex_currentfile_seq)
3444       \\\ \\\
3445     }
3446     \msg_warning:nn{stex}{warning/deprecated}
3447     \STEXsymbol { #2~#3~#4 }![ \comp{#2~#3~#4} ]
3448   } {
3449     \msg_set:nnn{stex}{warning/deprecated}{
3450       \\\
3451       \c_backslash_str defiii~is-deprecated! \\\
3452       Please~use~\c_backslash_str STEXsymbol { \l_tmpa_str }[ #2~#3~#4 ]~instead!~(in~file~
3453       \stex_path_to_string:N \g_stex_currentfile_seq)
3454       \\\ \\\
3455     }
3456     \msg_warning:nn{stex}{warning/deprecated}
3457     \exp_args:No \STEXsymbol { \l_tmpa_str }![ \comp{#2~#3~#4} ]
3458   }
3459 }
3460
3461 %\RequirePackage[hyperref]{ntheorem}
3462 %\theoremstyle{plain}
3463 %\RequirePackage{amsthm}
3464
3465 \NewDocumentEnvironment {definition} { 0{} } {
3466   \begin{STEXdefinition}{}
3467 }{
3468   \end{STEXdefinition}
3469 }
3470 \keys_define:nn { stex / omtex } {
3471   id      .tl_set_x:N = \l_stex_omtext_id_str ,
3472   title   .tl_set_x:N = \l_stex_omtext_title_str ,
3473   type    .tl_set_x:N = \l_stex_omtext_type_tl ,
3474   for     .tl_set_x:N = \l_stex_omtext_for_tl ,
3475   from    .tl_set_x:N = \l_stex_omtext_from_tl ,
3476   start   .tl_set_x:N = \l_stex_omtext_start_str ,
3477 }
3478 \cs_new_protected:Nn \stex_omtext_args:n {
3479   \str_clear:N \l_stex_omtext_title_str

```

```

3480 \str_clear:N \l_stex_omtext_start_str
3481 \keys_set:nn { stex / omtext }{ #1 }
3482 \exp_args:NNo \str_set:Nn \l_stex_omtext_title_str
3483 \l_stex_omtext_title_str
3484 \exp_args:NNo \str_set:Nn \l_stex_omtext_start_str
3485 \l_stex_omtext_start_str
3486 }
3487 \NewDocumentEnvironment {omtext} { 0{ } } {
3488 \stex_omtext_args:n { #1 }
3489 \textbf{\str_if_empty:NTF \l_stex_omtext_start_str {
3490 \l_stex_omtext_title_str
3491 }{
3492 \l_stex_omtext_start_str :
3493 }}}
3494 }{
3495
3496 }
3497 \NewDocumentEnvironment {assertion} { 0{ } } {
3498
3499 }{
3500
3501 }
3502
3503 \NewDocumentCommand \inlinedef { m } {
3504 \begin{group}
3505 \let\definiendum\stex_definiendum:w
3506 \let\definame\stex_definame:w
3507 #1
3508 \end{group}
3509 }
3510
3511 \NewDocumentCommand \inlineass { m } { #1 }
3512
3513 \NewDocumentCommand \trefi { 0{ } m } {
3514 \str_set:Nn \l_tmpa_str { #1 }
3515 \str_if_empty:NTF \l_tmpa_str {
3516 \msg_set:nnn{stex}{warning/deprecated}{
3517 \\\
3518 \c_backslash_str trefi-is-deprecated! \\\
3519 Please~use~\c_backslash_str STEXsymbol{#2}![#2]~instead!~(in~file~
3520 \stex_path_to_string:N \g_stex_currentfile_seq)
3521 \\\ \\\
3522 }
3523 \msg_warning:nn{stex}{warning/deprecated}
3524 \STEXsymbol { #2 }![ \comp{#2} ]
3525 } {
3526 \msg_set:nnn{stex}{warning/deprecated}{
3527 \\\
3528 \c_backslash_str trefi-is-deprecated! \\\
3529 Please~use~\c_backslash_str STEXsymbol { #1?#2 }[ #2 ]~instead!~(in~file~
3530 \stex_path_to_string:N \g_stex_currentfile_seq)
3531 \\\ \\\
3532 }
3533 \msg_warning:nn{stex}{warning/deprecated}

```

```

3534 \STEXsymbol { #1 }![ \comp{#2} ]
3535 }
3536 }
3537
3538
3539 \NewDocumentCommand \Trefi { 0{} 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}![ \exp_after:wN \stex_capitalize:n #2]~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{\exp_after:wN \stex_capitalize:n #2} ]
3551 } {
3552 \msg_set:nnn{stex}{warning/deprecated}{
3553 \\\
3554 \c_backslash_str Trefi-is-deprecated! \\\
3555 Please~use~\c_backslash_str STEXsymbol { #1 }[ \exp_after:wN \stex_capitalize:n #2 ]~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{\exp_after:wN \stex_capitalize:n #2} ]
3561 }
3562 }
3563
3564 \NewDocumentCommand \trefis { 0{} m } {
3565 \str_set:Nn \l_tmpa_str { #1 }
3566 \str_if_empty:NTF \l_tmpa_str {
3567 \msg_set:nnn{stex}{warning/deprecated}{
3568 \\\
3569 \c_backslash_str trefi-is-deprecated! \\\
3570 Please~use~\c_backslash_str STEXsymbol{#2}![#2s]~instead!~(in-file~
3571 \stex_path_to_string:N \g_stex_currentfile_seq)
3572 \\\ \\\
3573 }
3574 \msg_warning:nn{stex}{warning/deprecated}
3575 \STEXsymbol { #2 }![ \comp{#2s} ]
3576 } {
3577 \msg_set:nnn{stex}{warning/deprecated}{
3578 \\\
3579 \c_backslash_str trefi-is-deprecated! \\\
3580 Please~use~\c_backslash_str STEXsymbol { #1 }[ #2s ]~instead!~(in-file~
3581 \stex_path_to_string:N \g_stex_currentfile_seq)
3582 \\\ \\\
3583 }
3584 \msg_warning:nn{stex}{warning/deprecated}
3585 \STEXsymbol { #1 }![ \comp{#2s} ]
3586 }
3587 }

```

```

3588
3589
3590 \NewDocumentCommand \Trefis { O{} 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 Trefis~is~deprecated! \\
3596       Please~use~\c_backslash_str STExsymbol{#2}![\exp_after:wN \stex_capitalize:n #2s]~inst
3597       \stex_path_to_string:N \g_stex_currentfile_seq)
3598     \\ \\
3599   }
3600   \msg_warning:nn{stex}{warning/deprecated}
3601   \STExsymbol { #2 }![ \comp{\exp_after:wN \stex_capitalize:n #2s} ]
3602 } {
3603   \msg_set:nnn{stex}{warning/deprecated}{
3604     \\
3605     \c_backslash_str Trefis~is~deprecated! \\
3606     Please~use~\c_backslash_str STExsymbol { #1 }[ \exp_after:wN \stex_capitalize:n #2s ]~
3607     \stex_path_to_string:N \g_stex_currentfile_seq)
3608     \\ \\
3609   }
3610   \msg_warning:nn{stex}{warning/deprecated}
3611   \STExsymbol { #1 }![ \comp{\exp_after:wN \stex_capitalize:n #2s} ]
3612 }
3613 }
3614
3615 \NewDocumentCommand \trefii { O{} 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 trefii~is~deprecated! \\
3621       Please~use~\c_backslash_str STExsymbol{#2-#3}![#2~#3]~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 }![ \comp{#2~#3} ]
3627   } {
3628     \msg_set:nnn{stex}{warning/deprecated}{
3629       \\
3630       \c_backslash_str trefii~is~deprecated! \\
3631       Please~use~\c_backslash_str STExsymbol { #1 }[ #2~#3 ]~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} ]
3637   }
3638 }
3639
3640 \NewDocumentCommand \trefiii { O{} m m m } {
3641   \str_set:Nn \l_tmpa_str { #1 }

```

```

3642 \str_if_empty:NTF \l_tmpa_str {
3643   \msg_set:nnn{stex}{warning/deprecated}{
3644     \
3645     \c_backslash_str trefiii~is-deprecated! \
3646     Please~use~\c_backslash_str STExsymbol{#2-#3-#4}![#2~#3~#4]~instead!~(in~file~
3647     \stex_path_to_string:N \g_stex_currentfile_seq)
3648     \
3649   }
3650   \msg_warning:nn{stex}{warning/deprecated}
3651   \STExsymbol { #2-#3-#4 }![ \comp{#2~#3~#4} ]
3652 } {
3653   \msg_set:nnn{stex}{warning/deprecated}{
3654     \
3655     \c_backslash_str trefiii~is-deprecated! \
3656     Please~use~\c_backslash_str STExsymbol { #1 }[ #2~#3~#4 ]~instead!~(in~file~
3657     \stex_path_to_string:N \g_stex_currentfile_seq)
3658     \
3659   }
3660   \msg_warning:nn{stex}{warning/deprecated}
3661   \STExsymbol { #1 }![ \comp{#2~#3~#4} ]
3662 }
3663 }
3664
3665
3666 \NewDocumentCommand \trefiis { O{} m m } {
3667   \str_set:Nn \l_tmpa_str { #1 }
3668   \str_if_empty:NTF \l_tmpa_str {
3669     \msg_set:nnn{stex}{warning/deprecated}{
3670       \
3671       \c_backslash_str trefiis~is-deprecated! \
3672       Please~use~\c_backslash_str STExsymbol{#2-#3}![#2~#3s]~instead!~(in~file~
3673       \stex_path_to_string:N \g_stex_currentfile_seq)
3674       \
3675     }
3676     \msg_warning:nn{stex}{warning/deprecated}
3677     \STExsymbol { #2-#3 }![ \comp{#2~#3s} ]
3678   } {
3679     \msg_set:nnn{stex}{warning/deprecated}{
3680       \
3681       \c_backslash_str trefiis~is-deprecated! \
3682       Please~use~\c_backslash_str STExsymbol { #1 }[ #2~#3s ]~instead!~(in~file~
3683       \stex_path_to_string:N \g_stex_currentfile_seq)
3684       \
3685     }
3686     \msg_warning:nn{stex}{warning/deprecated}
3687     \STExsymbol { #1 }![ \comp{#2~#3s} ]
3688   }
3689 }
3690
3691 \NewDocumentCommand \symvariant { O{} m O{0} m m } {
3692   \msg_set:nnn{stex}{warning/deprecated}{
3693     \
3694     \c_backslash_str symvariant~is-deprecated! \
3695     Please~use~\c_backslash_str notation[#4]{ #2 }~instead!~(in~file~

```

```

3696     \stex_path_to_string:N \g_stex_currentfile_seq)
3697     \\ \\
3698   }
3699   \msg_warning:nn{stex}{warning/deprecated}
3700
3701   \notation[variant=#4]{#2}{#5}
3702 }
3703
3704 \NewDocumentCommand \mixfixi { 0{} m m m } {
3705   \msg_set:nnn{stex}{warning/deprecated}{
3706     \c_backslash_str mixfixi~is~fatally~deprecated!\\
3707     Symbol:~\l__stex_term_highlight_uri_str\\
3708     Current~file:~\stex_path_to_string:N \g_stex_currentfile_seq
3709   }
3710   \msg_error:nn{stex}{warning/deprecated}
3711 }
3712
3713
3714 \NewDocumentCommand \infix {} {
3715   \msg_set:nnn{stex}{warning/deprecated}{
3716     \c_backslash_str infix~is~fatally~deprecated!\\
3717     Symbol:~\l__stex_term_highlight_uri_str\\
3718     Current~file:~\stex_path_to_string:N \g_stex_currentfile_seq
3719   }
3720   \msg_error:nn{stex}{warning/deprecated}
3721 }
3722
3723 \let\iprec\infprec
3724
3725 \NewDocumentCommand \inlineex { m } {
3726   \msg_set:nnn{stex}{warning/deprecated}{
3727     \c_backslash_str inlineex~is~deprecated!\\
3728     No~replacement~exists~yet.\\
3729     Current~file:~\stex_path_to_string:N \g_stex_currentfile_seq
3730   }
3731   \msg_warning:nn{stex}{warning/deprecated}
3732   #1
3733 }
3734
3735
3736 \NewDocumentCommand \term { m } {
3737   \msg_set:nnn{stex}{warning/deprecated}{
3738     \c_backslash_str term~is~deprecated!\\
3739     No~replacement~exists~yet.\\
3740     Current~file:~\stex_path_to_string:N \g_stex_currentfile_seq
3741   }
3742   \msg_warning:nn{stex}{warning/deprecated}
3743   #1
3744 }
3745
3746
3747 \NewDocumentCommand \Definame { 0{} m } {
3748   \stex_get_symbol:n { #2 }
3749   \str_set:Nx \l_tmpa_str {

```



```

3750     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3751   }
3752   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3753   \scalate_if:TF {
3754     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3755       \l_tmpa_str
3756     }
3757   } {
3758     \@defemph {
3759       \exp_after:wN \stex_capitalize:n \l_tmpa_str
3760     } { \l_stex_get_symbol_uri_str }
3761   }
3762 }
3763
3764 \NewDocumentCommand \Definiendum { O{} m m } {
3765   \stex_get_symbol:n { #2 }
3766   \str_set:Nx \l_tmpa_str {
3767     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3768   }
3769   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3770   \scalate_if:TF {
3771     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3772       \l_tmpa_str
3773     }
3774   } {
3775     \@defemph {
3776       \exp_after:wN \stex_capitalize:n \l_tmpa_str
3777     } { \l_stex_get_symbol_uri_str }
3778   }
3779 }
3780
3781 \NewDocumentCommand \Symname { O{} m }{
3782   \stex_symname_args:n { #1 }
3783   \stex_get_symbol:n { #2 }
3784   \str_set:Nx \l_tmpa_str {
3785     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3786   }
3787   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3788   \exp_args:NNx \use:nn
3789   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }![
3790     \exp_after:wN \stex_capitalize:n \l_tmpa_str
3791     \l_stex_symname_post_str
3792   ] }
3793 }
3794
3795
3796 \seq_gput_right:Nx \g_stex_smsmode_allowedenvs_seq { \tl_to_str:n { mhmodnl } }
3797 \seq_gput_right:Nx \g_stex_smsmode_allowedenvs_seq { \tl_to_str:n { modsig } }
3798 \tl_gput_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\gimport\syml\syml\syml\symliv}
3799
3800 % omtex:
3801 \cs_new_protected:Npn \lec #1 {
3802   \strut\hfil\strut\hfill(#1)
3803 }

```

```

3804 \cs_new_protected:Npn \nlex #1 {
3805   \textcolor{green}{\s1 #1}}
3806 }
3807
3808 \newcommand\hateq{\ensuremath{\widehat{=}}\xspace}
3809 \newcommand\hatequiv{\ensuremath{\widehat{\equiv}}\xspace}
3810 \@ifundefined{ergo}%
3811 {\newcommand\ergo{\ensuremath{\leadsto}\xspace}}%
3812 {\renewcommand\ergo{\ensuremath{\leadsto}\xspace}}%
3813 \newcommand{\reflect@squig}[2]{\reflectbox{$\m@th#1\rightsquigarrow$}}%
3814 \newcommand\ogre{\ensuremath{\mathrel{\mathpalette\reflect@squig\relax}}\xspace}%
3815 \newcommand\notergo{\ensuremath{\not\leadsto}}
3816 \newcommand\notogre{\ensuremath{\not\mathrel{\mathpalette\reflect@squig\relax}}\xspace}%
3817
3818 % mathhub convenience macros
3819
3820 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
3821 \newcommand\mhgraphics[2][{}]{%
3822   \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
3823   \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}
3824 \newcommand\cmhgraphics[2][{}]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
3825
3826 \newcommand\mhtikzinput[2][{}]{%
3827   \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
3828   \stex_in_repository:nn\Gin@mhrepos{
3829     \tikzinput[#1]{\mhp\Gin@mhrepos{#2}}
3830   }
3831 }
3832 \newcommand\cmhtikzinput[2][{}]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
3833
3834 \newcommand\lstinputmhlisting[2][{}]{%
3835   \def\lst@mhrepos{}\setkeys{lst}{#1}%
3836   \lstinputlisting[#1]{\mhp\lst@mhrepos{#2}}
3837 \newcommand\clstinputmhlisting[2][{}]{\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
3838
3839 %%%%%%%%%%% CHEATING --> implement us %%%%%%%%%%%
3840 \newcommand\assdef[2][{}]{#2}
3841 \newcommand\impdec[1]{#1}
3842 \newenvironment{inlineAssertion}{}{}
3843 \newenvironment{sproof}[2][{}]{#2}
3844 \newcommand\spfsketch[2][{}]{#2}
3845 \newenvironment{spfstep}[1][{}]{#1}
3846 \newenvironment{spfcases}[2][{}]{#2}
3847 \newenvironment{spfcase}[2][{}]{#2}
3848 \newcommand\gstructure[3][{}]{\importmodule[#1]{#3}}
3849 \newcommand\fassign[3]{}
3850 \newcommand\vassign[2]{}
3851 \newcommand\tassign[2]{}
3852 \newenvironment{gviewsig}[4][{}]{#4}
3853 \newenvironment{gviewnl}[5][{}]{#5}
3854 \newenvironment{mhview}[5][{}]{#5}
3855 \newenvironment{axiom}[1][{}]{#1}
3856 \newenvironment{example}[1][{}]{#1}
3857 \newenvironment{sblockquote}[1][{}]{#1}

```

```

3858 \newcommand\hypernym[3][1]{}
3859 \newcommand\withcite[2]{}
3860 \newcommand\sref[2][]{}
3861 \</compat>

```