The sTeX3 Package [*]

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

2022-01-17

Abstract

TODO

[*]Version 3.0 (last revised 2022-01-17)

Contents

Part I
Manual

Chapter 1

Stuff

1.1 Modules

`\sTeX`
`\stex`

Both print this sTEX logo.

1.1.1 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 changed[1]. EdN:1

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...
```

Multiplyingagain 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 $x \in A$

[1]EdNote: TODO

.

When using `*[`n`]`, after reading the provided (nth) 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 themself.

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]`

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.

EdN:2
EdN:3
[2] [3]

[2]EdNote: what about e.g. \int _x\int _y\int _z f dx dy dz?

[3]EdNote: "decompose" a-type arguments into fixed-arity operators?

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.

sTeX 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 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 As operator precedence should be smaller than Bs 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 $a\cdot(b+c)$

.

1.1.2 Archives and Imports

Namespaces

Ideally, sTeX 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, 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 sTeX can resolve them accordingly. Largely, users need not concern themselves with namespaces at all, but for completenesses 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 suffix[1].

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.

Paths in Import-Statements

Conversely, here is how namespaces/URIs and file paths are computed in import statements, examplary `\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.

- 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, is 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.

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

Part II
Documentation

Chapter 2

sTeX-Basics

Both the sTeX package and class offer the following package options:

`debug` (⟨*log-prefix*⟩∗) Logs debugging information with the given prefixes to the terminal, or all if `all` is given.

`showmods` (⟨*boolean*⟩) Shows explicit module information at the document margins.

`lang` (⟨*language*⟩∗) Languages to load with the babel package.

`mathhub` (⟨*directory*⟩) MathHub folder to search for repositories.

`sms` (⟨*boolean*⟩) use *persisted* mode (see ???).

`image` (⟨*boolean*⟩) passed on to tikzinput.

2.1 Macros and Environments

`\sTeX`
`\stex`

Both print this sTeX logo.

`\stex_debug:nn`

`\stex_debug:nn {⟨log-prefix⟩} {⟨message⟩}`

Logs ⟨*message*⟩, if the package option `debug` contains ⟨*log-prefix*⟩.

`\stex_add_to_sms:n`

Adds the provided code to the `.sms`-file of the document.

`\if@latexml`
`\latexml_if_p:`
`\latexml_if:T`
`\latexml_if:F`
`\latexml_if:TF`

LaTeX2e and LaTeX3 conditionals for LaTeXML.

We have four macros for annotating generated HTML (via LaTeXML or RusTeX) with attributes:

`\stex_annotate:nnn`
`\stex_annotate_invisible:nnn`
`\stex_annotate_invisible:n`

`\stex_annotate:nnn {⟨property⟩} {⟨resource⟩} {⟨content⟩}`

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.

`stex_annotate_env`

```
\begin{stex_annotate_env}{⟨property⟩}{⟨resource⟩}
⟨content⟩
\end{stex_annotate_env}
```

behaves like `\stex_annotate:nnn {⟨property⟩} {⟨resource⟩} {⟨content⟩}`.

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

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`.

`\stex_deactivate_macro:Nn`
`\stex_reactivate_macro:N`

`\stex_deactivate_macro:Nn⟨cs⟩{⟨environments⟩}`

Makes the macro ⟨*cs*⟩ throw an error, indicating that it is only allowed in the context of ⟨*environments*⟩.

`\stex_reactivate_macro:N⟨cs⟩` reactivates it again, i.e. this happens ideally in the ⟨*begin*⟩-code of the associated environments.

`\MSC`

`\MSC{⟨msc⟩}`

Designates the *math subject classifier* of the current module / file.

Chapter 3

sTeX-MathHub

Code related to managing and using MathHub repositories, files, paths and related hooks and methods.

3.1 Macros and Environments

`\stex_kpsewhich:n`

`\stex_kpsewhich:n` executes kpsewhich and stores the return in `\l_stex_kpsewhich_return_str`. This does not require shell escaping.

3.1.1 Files, Paths, URIs

`\stex_path_from_string:Nn`
`\stex_path_from_string:(NV|cn|cV)`

`\stex_path_from_string:Nn` ⟨*path-variable*⟩ `{`⟨*string*⟩`}`

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

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

The inverse; turns a path into a string and stores it in the second argument variable, or leaves it in the input stream.

`\stex_path_canonicalize:N`

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

`\stex_path_if_absolute_p:N` ⋆
`\stex_path_if_absolute:N`*TF* ⋆

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`
`\c_stex_mainfile_str`

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/../..} & \\\hline
\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/../..		

.

3.1.2 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.

`\stex_in_repository:nn`

`\stex_in_repository:nn{⟨repository-name⟩}{⟨code⟩}`

Change the current repository to `{`⟨*repository-name*⟩`}` (or not, if `{`⟨*repository-name*⟩`}` is empty), and passes its ID on to `{`⟨*code*⟩`}` as `#1`. Switches back to the previous repository after executing `{`⟨*code*⟩`}`.

`\mhpath` ⋆

`\mhpath{⟨archive-ID⟩}{⟨filename⟩}`

Expands to the full path of file ⟨*filename*⟩ in repository ⟨*archive-ID*⟩. Does not check whether the file or the repository exist.

`\inputref`
`\inputref:nn`

`\inputref[⟨archive-ID⟩]{⟨filename⟩}`

`\input`s the file ⟨*filename*⟩ in repository ⟨*archive-ID*⟩.

`\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:
```

.

Chapter 4

sTeX-References

Code related to links and cross-references

4.1 Macros and Environments

Chapter 5

sTeX-Modules

Code related to Modules

5.1 Macros and Environments

`\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`,

`\l_stex_all_modules_seq` Stores full URIs for all modules currently in scope.

`\g_stex_module_files_prop`
`\g_stex_modules_in_file_seq`

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

`\stex_if_in_module_p:` ⋆
`\stex_if_in_module:`*TF* ⋆

Conditional for whether we are currently in a module

`\stex_if_module_exists_p:n` ⋆
`\stex_if_module_exists:n`*TF* ⋆

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.

`\stex_modules_compute_namespace:nN`

`\stex_modules_compute_namespace:nN` {⟨*namespace*⟩} {⟨*path*⟩}

Computes the namespace for file ⟨*path*⟩ in repository with namespace ⟨*namespace*⟩ 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`.

`\stex_modules_current_namespace:`

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/doc/stextest
Faking a repository:
Namespace 2:
http://mathhub.info/tests/Foo/Bar/test/stextest
```

.

5.1.1 The `module`-environment

`module` `\begin{module}[⟨options⟩]{⟨name⟩}`

Opens a new module with name ⟨*name*⟩.
TODO document options.

`\stex_module_setup:nn` `\stex_module_setup:nn{⟨params⟩}{⟨name⟩}`

Sets up a new module with name ⟨*name*⟩ and optional parameters ⟨*params*⟩. In particular, sets `\l_stex_current_module_prop` appropriately.

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

`@module` `\begin{@module}[⟨options⟩]{⟨name⟩}`

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:nn{modules}{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:nn{modules}{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 5.1.1[Bar] (FooBar)
Module path: http://mathhub.info/tests/Foo/Bar/Foo?Bar
Language:
Signature:
Metatheory:

.

`\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 `!`⟨*macro*⟩ or `?{`⟨*symbolname*⟩`}`. In the first case, it stores the full URI in ⟨*macro*⟩; in the second case, it invokes the symbol ⟨*symbolname*⟩ in the selected module.

Test 6

```
\begin{module}{STEXModuleTest1}
\symdecl{foo}
\end{module}
\begin{module}{STEXModuleTest2}
\importmodule{STEXModuleTest1}
\symdecl{foo}
\end{module}
\begin{module}{STEXModuleTest3}
\importmodule{STEXModuleTest2}
\symdecl{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 5.1.2[STEXModuleTest1]

Module 5.1.3[STEXModuleTest2]
modulesImporting module: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?STEXModuleTest1

Module 5.1.4[STEXModuleTest3]
modulesImporting module: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?STEXModuleTest2
file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?STEXModuleTest1
file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?STEXModuleTest2
file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?STEXModuleTest3
foo1
foo2
foo3

.

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

Chapter 6

sTeX-Module Inheritance

Code related to Module Inheritance, in particular *sms mode*.

6.1 Macros and Environments

6.1.1 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
```

.

6.1.2 Imports and Inheritance

`\importmodule`

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

Imports a module by reading it from a file and "activating" it. sTEX 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 6.1.1[Foo]
Meaning: »`macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo?foo}`«

Meaning: »`macro:->\protect \bar `«

Module 6.1.2[Importtest]
modulesImporting module: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo Meaning: »`macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo?foo}`«

Module 6.1.3[Importtest2]
modulesImporting module: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Importtest
Meaning: »`macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/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 6.1.4[UseTest1]

Module 6.1.5[UseTest2]
file://home/jazzpirate/work/Software/ext/sTeX/doc/stextestUseTest1 Meaning: »`undefined`«

Module 6.1.6[UseTest3]
modulesImporting module: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest2 Meaning: »`undefined`«
Meaning: »`macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest2?bar}`«

All modules: http://mathhub.info/sTeX?Metatheory, file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest3,
file://home/jazzpirate/work/Software/ext/sTeX/doc/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/doc/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 6.1.7[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}`«

.

`\stex_import_module_uri:nn` `\stex_import_module_uri:nn {⟨archive-ID⟩} {⟨module-path⟩}`

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

If ⟨*archive-ID*⟩ is empty, it is automatically set to the ID of the current archive (if one exists).

1. If ⟨*archive-ID*⟩ is empty:
   (a) If ⟨*path*⟩ is empty, then ⟨*name*⟩ must have been declared earlier in the same file and retrievable from `\g_stex_modules_in_file_seq`, or a file with name ⟨*name*⟩`.`⟨*lang*⟩`.tex` must exist in the same folder, containing a module ⟨*name*⟩.

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

   (b) If ⟨*path*⟩ is not empty, it must point to the relative path of the containing file as well as the namespace.
2. Otherwise:
   (a) If ⟨*path*⟩ is empty, then ⟨*name*⟩ must have been declared earlier in the same file and retrievable from `\g_stex_modules_in_file_seq`, or a file with name ⟨*name*⟩`.`⟨*lang*⟩`.tex` must exist in the top `source` folder of the archive, containing a module ⟨*name*⟩.

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

   (b) If ⟨*path*⟩ is not empty, it must point to the path of the containing file as well as the namespace, relative to the namespace of the archive.

       If a module by that namespace exists, it is returned. Otherwise, we call `\stex_require_module:nn` on the `source` directory of the archive to find the file.

`\stex_import_require_module:nnnn` `{⟨ns⟩} {⟨archive-ID⟩} {⟨path⟩} {⟨name⟩}`

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

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

Chapter 7

sTeX-Symbols

Code related to symbol declarations and notations

7.1 Macros and Environments

`\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 sTeX, 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 sTeX 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\Nat}{x\geq0}`.

`\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 `i`s, `a`s and `b`s),
- `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 7.1.1[SymdeclTest]
      Meaning: »macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?SymdeclTest?foo}«
Result: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?SymdeclTest?foo
Meaning: »macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/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...

`\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}_{\com
\end{module}
```

Module 7.1.2[NotationTest]
modulesImporting module: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo

.

`\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 7.1.3[SymdefTest]
$a+b+c$

.

Chapter 8

sTeX-Terms

Code related to symbolic expressions, typesetting notations, notation components, etc.

8.1 Macros and Environments

`\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}`.

`\_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.

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

Annotates ⟨*body*⟩ as the ⟨*int*⟩th argument of the current `OMA` or `OMBIND`, with (downwards) argument precedence ⟨*prec*⟩.

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

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

`\infprec`
`\neginfprec`

Maximal and minimal notation precedences.

`\dobrackets`

`\dobrackets {⟨body⟩}`

Puts ⟨*body*⟩ in parentheses; scaled if in display mode unscaled otherwise. Uses the current sTeX brackets (by default `(` and `)`), which can be changed temporarily using `\withbrackets`.

`\withbrackets`

`\withbrackets ⟨left⟩ ⟨right⟩ {⟨body⟩}`

Temporarily (i.e. within ⟨*body*⟩) sets the brackets used by sTeX for automated bracketing (by default `(` and `)`) to ⟨*left*⟩ and ⟨*right*⟩.

Note that ⟨*left*⟩ and ⟨*right*⟩ need to be allowed after `\left` and `\right` in display-mode.

Test 14

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

\end{module}
```

Module 8.1.1[MathTest1]
modulesImporting module: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo $\langle {a^b}_c \rangle$ and $\langle {a^b}_c \rangle$.

.

Test 15

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

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

Module 8.1.2[MathTest2]
modulesImporting module: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo $\langle a|[b_{:c_{:d_{:e_{:f}}}}]^g\rangle$ and $\langle a|[b_{:c}]^g\rangle$ and $\langle a|[b]^c\rangle$
$a+(b\cdot c)$ and $a\cdot\frac{a}{b}+\frac{a}{c}$

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

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

.

`\stex_term_custom:nn`

`\stex_term_custom:nn{⟨URI⟩}{⟨args⟩}`

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

Test 16

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

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

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

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

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

\bar![bar]

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

\end{module}
```

Module 8.1.3[TextTest]
modulesImporting module: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo
some aand some band also some chere.
some a and some b and also some c here.
`bar`
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`
`\compemph`
`\compemph@uri`
`\defemph`
`\defemph@uri`
`\symrefemph`
`\symrefemph@uri`

`\comp{⟨args⟩}`

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`)

`\STEXinvisible`

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.

`\ellipses`

TODO

Chapter 9

sTeX-Structural Features

Code related to structural features

9.1 Macros and Environments

9.1.1 Structures

`mathstructure` TODO

Test 17

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

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

\present\magma

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

Test: $\mM{op}ab$

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

Module 9.1.1[StructureTest1]

$a \circ b : M$

file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?StructureTest1/Magma-feature?universe,file://home/jazzpirate/work/S
feature?op

```
»macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?StructureTest1?Magma}«
```

Test: $a+b$

Test2: $\langle U , + \rangle$

.

Chapter 10

sTeX-Statements

Code related to statements, e.g. definitions, theorems

10.1 Macros and Environments

symboldoc `\begin{`⟨*symboldoc*⟩`}{`⟨*symbols*⟩`}` ⟨*text*⟩ `\end{`⟨*symboldoc*⟩`}`

Declares ⟨*text*⟩ to be a (natural language, encyclopaedic) description of `{`⟨*symbols*⟩`}` (a comma separated list of symbol identifiers).

Chapter 11

sTeX-Proofs: Structural Markup for Proofs

The `sproof` package is part of the sTeX collection, a version of TeX/LaTeX that allows to markup TeX/LaTeX documents semantically without leaving the document format, essentially turning TeX/LaTeX into a document format for mathematical knowledge management (MKM).

This package supplies macros and environment that allow to annotate the structure of mathematical proofs in sTeX files. This structure can be used by MKM systems for added-value services, either directly from the sTeX sources, or after translation.

Contents

11.1 Introduction

The `sproof` (semantic proofs) package supplies macros and environment that allow to annotate the structure of mathematical proofs in sTeX files. This structure can be used by MKM systems for added-value services, either directly from the sTeX sources, or after translation. Even though it is part of the sTeX collection, it can be used independently, like it's sister package `statements`.

sTeX is a version of TeX/LaTeX that allows to markup TeX/LaTeX documents semantically without leaving the document format, essentially turning TeX/LaTeX into a document format for mathematical knowledge management (MKM).

```
\begin{sproof}[id=simple-proof,for=sum-over-odds]
   {We prove that $\sum_{i=1}^n{2i-1}=n^{2}$ by induction over $n$}
  \begin{spfcases}{For the induction we have to consider the following cases:}
   \begin{spfcase}{$n=1$}
    \begin{spfstep}[display=flow] then we compute $1=1^2$\end{spfstep}
   \end{spfcase}
   \begin{spfcase}{$n=2$}
      \begin{sproofcomment}[display=flow]
        This case is not really necessary, but we do it for the
        fun of it (and to get more intuition).
      \end{sproofcomment}
      \begin{spfstep}[display=flow] We compute $1+3=2^{2}=4$.\end{spfstep}
   \end{spfcase}
   \begin{spfcase}{$n>1$}
      \begin{spfstep}[type=assumption,id=ind-hyp]
        Now, we assume that the assertion is true for a certain $k\geq 1$,
        i.e. $\sum_{i=1}^k{(2i-1)}=k^{2}$.
      \end{spfstep}
      \begin{sproofcomment}
        We have to show that we can derive the assertion for $n=k+1$ from
        this assumption, i.e. $\sum_{i=1}^{k+1}{(2i-1)}=(k+1)^{2}$.
      \end{sproofcomment}
      \begin{spfstep}
        We obtain $\sum_{i=1}^{k+1}{2i-1}=\sum_{i=1}^k{2i-1}+2(k+1)-1$
        \begin{justification}[method=arith:split-sum]
          by splitting the sum.
        \end{justification}
      \end{spfstep}
      \begin{spfstep}
        Thus we have $\sum_{i=1}^{k+1}{(2i-1)}=k^2+2k+1$
        \begin{justification}[method=fertilize]
          by inductive hypothesis.
        \end{justification}
      \end{spfstep}
      \begin{spfstep}[type=conclusion]
        We can \begin{justification}[method=simplify]simplify\end{justification}
        the right-hand side to ${k+1}^2$, which proves the assertion.
      \end{spfstep}
   \end{spfcase}
    \begin{spfstep}[type=conclusion]
      We have considered all the cases, so we have proven the assertion.
    \end{spfstep}
  \end{spfcases}
\end{sproof}
```

Example 1: A very explicit proof, marked up semantically

EdN:4 We will go over the general intuition by way of our running example (see Figure 1 for the source and Figure 2 for the formatted result).[4]

[4]EdNote: talk a bit more about proofs and their structure,... maybe copy from OMDoc spec.

11.2 The User Interface

11.2.1 Package Options

showmeta The sproof package takes a single option: showmeta. If this is set, then the metadata keys are shown (see [**Kohlhase:metakeys**] for details and customization options).

11.2.2 Proofs and Proof steps

sproof The proof environment is the main container for proofs. It takes an optional KeyVal argument that allows to specify the id (identifier) and for (for which assertion is this a proof) keys. The regular argument of the proof environment contains an introductory comment, that may be used to announce the proof style. The proof environment contains a sequence of \step, proofcomment, and pfcases environments that are used to markup the proof steps. The proof environment has a variant Proof, which does not use the proof end marker. This is convenient, if a proof ends in a case distinction, which brings sProof it's own proof end marker with it. The Proof environment is a variant of proof that does not mark the end of a proof with a little box; presumably, since one of the subproofs already has one and then a box supplied by the outer proof would generate an otherwise \spfidea empty line. The \spfidea macro allows to give a one-paragraph description of the proof idea.

spfsketch For one-line proof sketches, we use the \spfsketch macro, which takes the KeyVal argument as sproof and another one: a natural language text that sketches the proof.

spfstep Regular proof steps are marked up with the step environment, which takes an optional KeyVal argument for annotations. A proof step usually contains a local assertion (the text of the step) together with some kind of evidence that this can be derived from already established assertions.

Note that both \premise and \justarg can be used with an empty second argument to mark up premises and arguments that are not explicitly mentioned in the text.

11.2.3 Justifications

justification This evidence is marked up with the justification environment in the sproof package. This environment totally invisible to the formatted result; it wraps the text in the proof step that corresponds to the evidence. The environment takes an optional KeyVal argument, which can have the method key, whose value is the name of a proof method (this will only need to mean something to the application that consumes the semantic annotations). Furthermore, the justification can contain "premises" (specifications to assertions that were used justify the step) and "arguments" (other information taken into account by the proof method).

\premise The \premise macro allows to mark up part of the text as reference to an assertion that is used in the argumentation. In the example in Figure 1 we have used the \premise macro to identify the inductive hypothesis.

\justarg The \justarg macro is very similar to \premise with the difference that it is used to mark up arguments to the proof method. Therefore the content of the first argument is interpreted as a mathematical object rather than as an identifier as in the case of \premise. In our example, we specified that the simplification should take place on the right hand side of the equation. Other examples include proof methods that instantiate. Here we would indicate the substituted object in a \justarg macro.

Proof: We prove that $\sum_{i=1}^{n} 2i - 1 = n^2$ by induction over n

P.1 For the induction we have to consider the following cases:

P.1.1 $n = 1$: then we compute $1 = 1^2$ □

P.1.1 $n = 2$: This case is not really necessary, but we do it for the fun of it (and to get more intuition). We compute $1 + 3 = 2^2 = 4$ □

P.1.1 $n > 1$:

P.1.1.1 Now, we assume that the assertion is true for a certain $k \geq 1$, i.e. $\sum_{i=1}^{k} (2i - 1) = k^2$.

P.1.1.1 We have to show that we can derive the assertion for $n = k + 1$ from this assumption, i.e. $\sum_{i=1}^{k+1} (2i - 1) = (k + 1)^2$.

P.1.1.1 We obtain $\sum_{i=1}^{k+1} (2i - 1) = \sum_{i=1}^{k} (2i - 1) + 2(k + 1) - 1$ by splitting the sum

P.1.1.1 Thus we have $\sum_{i=1}^{k+1} (2i - 1) = k^2 + 2k + 1$ by inductive hypothesis.

P.1.1.1 We can simplify the right-hand side to $(k+1)^2$, which proves the assertion. □

P.1.1 We have considered all the cases, so we have proven the assertion. □

Example 2: The formatted result of the proof in Figure 1

11.2.4 Proof Structure

`subproof` `method` The `pfcases` environment is used to mark up a subproof. This environment takes an optional `KeyVal` argument for semantic annotations and a second argument that allows to specify an introductory comment (just like in the `proof` environment). The `method` key can be used to give the name of the proof method executed to make this subproof.

`spfcases` The `pfcases` environment is used to mark up a proof by cases. Technically it is a variant of the `subproof` where the `method` is `by-cases`. Its contents are `spfcase` environments that mark up the cases one by one.

`spfcase` `\spfcasesketch` The content of a `pfcases` environment are a sequence of case proofs marked up in the `pfcase` environment, which takes an optional `KeyVal` argument for semantic annotations. The second argument is used to specify the the description of the case under consideration. The content of a `pfcase` environment is the same as that of a `proof`, i.e. `steps`, `proofcomments`, and `pfcases` environments. `\spfcasesketch` is a variant of the `spfcase` environment that takes the same arguments, but instead of the `spfsteps` in the body uses a third argument for a proof sketch.

`sproofcomment` The `proofcomment` environment is much like a `step`, only that it does not have an object-level assertion of its own. Rather than asserting some fact that is relevant for the proof, it is used to explain where the proof is going, what we are attempting to to, or what we have achieved so far. As such, it cannot be the target of a `\premise`.

11.2.5 Proof End Markers

Traditionally, the end of a mathematical proof is marked with a little box at the end of the last line of the proof (if there is space and on the end of the next line if there isn't), like so:

`\sproofend` `\sProofEndSymbol`

The `sproof` package provides the `\sproofend` macro for this. If a different symbol for the proof end is to be used (e.g. *q.e.d*), then this can be obtained by specifying it using the `\sProofEndSymbol` configuration macro (e.g. by specifying `\sProofEndSymbol{q.e.d}`).

Some of the proof structuring macros above will insert proof end symbols for sub-proofs, in most cases, this is desirable to make the proof structure explicit, but sometimes this wastes space (especially, if a proof ends in a case analysis which will supply its own proof end marker). To suppress it locally, just set `proofend={}` in them or use use `\sProofEndSymbol{}`.

11.2.6 Configuration of the Presentation

EdN:5

Finally, we provide configuration hooks in Figure 1 for the keywords in proofs. These are mainly intended for package authors building on `statements`, e.g. for multi-language support.[5]. The proof step labels can be customized via the `\pstlabelstyle` macro:

Environment	configuration macro	value
`sproof`	`\spf@proof@kw`	Proof
`sketchproof`	`\spf@sketchproof@kw`	ProofSketch

Figure 1: Configuration Hooks for Semantic Proof Markup

`\pstlabelstyle`

`\pstlabelstyle{`⟨*style*⟩`}` sets the style; see Figure 2 for an overview of styles. Package writers can add additional styles by adding a macro `\pst@make@label@`⟨*style*⟩ that takes two arguments: a comma-separated list of ordinals that make up the prefix and the current ordinal. Note that comma-separated lists can be conveniently iterated over by the LaTeX `\@for...:=...\do{...}` macro; see Figure 2 for examples.

style	example	configuration macro
`long`	0.8.1.5	`\def\pst@make@label@long#1#2{\@for\@I:=#1\do{\@I.}#2}`
`angles`	⟩⟩⟩5	`\def\pst@make@label@angles#1#2 {\ensuremath{\@for\@I:=#1\do{\rangle}}#2}`
`short`	5	`\def\pst@make@label@short#1#2{#2}`
`empty`		`\def\pst@make@label@empty#1#2{}`

Figure 2: Configuration Proof Step Label Styles

11.3 Limitations

In this section we document known limitations. If you want to help alleviate them, please feel free to contact the package author. Some of them are currently discussed in the sTeX issue tracker at [sTeX].

[5]EdNote: we might want to develop an extension `sproof-babel` in the future.

1. The numbering scheme of proofs cannot be changed. It is more geared for teaching proof structures (the author's main use case) and not for writing papers. reported by Tobias Pfeiffer (fixed)

2. currently proof steps are formatted by the LaTeX `description` environment. We would like to configure this, e.g. to use the `inparaenum` environment for more condensed proofs. I am just not sure what the best user interface would be I can imagine redefining an internal environment `spf@proofstep@list` or adding a key `prooflistenv` to the `proof` environment that allows to specify the environment directly. Maybe we should do both.

Chapter 12

sTeX-Metatheory

The default meta theory for an sTeX 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 (nth-order) logic, or a Π in dependent type theories.

12.1 Symbols

Part III
Extensions

Chapter 13

Tikzinput

13.1 Macros and Environments

LocalWords: bibfolder jobname.dtx tikzinput.dtx usetikzlibrary Gin@ewidth Gin@eheight
LocalWords: resizebox ctikzinput mhtikzinput Gin@mhrepos mhpath

Chapter 14

document-structure.sty: Semantic Markup for Open Mathematical Documents in LaTeX

The `omdoc` package is part of the sTeX collection, a version of TeX/LaTeX that allows to markup TeX/LaTeX documents semantically without leaving the document format, essentially turning TeX/LaTeX into a document format for mathematical knowledge management (MKM).

This package supplies an infrastructure for writing OMDoc documents in LaTeX. This includes a simple structure sharing mechanism for sTeX that allows to to move from a copy-and-paste document development model to a copy-and-reference model, which conserves space and simplifies document management. The augmented structure can be used by MKM systems for added-value services, either directly from the sTeX sources, or after translation.

14.1 Introduction

sTeX is a version of TeX/LaTeX that allows to markup TeX/LaTeX documents semantically without leaving the document format, essentially turning TeX/LaTeX into a document format for mathematical knowledge management (MKM). The package supports direct translation to the OMDoc format [Koh06]

The `omdoc` package supplies macros and environments that allow to label document fragments and to reference them later in the same document or in other documents. In essence, this enhances the document-as-trees model to documents-as-directed-acyclic-graphs (DAG) model. This structure can be used by MKM systems for added-value services, either directly from the sTeX sources, or after translation. Currently, trans-document referencing provided by this package can only be used in the sTeX collection.

DAG models of documents allow to replace the "Copy and Paste" in the source document with a label-and-reference model where document are shared in the document

EdN:6 source and the formatter does the copying during document formatting/presentation.[6]

14.2 The User Interface

The `omdoc` package generates two files: `omdoc.cls`, and `omdoc.sty`. The OMDoc class is a minimally changed variant of the standard `article` class that includes the functionality provided by `omdoc.sty`. The rest of the documentation pertains to the functionality introduced by `omdoc.sty`.

14.2.1 Package and Class Options

The `omdoc` class accept the following options:

`class=`⟨*name*⟩	load ⟨*name*⟩`.cls` instead of `article.cls`
`topsect=`⟨*sect*⟩	The top-level sectioning level; the default for ⟨*sect*⟩ is `section`
`showignores`	show the the contents of the `ignore` environment after all
`showmeta`	show the metadata; see `metakeys.sty`
`showmods`	show modules; see `modules.sty`
`extrefs`	allow external references; see `sref.sty`
`defindex`	index definienda; see `statements.sty`
`minimal`	for testing; do not load any sTeX packages

The `omdoc` package accepts the same except the first two.

14.2.2 Document Structure

`document` `\documentkeys` `id` The top-level `document` environment can be given key/value information by the `\documentkeys` macro in the preamble[2]. This can be used to give metadata about the document. For the moment only the `id` key is used to give an identifier to the `omdoc` element resulting from the LaTeXML transformation.

`omgroup` The structure of the document is given by the `omgroup` environment just like in OMDoc. In the LaTeX route, the `omgroup` environment is flexibly mapped to sectioning commands, inducing the proper sectioning level from the nesting of `omgroup` environments. Correspondingly, the `omgroup` environment takes an optional key/value argument for metadata followed by a regular argument for the (section) title of the omgroup. The optional metadata argument has the keys `id` for an identifier, `creators` and `contributors` for the Dublin Core metadata [DCM03]; see [Koh20a] for details of the format. The `short` allows to give a short title for the generated section. If the title contains semantic macros, they need to be protected by `\protect`, and we need to give the `loadmodules` key it needs no value. For instance we would have

`id` `creators` `contributors` `short` `loadmodules`

```
\begin{module}{foo}
\symdef{bar}{B^a_r}
 ...
\begin{omgroup}[id=sec.barderiv,loadmodules]{Introducing $\protect\bar$ Derivations}
```

sTeX automatically computes the sectioning level, from the nesting of `omgroup` environments. But sometimes, we want to skip levels (e.g. to use a subsection* as an introduction for a chapter). Therefore the `omdoc` package provides a variant `blindomgroup`

`blindomgroup`

[6]EdNote: integrate with latexml's XMRef in the Math mode.
[2]We cannot patch the document environment to accept an optional argument, since other packages we load already do; pity.

that does not produce markup, but increments the sectioning level and logically groups document parts that belong together, but where traditional document markup relies on convention rather than explicit markup. The `blindomgroup` environment is useful e.g. for creating frontmatter at the correct level. Example 3 shows a typical setup for the outer document structure of a book with parts and chapters. We use two levels of `blindomgroup`:

- The outer one groups the introductory parts of the book (which we assume to have a sectioning hierarchy topping at the part level). This `blindomgroup` makes sure that the introductory remarks become a "chapter" instead of a "part".
- Th inner one groups the frontmatter[3] and makes the preface of the book a section-level construct. Note that here the `display=flow` on the `omgroup` environment prevents numbering as is traditional for prefaces.

```
\begin{document}
\begin{blindomgroup}
\begin{blindomgroup}
\begin{frontmatter}
\maketitle\newpage
\begin{omgroup}[display=flow]{Preface}
... <<preface>> ...
\end{omgroup}
\clearpage\setcounter{tocdepth}{4}\tableofcontents\clearpage
\end{frontmatter}
\end{blindomgroup}
... <<introductory remarks>> ...
\end{blindomgroup}
\begin{omgroup}{Introduction}
... <<intro>> ...
\end{omgroup}
... <<more chapters>> ...
\bibliographystyle{alpha}\bibliography{kwarc}
\end{document}
```

Example 3: A typical Document Structure of a Book

`\skipomgroup` The `\skipomgroup` "skips an `omgroup`", i.e. it just steps the respective sectioning counter. This macro is useful, when we want to keep two documents in sync structurally, so that section numbers match up: Any section that is left out in one becomes a `\skipomgroup`.

`\currentsectionlevel` `\CurrentSectionLevel` The `\currentsectionlevel` macro supplies the name of the current sectioning level, e.g. "chapter", or "subsection". `\CurrentSectionLevel` is the capitalized variant. They are useful to write something like "In this `\currentsectionlevel`, we will…" in an `omgroup` environment, where we do not know which sectioning level we will end up.

14.2.3 Ignoring Inputs

`ignore` `showignores` The `ignore` environment can be used for hiding text parts from the document structure. The body of the environment is not PDF or DVI output unless the `showignores` option

[3] We shied away from redefining the `frontmatter` to induce a blindomgroup, but this may be the "right" way to go in the future.

is given to the `omdoc` class or `package`. But in the generated OMDoc result, the body is marked up with a `ignore` element. This is useful in two situations. For

editing One may want to hide unfinished or obsolete parts of a document

narrative/content markup In sTeX we mark up narrative-structured documents. In the generated OMDoc documents we want to be able to cache content objects that are not directly visible. For instance in the `statements` package [Koh20d] we use the `\inlinedef` macro to mark up phrase-level definitions, which verbalize more formal definitions. The latter can be hidden by an ignore and referenced by the `verbalizes` key in `\inlinedef`.

`\prematurestop`
`\afterprematurestop`

For prematurely stopping the formatting of a document, sTeX provides the `\prematurestop` macro. It can be used everywhere in a document and ignores all input after that – backing out of the omgroup environment as needed. After that – and before the implicit `\end{document}` it calls the internal `\afterprematurestop`, which can be customized to do additional cleanup or e.g. print the bibliography.

`\prematurestop` is useful when one has a driver file, e.g. for a course taught multiple years and wants to generate course notes up to the current point in the lecture. Instead of commenting out the remaining parts, one can just move the `\prematurestop` macro. This is especially useful, if we need the rest of the file for processing, e.g. to generate a theory graph of the whole course with the already-covered parts marked up as an overview over the progress; see `import_graph.py` from the `lmhtools` utilities [LMH].

14.2.4 Structure Sharing

`\STRlabel`
`\STRcopy`

The `\STRlabel` macro takes two arguments: a label and the content and stores the the content for later use by `\STRcopy[`⟨*URL*⟩`]{`⟨*label*⟩`}`, which expands to the previously stored content. If the `\STRlabel` macro was in a different file, then we can give a URL ⟨*URL*⟩ that lets LaTeXML generate the correct reference.

`\STRsemantics`

The `\STRlabel` macro has a variant `\STRsemantics`, where the label argument is optional, and which takes a third argument, which is ignored in LaTeX. This allows to specify the meaning of the content (whatever that may mean) in cases, where the source document is not formatted for presentation, but is transformed into some content markup format.[7]

EdN:7

14.2.5 Global Variables

Text fragments and modules can be made more re-usable by the use of global variables. For instance, the admin section of a course can be made course-independent (and therefore re-usable) by using variables (actually token registers) `courseAcronym` and `courseTitle` instead of the text itself. The variables can then be set in the sTeX preamble of the course notes file. `\setSGvar{`⟨*vname*⟩`}{`⟨*text*⟩`}` to set the global variable ⟨*vname*⟩ to ⟨*text*⟩ and `\useSGvar{`⟨*vname*⟩`}` to reference it.

`\setSGvar`
`\useSGvar`
`\ifSGvar`

With `\ifSGvar` we can test for the contents of a global variable: the macro call `\ifSGvar{`⟨*vname*⟩`}{`⟨*val*⟩`}{`⟨*ctext*⟩`}` tests the content of the global variable ⟨*vname*⟩, only if (after expansion) it is equal to ⟨*val*⟩, the conditional text ⟨*ctext*⟩ is formatted.

[7] EdNote: document LMID und LMXREf here if we decide to keep them.

14.2.6 Colors

`\blue` `\red` ... `\black`

For convenience, the `omdoc` package defines a couple of color macros for the `color` package: For instance `\blue` abbreviates `\textcolor{blue}`, so that `\blue{`⟨*something*⟩`}` writes ⟨*something*⟩ in blue. The macros `\red` `\green`, `\cyan`, `\magenta`, `\brown`, `\yellow`, `\orange`, `\gray`, and finally `\black` are analogous.

14.3 Limitations

In this section we document known limitations. If you want to help alleviate them, please feel free to contact the package author. Some of them are currently discussed in the sTEX GitHub repository [sTeX].

1. when option `book` which uses `\pagestyle{headings}` is given and semantic macros are given in the `omgroup` titles, then they sometimes are not defined by the time the heading is formatted. Need to look into how the headings are made.

Chapter 15

Slides and Course Notes

We present a document class from which we can generate both course slides and course notes in a transparent way.

15.1 Introduction

The `mikoslides` document class is derived from `beamer.cls` [Tana], it adds a "notes version" for course notes derived from the `omdoc` class [**Kohlhase:smomdl**] that is more suited to printing than the one supplied by `beamer.cls`.

15.2 The User Interface

The `mikoslides` class takes the notion of a slide frame from Till Tantau's excellent `beamer` class and adapts its notion of frames for use in the sTEXand OMDoc. To support semantic course notes, it extends the notion of mixing frames and explanatory text, but rather than treating the frames as images (or integrating their contents into the flowing text), the `mikoslides` package displays the slides as such in the course notes to give students a visual anchor into the slide presentation in the course (and to distinguish the different writing styles in slides and course notes).

In practice we want to generate two documents from the same source: the slides for presentation in the lecture and the course notes as a narrative document for home study. To achieve this, the `mikoslides` class has two modes: *slides mode* and *notes mode* which are determined by the package option.

15.2.1 Package Options

EdN:8 The `mikoslides` class takes a variety of class options:[8]

`slides` `notes`

- The options `slides` and `notes` switch between slides mode and notes mode (see Section 15.2.2).

`sectocframes`

- If the option `sectocframes` is given, then for the `omgroups`, special frames with the `omgroup` title (and number) are generated.

showmeta

- showmeta. If this is set, then the metadata keys are shown (see [Koh20b] for details and customization options).

frameimages fiboxed

- If the option frameimages is set, then slide mode also shows the \frameimage-generated frames (see section 15.2.4). If also the fiboxed option is given, the slides are surrounded by a box.

topsect

- topsect=⟨*sect*⟩ can be used to specify the top-level sectioning level; the default for ⟨*sect*⟩ is section.

15.2.2 Notes and Slides

frame note

Slides are represented with the frame just like in the beamer class, see [Tanb] for details. The mikoslides class adds the note environment for encapsulating the course note fragments.[4]

⚠ Note that it is essential to start and end the notes environment at the start of the line – in particular, there may not be leading blanks – else LaTeX becomes confused and throws error messages that are difficult to decipher.

```
\ifnotes\maketitle\else
\frame[noframenumbering]\maketitle\fi

\begin{note}
  We start this course with ...
\end{note}

\begin{frame}
  \frametitle{The first slide}
  ...
\end{frame}
\begin{note}
  ... and more explanatory text
\end{note}

\begin{frame}
  \frametitle{The second slide}
  ...
\end{frame}
...
```

Example 4: A typical Course Notes File

By interleaving the frame and note environments, we can build course notes as shown in Figure 4.

\ifnotes

Note the use of the \ifnotes conditional, which allows different treatment between notes and slides mode – manually setting \notestrue or \notesfalse is strongly discouraged however.

[8]EdNote: leaving out noproblems for the moment until we decide what to do with it.

[4]MK: it would be very nice, if we did not need this environment, and this should be possible in principle, but not without intensive LaTeX trickery. Hints to the author are welcome.

⚠: We need to give the title frame the `noframenumbering` option so that the frame numbering is kept in sync between the slides and the course notes.

⚠: The `beamer` class recommends not to use the `allowframebreaks` option on frames (even though it is very convenient). This holds even more in the `mikoslides` case: At least in conjunction with `\newpage`, frame numbering behaves funnily (we have tried to fix this, but who knows).

`\inputref*` If we want to transclude a the contents of a file as a note, we can use a new variant `\inputref*` of the `\inputref` macro from [KGA20]: `\inputref*{foo}` is equivalent to `\begin{note}\inputref{foo}\end{note}`.

`nomtext` `nomgroup` `ndefinition` `nexample` `nsproof` `nassertion` There are some environments that tend to occur at the top-level of `note` environments. We make convenience versions of these: e.g. the `nomtext` environment is just an `omtext` inside a `note` environment (but looks nicer in the source, since it avoids one level of source indenting). Similarly, we have the `nomgroup`, `ndefinition`, `nexample`, `nsproof`, and `nassertion` environments.

15.2.3 Header and Footer Lines of the Slides

`\setslidelogo` The default logo provided by the `mikoslides` package is the sTeX logo it can be customized using `\setslidelogo{`⟨*logo name*⟩`}`.

`\setsource` `\setlicensing` The default footer line of the `mikoslides` package mentions copyright and licensing. In the `beamer` class, `\source` stores the author's name as the copyright holder . By default it is *Michael Kohlhase* in the `mikoslides` package since he is the main user and designer of this package. `\setsource{`⟨*name*⟩`}` can change the writer's name. For licensing, we use the Creative Commons Attribuition-ShareAlike license by default to strengthen the public domain. If package `hyperref` is loaded, then we can attach a hyperlink to the license logo. `\setlicensing[`⟨*url*⟩`]{`⟨*logo name*⟩`}` is used for customization, where ⟨*url*⟩ is optional.

15.2.4 Frame Images

`\frameimage` Sometimes, we want to integrate slides as images after all – e.g. because we already have a PowerPoint presentation, to which we want to add sTeXnotes. In this case we can use `\frameimage[`⟨*opt*⟩`]{`⟨*path*⟩`}`, where ⟨*opt*⟩ are the options of `\includegraphics` from the `graphicx` package [CR99] and ⟨*path*⟩ is the file path (extension can be left off like in `\includegraphics`). We have added the `label` key that allows to give a frame label that can be referenced like a regular `beamer` frame.[9] EdN:9

`\mhframeimage` The `\mhframeimage` macro is a variant of `\frameimage` with repository support. Instead of writing

```
\frameimage{\MathHub{fooMH/bar/source/baz/foobar}}
```

we can simply write (assuming that `\MathHub` is defined as above)

```
\mhframeimage[fooMH/bar]{baz/foobar}
```

Note that the `\mhframeimage` form is more semantic, which allows more advanced document management features in MathHub.

If `baz/foobar` is the "current module", i.e. if we are on the MathHub path . . . `MathHub/fooMH/bar`. . . , then stating the repository in the first optional argument is redundant, so we can just use

[9]EdNote: MK: the hyperref link does not seem to work yet. I wonder why but do not have the time to fix it.

```
\mhframeimage{baz/foobar}
```

15.2.5 Colors and Highlighting

`\textwarning` The `\textwarning` macro generates a warning sign: ⚠

15.2.6 Front Matter, Titles, etc.

15.2.7 Excursions

In course notes, we sometimes want to point to an "excursion" – material that is either presupposed or tangential to the course at the moment – e.g. in an appendix. The typical setup is the following:

```
\excursion{founif}{../ex/founif}{We will cover first-order unification in}
...
\begin{appendix}\printexcursions\end{appendix}
```

`\excursion` `\activateexcursion` The `\excursion{`⟨*ref*⟩`}{`⟨*path*⟩`}{`⟨*text*⟩`}` is syntactic sugar for

```
\begin{nomtext}[title=Excursion]
  \activateexcursion{founif}{../ex/founif}
  We will cover first-order unification in \sref{founif}.
\end{nomtext}
```

`\activateexcursion` `\printexcursions` where `\activateexcursion{`⟨*path*⟩`}` augments the `\printexcursions` macro by a call `\inputref{`⟨*path*⟩`}`. In this way, the3 `\printexcursions` macro (usually in the appendix) will collect up all excursions that are specified in the main text.

`\excursionref` Sometimes, we want to reference – in an excursion – part of another. We can use `\excursionref{`⟨*label*⟩`}` for that.

`\excursiongroup` Finally, we usually want to put the excursions into an `omgroup` environment and add an introduction, therefore we provide the a variant of the `\printexcursions` macro: `\excursiongroup[id=`⟨*id*⟩`,intro=`⟨*path*⟩`]` is equivalent to

```
\begin{note}
\begin{omgroup}[id=<id>]{Excursions}
  \inputref{<path>}
  \printexcursions
\end{omgroup}
\end{note}
```

15.2.8 Miscellaneous

15.3 Limitations

In this section we document known limitations. If you want to help alleviate them, please feel free to contact the package author. Some of them are currently discussed in the sTEXGitHub repository [sTeX].

1. when option `book` which uses `\pagestyle{headings}` is given and semantic macros are given in the `omgroup` titles, then they sometimes are not defined by the time the heading is formatted. Need to look into how the headings are made. This is a problem of the underlying `omdoc` package.

Chapter 16

problem.sty: An Infrastructure for formatting Problems

The problem package supplies an infrastructure that allows specify problems and to reuse them efficiently in multiple environments.

16.1 Introduction

The problem package supplies an infrastructure that allows specify problem. Problems are text fragments that come with auxiliary functions: hints, notes, and solutions[5]. Furthermore, we can specify how long the solution to a given problem is estimated to take and how many points will be awarded for a perfect solution.

Finally, the problem package facilitates the management of problems in small files, so that problems can be re-used in multiple environment.

16.2 The User Interface

16.2.1 Package Options

solutions notes hints gnotes pts min

The problem package takes the options solutions (should solutions be output?), notes (should the problem notes be presented?), hints (do we give the hints?), gnotes (do we show grading notes?), pts (do we display the points awarded for solving the problem?), min (do we display the estimated minutes for problem soling). If theses are specified, then the corresponding auxiliary parts of the problems are output, otherwise, they remain invisible.

boxed test

The boxed option specifies that problems should be formatted in framed boxes so that they are more visible in the text. Finally, the test option signifies that we are in a test situation, so this option does not show the solutions (of course), but leaves space for the students to solve them.

mh

The mh option turns on MathHub support; see [**Kohlhase:mss**].

showmeta

Finally, if the showmeta is set, then the metadata keys are shown (see [**Kohlhase:metakeys**] for details and customization options).

[5]for the moment multiple choice problems are not supported, but may well be in a future version

16.2.2 Problems and Solutions

problem id pts min title

The main environment provided by the problem package is (surprise surprise) the problem environment. It is used to mark up problems and exercises. The environment takes an optional KeyVal argument with the keys id as an identifier that can be reference later, pts for the points to be gained from this exercise in homework or quiz situations, min for the estimated minutes needed to solve the problem, and finally title for an informative title of the problem. For an example of a marked up problem see Figure 5 and the resulting markup see Figure 6.

```
\usepackage[solutions,hints,pts,min]{problem}
\begin{document}
  \begin{problem}[id=elefants,pts=10,min=2,title=Fitting Elefants]
    How many Elefants can you fit into a Volkswagen beetle?
\begin{hint}
  Think positively, this is simple!
\end{hint}
\begin{exnote}
  Justify your answer
\end{exnote}
\begin{solution}[for=elefants,height=3cm]
  Four, two in the front seats, and two in the back.
\begin{gnote}
  if they do not give the justification deduct 5 pts
\end{gnote}
\end{solution}
  \end{problem}
\end{document}
```

Example 5: A marked up Problem

solution solutions id for height test

The solution environment can be to specify a solution to a problem. If the solutions option is set or \solutionstrue is set in the text, then the solution will be presented in the output. The solution environment takes an optional KeyVal argument with the keys id for an identifier that can be reference for to specify which problem this is a solution for, and height that allows to specify the amount of space to be left in test situations (i.e. if the test option is set in the \usepackage statement).

Problem0.0 ()
How many Elefants can you fit into a Volkswagen beetle?

Hint: Think positively, this is simple!

Note: Justify your answer

Solution: Four, two in the front seats, and two in the back.

Example 6: The Formatted Problem from Figure 5

hint exnote gnote

The hint and exnote environments can be used in a problem environment to give hints and to make notes that elaborate certain aspects of the problem.

The gnote (grading notes) environment can be used to document situtations that may arise in grading.

Sometimes we would like to locally override the `solutions` option we have given to the package. To turn on solutions we use the `\startsolutions`, to turn them off, `\stopsolutions`. These two can be used at any point in the documents.

`\startsolutions` `\stopsolutions`

Also, sometimes, we want content (e.g. in an exam with master solutions) conditional on whether solutions are shown. This can be done with the `\ifsolutions` conditional.

`\ifsolutions`

16.2.3 Multiple Choice Blocks

`mcb` `\mcc`

Multiple choice blocks can be formatted using the `mcb` environment, in which single choices are marked up with `\mcc[⟨keyvals⟩]{⟨text⟩}` macro, which takes an optional key/value argument ⟨*keyvals*⟩ for choice metadata and a required argument ⟨*text*⟩ for the proposed answer text. The following keys are supported

`T` `F` `Ttext` `Ftext` `feedback`

- `T` for true answers, `F` for false ones,
- `Ttext` the verdict for true answers, `Ftext` for false ones, and
- `feedback` for a short feedback text given to the student.

See Figure **??** for an example

16.2.4 Including Problems

`\includeproblem`

The `\includeproblem` macro can be used to include a problem from another file. It takes an optional KeyVal argument and a second argument which is a path to the file containing the problem (the macro assumes that there is only one problem in the include file). The keys `title`, `min`, and `pts` specify the problem title, the estimated minutes for solving the problem and the points to be gained, and their values (if given) overwrite the ones specified in the `problem` environment in the included file.

`title` `min` `pts`

16.2.5 Reporting Metadata

The sum of the points and estimated minutes (that we specified in the `pts` and `min` keys to the `problem` environment or the `\includeproblem` macro) to the log file and the screen after each run. This is useful in preparing exams, where we want to make sure that the students can indeed solve the problems in an allotted time period.

The `\min` and `\pts` macros allow to specify (i.e. to print to the margin) the distribution of time and reward to parts of a problem, if the `pts` and `pts` package options are set. This allows to give students hints about the estimated time and the points to be awarded.

16.3 Limitations

In this section we document known limitations. If you want to help alleviate them, please feel free to contact the package author. Some of them are currently discussed in the sTEXGitHub repository [sTeX].

1. none reported yet

```
\begin{problem}[title=Functions]
  What is the keyword to introduce a function definition in python?
  \begin{mcb}
    \mcc[T]{def}
    \mcc[F,feedback=that is for C and C++]{function}
    \mcc[F,feedback=that is for Standard ML]{fun}
    \mcc[F,Ftext=Nooooooooo,feedback=that is for Java]{public static void}
  \end{mcb}
\end{problem}
```

Problem0.0 ()
What is the keyword to introduce a function definition in python?

1. def
2. function
3. fun
4. public static void

Problem0.0 ()
What is the keyword to introduce a function definition in python?

1. def
   !
2. function
   that is for C and C++
3. fun
   that is for Standard ML
4. public static void
   that is for Java

Example 7: A Problem with a multiple choice block

Chapter 17

hwexam.sty/cls: An Infrastructure for formatting Assignments and Exams

The hwexam package and class allows individual course assignment sheets and compound assignment documents using problem files marked up with the problem package.

Contents

17.1 Introduction

The `hwexam` package and class supplies an infrastructure that allows to format nice-looking assignment sheets by simply including problems from problem files marked up with the `problem` package [**Kohlhase:problem**]. It is designed to be compatible with `problems.sty`, and inherits some of the functionality.

17.2 The User Interface

17.2.1 Package and Class Options

The `hwexam` package and class take the options `solutions`, `notes`, `hints`, `gnotes`, `pts`, `min`, and `boxed` that are just passed on to the `problems` package (cf. its documentation for a description of the intended behavior).

`showmeta` If the `showmeta` option is set, then the metadata keys are shown (see [**Kohlhase:metakeys**] for details and customization options).

The `hwexam` class additionally accepts the options `report`, `book`, `chapter`, `part`, and `showignores`, of the `omdoc` package [**Kohlhase:smomdl**] on which it is based and passes them on to that. For the `extrefs` option see [**Kohlhase:sref**].

17.2.2 Assignments

`assignment` `number` `title` `type` `given` `due` This package supplies the `assignment` environment that groups problems into assignment sheets. It takes an optional KeyVal argument with the keys `number` (for the assignment number; if none is given, 1 is assumed as the default or — in multi-assignment documents — the ordinal of the `assignment` environment), `title` (for the assignment title; this is referenced in the title of the assignment sheet), `type` (for the assignment type; e.g. “quiz”, or “homework”), `given` (for the date the assignment was given), and `due` (for the date the assignment is due).

17.2.3 Typesetting Exams

`multiple` Furthermore, the `hwexam` package takes the option `multiple` that allows to combine multiple assignment sheets into a compound document (the assignment sheets are treated as section, there is a table of contents, etc.).

`test` Finally, there is the option `test` that modifies the behavior to facilitate formatting tests. Only in `test` mode, the macros `\testspace`, `\testnewpage`, and `\testemptypage` have an effect: they generate space for the students to solve the given problems. Thus they can be left in the LaTeX source.

`\testspace` `\testnewpage` `\testemptypage` `\testspace` takes an argument that expands to a dimension, and leaves vertical space accordingly. `\testnewpage` makes a new page in `test` mode, and `\testemptypage` generates an empty page with the cautionary message that this page was intentionally left empty.

`testheading` `duration` `min` `reqpts` Finally, the `\testheading` takes an optional keyword argument where the keys `duration` specifies a string that specifies the duration of the test, `min` specifies the equivalent in number of minutes, and `reqpts` the points that are required for a perfect grade.

17.2.4 Including Assignments

`\inputassignment` The `\inputassignment` macro can be used to input an assignment from another file. It takes an optional KeyVal argument and a second argument which is a path to the file containing the problem (the macro assumes that there is only one `assignment` environment in the included file). The keys `number`, `title`, `type`, `given`, and `due` are just as for the `assignment` environment and (if given) overwrite the ones specified in the `assignment` environment in the included file.

`number` `title` `type` `given` `due`

17.3 Limitations

In this section we document known limitations. If you want to help alleviate them, please feel free to contact the package author. Some of them are currently discussed in the sTEXGitHub repository [sTeX].

1. none reported yet.

```
\title{320101 General Computer Science (Fall 2010)}
\begin{testheading}[duration=one hour,min=60,reqpts=27]
  Good luck to all students!
\end{testheading}
```

formats to

Name: MatriculationNumber:

320101 General Computer Science (Fall 2010)

2022-01-17

You have 60minutes (sharp) for the test;
Write the solutions to the sheet.
The estimated time for solving this exam is 58 minutes, leaving you 2 minutes for revising your exam.
You can reach 30 points if you solve all problems. You will only need 27 points for a perfect score, i.e. 3 points are bonus points.

You have ample time, so take it slow and avoid rushing to mistakes!

Different problems test different skills and knowledge, so do not get stuck on one problem.

	Tobeusedforgrading,donotwritehere											
prob.	0.0	0.0	0.0	1.1	2.1	2.2	2.3	3.1	3.2	3.3	Sum	grade
total				4	4	6	6	4	4	2	30	
reached												

good luck

Example 8: A generated test heading.

Part IV
Implementation

Chapter 18

sTeX -Basics Implementation

18.1 The sTeXDocument Class

The stex document class is pretty straight-forward: It largely extends the standalone package and loads the stex package, passing all provided options on to the package.

```
⟨*cls⟩

%%%%%%%%%%%%%   basics.dtx   %%%%%%%%%%%%%

\RequirePackage{expl3,l3keys2e}
\ProvidesExplClass{stex}{2021/08/01}{1.9}{bla}
\LoadClass[border=1px,varwidth]{standalone}
\setlength\textwidth{15cm}

\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
\ProcessOptions

\RequirePackage{stex}
⟨/cls⟩
```

18.2 Preliminaries

```
⟨*package⟩

%%%%%%%%%%%%%   basics.dtx   %%%%%%%%%%%%%

\RequirePackage{expl3,l3keys2e,ltxcmds}
\ProvidesExplPackage{stex}{2021/08/01}{1.9}{bla}
\RequirePackage{expl-keystr-compat}
\RequirePackage{morewrites}
```

Package options:

```
\keys_define:nn { stex } {
  debug     .clist_set:N  = \c_stex_debug_clist ,
  showmods  .bool_set:N   = \c_stex_showmods_bool ,
  lang       .clist_set:N  = \c_stex_languages_clist ,
  mathhub    .tl_set_x:N   = \mathhub ,
  sms        .bool_set:N   = \c_stex_persist_mode_bool ,
  image      .bool_set:N   = \c_tikzinput_image_bool,
  unknown    .code:n       = {}
}
\ProcessKeysOptions { stex }
```

\stex
\sTeX

The sTEXlogo:

```
\protected\def\stex{%
  \@ifundefined{texorpdfstring}%
  {\let\texorpdfstring\@firstoftwo}%
  {}%
  \texorpdfstring{\raisebox{-.5ex}S\kern-.5ex\TeX}{sTeX}\xspace%
}
\def\sTeX{\stex}
```

(*End definition for* `\stex` *and* `\sTeX`*. These functions are documented on page 9.*)

18.3 Messages and logging

```
⟨@@=stex_log⟩
```

Warnings and error messages

```
\msg_new:nnn{stex}{error/unknownlanguage}{
  Unknown~language:~#1
}
\msg_new:nnn{stex}{warning/nomathhub}{
  MATHHUB~system~variable~not~found~and~no~
  \detokenize{\mathhub}-value~set!
}
\msg_new:nnn{stex}{error/deactivated-macro}{
  The~\detokenize{#1}~command~is~only~allowed~in~#2!
}
```

\stex_debug:nn

A simple macro issuing package messages with subpath.

```
\cs_new_protected:Nn \stex_debug:nn {
  \clist_if_in:NnTF \c_stex_debug_clist { all } {
    \exp_args:Nnnx\msg_set:nnn{stex}{debug / #1}{
      \\Debug~#1:~#2\\
    }
    \msg_none:nn{stex}{debug / #1}
  }{
    \clist_if_in:NnT \c_stex_debug_clist { #1 } {
      \exp_args:Nnnx\msg_set:nnn{stex}{debug / #1}{
        \\Debug~#1:~#2\\
      }
      \msg_none:nn{stex}{debug / #1}
    }
  }
}
```

(*End definition for* `\stex_debug:nn`*. This function is documented on page 9.*)

Redirecting messages:

```
\clist_if_in:NnTF \c_stex_debug_clist {all} {
    \msg_redirect_module:nnn{ stex }{ none }{ term }
}{
  \clist_map_inline:Nn \c_stex_debug_clist {
    \msg_redirect_name:nnn{ stex }{ debug / ##1 }{ term }
  }
}

\stex_debug:nn{log}{debug~mode~on}
```

18.4 Persistence

```
⟨@@=stex_persist⟩
```

`\c__stex_persist_sms_iow` File variable used for the sms-File

```
\iow_new:N \c__stex_persist_sms_iow
\AddToHook{begindocument}{
  \bool_if:NTF \c_stex_persist_mode_bool {
    \ExplSyntaxOn \input{\jobname.sms} \ExplSyntaxOff
  } {
    \iow_open:Nn \c__stex_persist_sms_iow {\jobname.sms}
  }
}
\AddToHook{enddocument}{
  \bool_if:NF \c_stex_persist_mode_bool {
    \iow_close:N \c__stex_persist_sms_iow
  }
}
```

(*End definition for* `\c__stex_persist_sms_iow`.)

`\stex_add_to_sms:n` Adds the provided code to the `.sms`-file of the document.

```
\cs_new_protected:Nn \stex_add_to_sms:n {
  \bool_if:NF \c_stex_persist_mode_bool {
    \iow_now:Nn \c__stex_persist_sms_iow { #1 }
  }
}
```

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

18.5 HTML Annotations

```
⟨@@=stex_annotate⟩
\RequirePackage{rustex}
```

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

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

`\if@latexml` `\latexml_if_p:` `\latexml_if:TF` Conditionals for LATEXML:

```
\ifcsname if@latexml\endcsname\else
    \expandafter\newif\csname if@latexml\endcsname\@latexmlfalse
\fi
```

```

\prg_new_conditional:Nnn \latexml_if: {p, T, F, TF} {
  \if@latexml
    \prg_return_true:
  \else:
    \prg_return_false:
  \fi:
}
```

(*End definition for* `\if@latexml` *and* `\latexml_if:TF`*. These functions are documented on page 9.*)

`\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.

```
\tl_new:N \l__stex_annotate_arg_tl
\tl_const:Nx \c__stex_annotate_emptyarg_tl {
  \rustex_if:TF {
    \rustex_direct_HTML:n { \c_ampersand_str lrm; }
  }{~}
}
```

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

`\__stex_annotate_checkempty:n`

```
\cs_new_protected:Nn \__stex_annotate_checkempty:n {
  \tl_set:Nn \l__stex_annotate_arg_tl { #1 }
  \tl_if_empty:NT \l__stex_annotate_arg_tl {
    \tl_set_eq:NN \l__stex_annotate_arg_tl \c__stex_annotate_emptyarg_tl
  }
}
```

(*End definition for* `\__stex_annotate_checkempty:n`*.*)

`\l_stex_html_do_output_bool`
`\stex_if_do_html:`

Whether to (locally) produce HTML output

```
\bool_new:N \l_stex_html_do_output_bool
\bool_set_true:N \l_stex_html_do_output_bool
\prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
  \bool_if:nTF \l_stex_html_do_output_bool
    \prg_return_true: \prg_return_false:
}
```

(*End definition for* `\l_stex_html_do_output_bool` *and* `\stex_if_do_html:`*. These functions are documented on page* ??.)

`\stex_suppress_html:n`

Whether to (locally) produce HTML output

```
\cs_new_protected:Nn \stex_suppress_html:n {
  \exp_args:Nne \use:nn {
    \bool_set_false:N \l_stex_html_do_output_bool
    #1
  }{
    \stex_if_do_html:T {
      \bool_set_true:N \l_stex_html_do_output_bool
    }
  }
}
```

(*End definition for* `\stex_suppress_html:n`*. This function is documented on page* ??.)

\stex_annotate:nnn
\stex_annotate_env
\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 (LaTeXML, RusTeX, `pdflatex`).

The `pdflatex`-macros largely do nothing; the RusTeX-implementations are pretty clear in what they do, the LaTeXML-implementations resort to perl bindings.

```
\rustex_if:TF{
  \cs_new_protected:Nn \stex_annotate:nnn {
    \__stex_annotate_checkempty:n { #3 }
    \rustex_annotate_HTML:nn {
      property="stex:#1" ~
      resource="#2"
    } {
      \mode_if_vertical:TF{
        \tl_use:N \l__stex_annotate_arg_tl\par
      }{
        \tl_use:N \l__stex_annotate_arg_tl
      }
    }
  }
  \cs_new_protected:Nn \stex_annotate_invisible:n {
    \__stex_annotate_checkempty:n { #1 }
    \rustex_annotate_HTML:nn {
      stex:visible="false" ~
      style:display="none"
    } {
      \mode_if_vertical:TF{
        \tl_use:N \l__stex_annotate_arg_tl\par
      }{
        \tl_use:N \l__stex_annotate_arg_tl
      }
    }
  }
  \cs_new_protected:Nn \stex_annotate_invisible:nnn {
    \__stex_annotate_checkempty:n { #3 }
    \rustex_annotate_HTML:nn {
      property="stex:#1" ~
      resource="#2" ~
      stex:visible="false" ~
      style:display="none"
    } {
      \mode_if_vertical:TF{
        \tl_use:N \l__stex_annotate_arg_tl\par
      }{
        \tl_use:N \l__stex_annotate_arg_tl
      }
    }
  }
  \NewDocumentEnvironment{stex_annotate_env} { m m } {
    \par
    \rustex_annotate_HTML_begin:n {
      property="stex:#1" ~
      resource="#2"
    }
  }{
    \par\rustex_annotate_HTML_end:
  }
}{
  \latexml_if:TF {
    \cs_new_protected:Nn \stex_annotate:nnn {
      \__stex_annotate_checkempty:n { #3 }
      \mode_if_math:TF {
        \cs:w latexml@annotate@math\cs_end:{#1}{#2}{
          \tl_use:N \l__stex_annotate_arg_tl
        }
      }{
        \cs:w latexml@annotate@text\cs_end:{#1}{#2}{
          \tl_use:N \l__stex_annotate_arg_tl
        }
      }
    }
    \cs_new_protected:Nn \stex_annotate_invisible:n {
      \__stex_annotate_checkempty:n { #1 }
      \mode_if_math:TF {
        \cs:w latexml@invisible@math\cs_end:{
          \tl_use:N \l__stex_annotate_arg_tl
        }
      } {
        \cs:w latexml@invisible@text\cs_end:{
          \tl_use:N \l__stex_annotate_arg_tl
        }
      }
    }
    \cs_new_protected:Nn \stex_annotate_invisible:nnn {
      \__stex_annotate_checkempty:n { #3 }
      \cs:w latexml@annotate@invisible\cs_end:{#1}{#2}{
        \tl_use:N \l__stex_annotate_arg_tl
      }
    }
    \NewDocumentEnvironment{stex_annotate_env} { m m } {
      \par\begin{latexml@annotateenv}{#1}{#2}
    }{
      \par\end{latexml@annotateenv}
    }
  }{
    \cs_new_protected:Nn \stex_annotate:nnn {#3}
    \cs_new_protected:Nn \stex_annotate_invisible:n {}
    \cs_new_protected:Nn \stex_annotate_invisible:nnn {}
    \NewDocumentEnvironment{stex_annotate_env} { m m } {}{}
  }
}
```

(*End definition for* `\stex_annotate:nnn`*,* `\stex_annotate_invisible:n`*, and* `\stex_annotate_invisible:nnn`*. These functions are documented on page 10.*)

18.6 Languages

```
⟨@@=stex_language⟩
```

\c_stex_languages_prop
\c_stex_language_abbrevs_prop

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

```
\prop_const_from_keyval:Nn \c_stex_languages_prop {
  en = english ,
  de = ngerman ,
  ar = arabic ,
  bg = bulgarian ,
  ru = russian ,
  fi = finnish ,
  ro = romanian ,
  tr = turkish ,
  fr = french
}

\prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop {
  english   = en ,
  ngerman   = de ,
  arabic    = ar ,
  bulgarian = bg ,
  russian   = ru ,
  finnish   = fi ,
  romanian  = ro ,
  turkish   = tr ,
  french    = fr
}
% todo: chinese simplified (zhs)
%       chinese traditional (zht)
```

(*End definition for* \c_stex_languages_prop *and* \c_stex_language_abbrevs_prop*. These variables are documented on page 10.*)

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

```
\clist_if_empty:NF \c_stex_languages_clist {
  \clist_clear:N \l_tmpa_clist
  \clist_map_inline:Nn \c_stex_languages_clist {
    \prop_get:NnNTF \c_stex_languages_prop { #1 } \l_tmpa_str {
      \clist_put_right:No \l_tmpa_clist \l_tmpa_str
    } {
      \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
    }
  }
  \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
  \RequirePackage[\clist_use:Nn \l_tmpa_clist,]{babel}
}
```

18.7 Activating/Deactivating Macros

\stex_deactivate_macro:Nn

```
\cs_new_protected:Nn \stex_deactivate_macro:Nn {
  \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
  \def#1{
    \msg_error:nnxx{stex}{error/deactivated-macro}{#1}{#2}
  }
}
```

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

`\stex_reactivate_macro:N`

```
\cs_new_protected:Nn \stex_reactivate_macro:N {
  \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname
}
```

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

```
⟨/package⟩
```

Chapter 19

sTeX -MathHub Implementation

```
⟨*package⟩

%%%%%%%%%%%%%   mathhub.dtx   %%%%%%%%%%%%%

⟨@@=stex_path⟩
```

Warnings and error messages

```
\msg_new:nnn{stex}{error/norepository}{
  No~archive~#1~found~in~#2
}
\msg_new:nnn{stex}{error/notinarchive}{
  Not~currently~in~an~archive,~but~\detokenize{#1}~
  needs~one!
}
\msg_new:nnn{stex}{error/nofile}{
  \detokenize{#1}~could~not~find~file~#2
}
```

19.1 Generic Path Handling

We treat paths as LaTeX3-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

```
\cs_new_protected:Nn \stex_path_from_string:Nn {
  \str_set:Nx \l_tmpa_str { #2 }
  \str_if_empty:NTF \l_tmpa_str {
    \seq_clear:N #1
  }{
    \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
    \sys_if_platform_windows:T{
      \seq_clear:N \l_tmpa_tl
      \seq_map_inline:Nn #1 {
        \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
        \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
      }
      \seq_set_eq:NN #1 \l_tmpa_tl
    }
    \stex_path_canonicalize:N #1
  }
}
\cs_generate_variant:Nn \stex_path_from_string:Nn
  { NV, cn, cV }
```

(*End definition for* `\stex_path_from_string:Nn`. *This function is documented on page 11.*)

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

```
\cs_new_protected:Nn \stex_path_to_string:NN {
  \exp_args:NNe \str_set:Nn #2 { \seq_use:Nn #1 / }
}

\cs_new:Nn \stex_path_to_string:N {
  \seq_use:Nn #1 /
}
```

(*End definition for* `\stex_path_to_string:NN` *and* `\stex_path_to_string:N`. *These functions are documented on page 11.*)

`\c__stex_path_dot_str`
`\c__stex_path_up_str`

. and .., respectively.

```
\str_const:Nn \c__stex_path_dot_str {.}
\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.

```
\cs_new_protected:Nn \stex_path_canonicalize:N {
  \seq_if_empty:NF #1 {
    \seq_clear:N \l_tmpa_seq
    \seq_get_left:NN #1 \l_tmpa_tl
    \str_if_empty:NT \l_tmpa_tl {
      \seq_put_right:Nn \l_tmpa_seq {}
    }
    \seq_map_inline:Nn #1 {
      \str_set:Nn \l_tmpa_tl { ##1 }
      \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_dot_str {} {
        \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
          \seq_if_empty:NTF \l_tmpa_seq {
            \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
              \c__stex_path_up_str
            }
          }{
            \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
            \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
              \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
                \c__stex_path_up_str
              }
            }{
              \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
            }
          }
        }{
          \str_if_empty:NF \l_tmpa_tl {
            \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
          }
        }
      }
    }
    \seq_gset_eq:NN #1 \l_tmpa_seq
  }
}
```

(*End definition for* `\stex_path_canonicalize:N`*. This function is documented on page 11.*)

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

```
\prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
  \seq_if_empty:NTF #1 {
    \prg_return_false:
  }{
    \seq_get_left:NN #1 \l_tmpa_tl
    \str_if_empty:NTF \l_tmpa_tl {
      \prg_return_true:
    }{
      \prg_return_false:
    }
  }
}
```

(*End definition for* `\stex_path_if_absolute:NTF`*. This function is documented on page 11.*)

19.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```
\str_new:N\l_stex_kpsewhich_return_str
\cs_new_protected:Nn \stex_kpsewhich:n {
  \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
  \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
  \tl_trim_spaces:N \l_stex_kpsewhich_return_str
}
```

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

We determine the PWD

`\c_stex_pwd_seq`
`\c_stex_pwd_str`

```
\sys_if_platform_windows:TF{
  \stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
}{
  \stex_kpsewhich:n{-var-value~PWD}
}

\stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
\stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
\stex_debug:nn {mathhub} {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 11.*)

19.3 File Hooks and Tracking

```
⟨@@=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 sTeX-purposes.

`\g__stex_files_stack` keeps track of file changes

```
\seq_gclear_new:N\g__stex_files_stack
```

(*End definition for* `\g__stex_files_stack`*.*)

`\c_stex_mainfile_seq`
`\c_stex_mainfile_str`

```
\str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
\stex_path_from_string:Nn \c_stex_mainfile_seq
  \c_stex_mainfile_str
```

(*End definition for* `\c_stex_mainfile_seq` *and* `\c_stex_mainfile_str`*. These variables are documented on page 11.*)

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

```
\seq_gclear_new:N\g_stex_currentfile_seq
\AddToHook{file/before}{
  \stex_path_from_string:Nn\g_stex_currentfile_seq{\CurrentFilePath}
  \stex_path_if_absolute:NTF\g_stex_currentfile_seq{
    \exp_args:NNe\seq_put_right:Nn\g_stex_currentfile_seq{\CurrentFile}
  }{
    \stex_path_from_string:Nn\g_stex_currentfile_seq{
      \c_stex_pwd_str/\CurrentFilePath/\CurrentFile
    }
  }
  \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
  \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
}
\AddToHook{file/after}{
  \seq_if_empty:NF\g__stex_files_stack{
    \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
  }
  \seq_if_empty:NTF\g__stex_files_stack{
    \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
  }{
    \seq_get:NN\g__stex_files_stack\l_tmpa_seq
    \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
  }
}
```

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

19.4 MathHub Repositories

```
⟨@@=stex_mathhub⟩
```

\mathhub
\c_stex_mathhub_seq
\c_stex_mathhub_str

```
\str_if_empty:NTF\mathhub{
  \stex_kpsewhich:n{-var-value~MATHHUB}
  \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str

  \str_if_empty:NTF\c_stex_mathhub_str{
    \msg_warning:nn{stex}{warning/nomathhub}
  }{
    \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
    \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
  }
}{
  \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
  \stex_path_if_absolute:NF \c_stex_mathhub_seq {
    \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
      \c_stex_pwd_str/\mathhub
    }
  }
  \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
  \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
}
```

(*End definition for* \mathhub*,* \c_stex_mathhub_seq*, and* \c_stex_mathhub_str*. These variables are documented on page 12.*)

__stex_mathhub_do_manifest:n

```
\cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
  \str_set:Nx \l_tmpa_str { #1 }
  \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
    \prop_new:c { c_stex_mathhub_#1_manifest_prop }
    \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
    \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
    \__stex_mathhub_find_manifest:N \l_tmpa_seq
    \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
      \msg_error:nnxx{stex}{error/norepository}{#1}{
        \stex_path_to_string:N \c_stex_mathhub_str
      }
    } {
      \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
    }
  }
}
```

(*End definition for* __stex_mathhub_do_manifest:n*.*)

\l__stex_mathhub_manifest_file_seq

```
\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:

```
\cs_new_protected:Nn \__stex_mathhub_find_manifest:N {
  \seq_set_eq:NN\l_tmpa_seq #1
  \bool_set_true:N\l_tmpa_bool
  \bool_while_do:Nn \l_tmpa_bool {
    \seq_if_empty:NTF \l_tmpa_seq {
      \bool_set_false:N\l_tmpa_bool
    }{
      \file_if_exist:nTF{
        \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
      }{
        \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
        \bool_set_false:N\l_tmpa_bool
      }{
        \file_if_exist:nTF{
          \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
        }{
          \seq_put_right:Nn\l_tmpa_seq{META-INF}
          \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
          \bool_set_false:N\l_tmpa_bool
        }{
          \file_if_exist:nTF{
            \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
          }{
            \seq_put_right:Nn\l_tmpa_seq{meta-inf}
            \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
            \bool_set_false:N\l_tmpa_bool
          }{
            \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
          }
        }
      }
    }
  }
  \seq_set_eq:NN\l__stex_mathhub_manifest_file_seq\l_tmpa_seq
}
```

(*End definition for __stex_mathhub_find_manifest:N.*)

\c__stex_mathhub_manifest_ior

File variable used for MANIFEST-files

```
\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:

```
\cs_new_protected:Nn \__stex_mathhub_parse_manifest:n {
  \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
  \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
  \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
    \str_set:Nn \l_tmpa_str {##1}
    \exp_args:NNoo \seq_set_split:Nnn
        \l_tmpb_seq \c_colon_str \l_tmpa_str
    \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
      \exp_args:NNe \str_set:Nn \l_tmpb_tl {
        \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
      }
      \exp_args:No \str_case:nnTF \l_tmpa_tl {
        {id} {
          \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
            { id } \l_tmpb_tl
        }
        {narration-base} {
          \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
            { narr } \l_tmpb_tl
        }
        {url-base} {
          \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
            { docurl } \l_tmpb_tl
        }
        {source-base} {
          \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
            { ns } \l_tmpb_tl
        }
        {ns} {
          \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
            { ns } \l_tmpb_tl
        }
        {dependencies} {
          \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
            { deps } \l_tmpb_tl
        }
      }{}{}
    }{}
  }
  \ior_close:N \c__stex_mathhub_manifest_ior
}
```

(*End definition for* `\__stex_mathhub_parse_manifest:n`*.*)

\stex_set_current_repository:n

```
\cs_new_protected:Nn \stex_set_current_repository:n {
  \stex_require_repository:n { #1 }
  \prop_set_eq:Nc \l_stex_current_repository_prop {
    c_stex_mathhub_#1_manifest_prop
  }
}
```

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

\stex_require_repository:n

```
\cs_new_protected:Nn \stex_require_repository:n {
  \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
    \stex_debug:nn{mathhub}{Opening~archive:~#1}
    \__stex_mathhub_do_manifest:n { #1 }
    \exp_args:Nx \stex_add_to_sms:n {
      \prop_const_from_keyval:cn { c_stex_mathhub_#1_manifest_prop } {
        id   = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } {  id  } ,
        ns   = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } {  ns  } ,
        narr = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { narr } ,
        deps = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { deps }
      }
    }
  }
}
```

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

`\l_stex_current_repository_prop` Current MathHub repository

```
\prop_new:N \l_stex_current_repository_prop

\__stex_mathhub_find_manifest:N \c_stex_pwd_seq
\seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
  \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
} {
  \__stex_mathhub_parse_manifest:n { main }
  \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
    \l_tmpa_str
  \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str _manifest_prop }
    \c_stex_mathhub_main_manifest_prop
  \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
  \stex_debug:nn{mathhub}{Current~repository:~
    \prop_item:Nn \l_stex_current_repository_prop {id}
  }
}
```

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

`\stex_in_repository:nn` Executes the code in the second argument in the context of the repository whose ID is provided as the first argument.

```
\cs_new_protected:Nn \stex_in_repository:nn {
  \str_set:Nx \l_tmpa_str { #1 }
  \cs_set:Npn \l_tmpa_cs ##1 { #2 }
  \str_if_empty:NTF \l_tmpa_str {
    \exp_args:Ne \l_tmpa_cs{
      \prop_item:Nn \l_stex_current_repository_prop { id }
    }
  }{
    \stex_require_repository:n \l_tmpa_str
    \str_set:Nx \l_tmpa_str { #1 }
    \exp_args:Nne \use:nn {
      \stex_set_current_repository:n \l_tmpa_str
      \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
    }{
       \stex_set_current_repository:n {
        \prop_item:Nn \l_stex_current_repository_prop { id }
       }
    }
  }
}
```

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

\inputref
\stex_inputref:nn

```
\newif \ifinputref \inputreffalse

\cs_new_protected:Nn \stex_inputref:nn {
  \stex_in_repository:nn {#1} {
    \ifinputref
      \input{ \c_stex_mathhub_str / ##1 / source / #2 }
    \else
      \inputreftrue
      \input{ \c_stex_mathhub_str / ##1 / source / #2 }
      \inputreffalse
    \fi
  }
}
\NewDocumentCommand \inputref { O{} m}{
  \stex_inputref:nn{ #1 }{ #2 }
}

\cs_new_protected:Nn \stex_mhbibresource:nn {
  \stex_in_repository:nn {#1} {
    \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
  }
}
\newcommand\addmhbibresource[2][]{
  \stex_mhbibresource:nn{ #1 }{ #2 }
}
```

(*End definition for* `\inputref` *and* `\stex_inputref:nn`*. These functions are documented on page 13.*)

\mhpath

```
  \def \mhpath #1 #2 {
    \exp_args:Ne \str_if_eq:nnTF{#1}{}{
      \c_stex_mathhub_str /
        \prop_item:Nn \l_stex_current_repository_prop { id }
        / source / #2
    }{
      \c_stex_mathhub_str / #1 / source / #2
    }
  }
```

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

\libinput

```
\cs_new_protected:Npn \libinput #1 {
  \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
    \msg_error:nnn{stex}{error/notinarchive}\libinput
  }
  \bool_set_false:N \l_tmpa_bool
  \tl_clear:N \l_tmpa_tl
  \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
  \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
  \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str
  \seq_pop_left:NNT \l_tmpb_seq \l_tmpb_str {
    \seq_put_right:No \l_tmpa_seq \l_tmpb_str
    \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
      / meta-inf / lib / #1.tex}{
        \bool_set_true:N \l_tmpa_bool
        \tl_put_right:Nx \l_tmpa_tl {
          \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
          / meta-inf / lib / #1.tex}
        }
      }{}
  }
  \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
    / \l_tmpa_str / lib / #1.tex
  }{
    \bool_set_true:N \l_tmpa_bool
    \tl_put_right:Nx \l_tmpa_tl {
      \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
      / \l_tmpa_str / lib / #1.tex}
    }
  }{}
  \bool_if:NF \l_tmpa_bool {
    \msg_error:nnnx{stex}{error/nofile}\libinput{#1.tex}
  }
  \l_tmpa_tl
}
```

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

```
⟨/package⟩
```

Chapter 20

sTeX -References Implementation

```
⟨*package⟩

%%%%%%%%%%%%%   references.dtx   %%%%%%%%%%%%%

%\RequirePackage{hyperref}
%\RequirePackage{cleveref}
⟨@@=stex_refs⟩
```

Warnings and error messages

```

\iow_new:N \c__stex_refs_refs_iow
\AddToHook{begindocument}{
  \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
}
\AddToHook{enddocument}{
  \iow_close:N \c__stex_refs_refs_iow
}

\str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}

\NewDocumentCommand \STEXreftitle { m } {
  \tl_gset:Nx \g__stex_refs_title_tl { #1 }
}
```

20.1 Document URIs and URLs

```
\seq_new:N \g__stex_refs_all_refs_seq

\str_new:N \l_stex_current_docns_str

\cs_new_protected:Nn \stex_get_document_uri: {
  \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
  \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
  \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
  \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
  \seq_put_right:No \l_tmpa_seq \l_tmpb_str

  \str_clear:N \l_tmpa_str
  \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
    \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
  }

  \str_if_empty:NTF \l_tmpa_str {
    \str_set:Nx \l_stex_current_docns_str {
      file:/\stex_path_to_string:N \l_tmpa_seq
    }
  }{
    \bool_set_true:N \l_tmpa_bool
    \bool_while_do:Nn \l_tmpa_bool {
      \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
      \exp_args:No \str_case:nnTF { \l_tmpb_str } {
        {source} { \bool_set_false:N \l_tmpa_bool }
      }{}{
        \seq_if_empty:NT \l_tmpa_seq {
          \bool_set_false:N \l_tmpa_bool
        }
      }
    }

    \seq_if_empty:NTF \l_tmpa_seq {
      \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
    }{
      \str_set:Nx \l_stex_current_docns_str {
        \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
      }
    }
  }
}
\str_new:N \l_stex_current_docurl_str
\cs_new_protected:Nn \stex_get_document_url: {
  \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
  \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
  \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
  \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
  \seq_put_right:No \l_tmpa_seq \l_tmpb_str

  \str_clear:N \l_tmpa_str
  \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
    \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
      \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
    }
  }

  \str_if_empty:NTF \l_tmpa_str {
    \str_set:Nx \l_stex_current_docurl_str {
      file:/\stex_path_to_string:N \l_tmpa_seq
    }
  }{
    \bool_set_true:N \l_tmpa_bool
    \bool_while_do:Nn \l_tmpa_bool {
      \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
      \exp_args:No \str_case:nnTF { \l_tmpb_str } {
        {source} { \bool_set_false:N \l_tmpa_bool }
      }{}{
        \seq_if_empty:NT \l_tmpa_seq {
          \bool_set_false:N \l_tmpa_bool
        }
      }
    }

    \seq_if_empty:NTF \l_tmpa_seq {
      \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
    }{
      \str_set:Nx \l_stex_current_docurl_str {
        \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
      }
    }
  }
}
```

20.2 Setting Reference Targets

```
\str_const:Nn \c__stex_refs_url_str{URL}
\str_const:Nn \c__stex_refs_ref_str{REF}
% @currentlabel -> number
% @currentlabelname -> title
% @currentHref -> name.number <- id of some kind
% \theH# -> \arabic{section}
% \the#  -> number
% \hyper@makecurrent{#}
\cs_new_protected:Nn \stex_ref_new_doc_target:n {
  \stex_get_document_uri:
  \str_set:Nx \l_tmpa_str { #1 }
  \str_if_empty:NT \l_tmpa_str {
    \int_zero:N \l_tmpa_int
    \bool_set_true:N \l_tmpa_bool
    \bool_while_do:Nn \l_tmpa_bool {
      \cs_if_exist:cTF {
        sref_\l_stex_current_docns_str\c_hash_str REF_\int_use:N \l_tmpa_int _type
      }{
        \int_incr:N \l_tmpa_int
      }{
        \str_set:Nx \l_tmpa_str { REF_\int_use:N \l_tmpa_int }
        \bool_set_false:N \l_tmpa_bool
      }
    }
  }
  \str_set:Nx \l_tmpa_str {
    \l_stex_current_docns_str\c_hash_str\l_tmpa_str
  }
  \seq_gput_right:No \g__stex_refs_all_refs_seq \l_tmpa_str
  \stex_if_smsmode:TF {
    \stex_get_document_url:
```

```
    \str_gset_eq:cN {sref_url_\l_tmpa_str _str}\l_stex_current_docurl_str
    \str_gset_eq:cN {sref_\l_tmpa_str _type}\c__stex_refs_url_str
  }{
    \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=~\expandafter{\@currentlabel\iffalse}{
    \exp_after:wN\label\exp_after:wN{sref_\l_tmpa_str}
    \str_gset:cn {sref_\l_tmpa_str _type}\c__stex_refs_ref_str
  }
}
```

```
\cs_new_protected:Nn \stex_ref_new_sym_target:n {
  \str_gset_eq:cN {sref_sym_#1_uri} \l_stex_current_docns_str
}
```

20.3 Using References

```
\str_new:N \l__stex_refs_indocument_str
\keys_define:nn { stex / sref } {
  linktext      .tl_set:N  = \l__stex_refs_linktext_tl ,
  fallback      .tl_set:N  = \l__stex_refs_fallback_tl ,
  pre           .tl_set:N  = \l__stex_refs_pre_tl ,
  post          .tl_set:N  = \l__stex_refs_post_tl ,
  %indoc         .str_set_x:N  = \l__stex_refs_repo_str ,
}

\bool_new:N \c__stex_refs_hyperref_bool
\bool_set_false:N \c__stex_refs_hyperref_bool
\AddToHook{begindocument}{
  \@ifpackageloaded{hyperref}{
    \bool_set_true:N \c__stex_refs_hyperref_bool
  }{}
}


\cs_new_protected:Nn \__stex_refs_args:n {
  \tl_clear:N \l__stex_refs_linktext_tl
  \tl_clear:N \l__stex_refs_fallback_tl
  \tl_clear:N \l__stex_refs_pre_tl
  \tl_clear:N \l__stex_refs_post_tl
  \str_clear:N \l__stex_refs_repo_str
  \keys_set:nn { stex / sref } { #1 }
}

\NewDocumentCommand \sref { O{} m}{
  \__stex_refs_args:n { #1 }
  \str_if_empty:NTF \l__stex_refs_indocument_str {
    \str_set:Nn \l_tmpa_str { #2 }
    \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
    \tl_set:Nn \l_tmpa_tl {
      \l__stex_refs_fallback_tl
    }
    \seq_map_inline:Nn \g__stex_refs_all_refs_seq {
      \str_set:Nn \l_tmpb_str { ##1 }
      \str_if_eq:eeT { \l_tmpa_str } {
        \str_range:Nnn \l_tmpb_str { -\l_tmpa_int }{ -1 }
      } {
        \seq_map_break:n {
          \tl_set:Nn \l_tmpa_tl {
            % doc uri in \l_tmpb_str
            \str_set:Nx \l_tmpa_str {sref_url_\l_tmpb_str _type}
            \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
              % reference
              \l__stex_refs_pre_tl\ref{sref_\l_tmpb_str}\l__stex_refs_post_tl
            }{
              % URL
              \if_bool:N \c__stex_refs_hyperref_bool {
                \exp_args:Nx \href{\use:c{sref_url_\l_tmpb_str _str}}{\l__stex_refs_fallback
              }{
                \l__stex_refs_fallback_tl
              }
            }
          }
        }
      }
    }
    \l_tmpa_tl
  }{
    % TODO
  }
}

⟨/package⟩
```

Chapter 21

sTEX -Modules Implementation

```
⟨*package⟩

%%%%%%%%%%%%%   modules.dtx   %%%%%%%%%%%%%

⟨@@=stex_modules⟩
```

Warnings and error messages

```
\msg_new:nnn{stex}{error/unknownmodule}{
  No~module~#1~found
}
\msg_new:nnn{stex}{error/syntax}{
  Syntax~error:~#1
}
\msg_new:nnn{stex}{error/siglanguage}{
  Module~#1~declares~signature~#2,~but~does~not~
  declare~its~language
}
```

\l_stex_current_module_prop

The current module:

```
\prop_new:N \l_stex_current_module_prop
```

(*End definition for* `\l_stex_current_module_prop`. *This variable is documented on page 15.*)

\l_stex_all_modules_seq

Stores all available modules

```
\seq_new:N \l_stex_all_modules_seq
```

(*End definition for* `\l_stex_all_modules_seq`. *This variable is documented on page 15.*)

\g_stex_modules_in_file_seq
\g_stex_module_files_prop

All modules sorted by containing file; used e.g. in `\importmodule`

```
\seq_new:N \g_stex_modules_in_file_seq
\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 16.*)

\stex_if_in_module_p:
\stex_if_in_module:*TF*

```
\prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
  \prop_if_empty:NTF \l_stex_current_module_prop
    \prg_return_false: \prg_return_true:
}
```

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

\stex_if_module_exists_p:n
\stex_if_module_exists:n*TF*

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

(*End definition for* \stex_if_module_exists:nTF*. This function is documented on page 16.*)

\stex_add_to_current_module:n
\STEXexport

Only allowed within modules:

```
\cs_new_protected:Nn \stex_add_to_current_module:n {
  \prop_get:NnN \l_stex_current_module_prop { content } \l_tmpa_tl
  \tl_put_right:Nn \l_tmpa_tl { #1 }
  \prop_put:Nno \l_stex_current_module_prop { content } { \l_tmpa_tl }
}
\cs_new_protected:Npn \STEXexport {
  \begingroup
  \newlinechar=-1\relax
  \endlinechar=-1\relax
  %\catcode`\ = 9\relax
  \expandafter\endgroup\STEXexport:n
}
\cs_new_protected:Nn \STEXexport:n {
  \ignorespaces #1
  \stex_add_to_current_module:n { \ignorespaces #1 }
  \stex_smsmode_set_codes:
}
\stex_deactivate_macro:Nn \STEXexport {module~environments}
```

(*End definition for* \stex_add_to_current_module:n *and* \STEXexport*. These functions are documented on page 16.*)

\stex_add_constant_to_current_module:n

```
\cs_new_protected:Nn \stex_add_constant_to_current_module:n {
  \str_set:Nx \l_tmpa_str { #1 }
  \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
  \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
  \prop_put:Nno \l_stex_current_module_prop { constants } \l_tmpa_seq
}
```

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

\stex_add_import_to_current_module:n

```
\cs_new_protected:Nn \stex_add_import_to_current_module:n {
  \str_set:Nx \l_tmpa_str { #1 }
  \prop_get:NnN \l_stex_current_module_prop { imports } \l_tmpa_seq
  \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
  \prop_put:Nno \l_stex_current_module_prop { imports } \l_tmpa_seq
}
```

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

\stex_modules_compute_namespace:nN Computer the appropriate namespace from the top-level namespace of a repository (`#1`) and a file path (`#2`).

```
\cs_new_protected:Nn \stex_modules_compute_namespace:nN {
  \str_set:Nx \l_tmpa_str { #1 }
  \seq_set_eq:NN \l_tmpa_seq #2
  % split off file extension
  \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
  \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
  \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
  \seq_put_right:No \l_tmpa_seq \l_tmpb_str

  \bool_set_true:N \l_tmpa_bool
  \bool_while_do:Nn \l_tmpa_bool {
    \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
    \exp_args:No \str_case:nnTF { \l_tmpb_str } {
      {source} { \bool_set_false:N \l_tmpa_bool }
    }{}{
      \seq_if_empty:NT \l_tmpa_seq {
        \bool_set_false:N \l_tmpa_bool
      }
    }
  }

  \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
  \str_if_empty:NTF \l_stex_modules_subpath_str {
    \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
  }{
    \str_set:Nx \l_stex_modules_ns_str {
      \l_tmpa_str/\l_stex_modules_subpath_str
    }
  }
}
```

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

Stores its return values in:

`\l_stex_modules_ns_str`

```
\str_new:N \l_stex_modules_ns_str
\str_new:N \l_stex_modules_subpath_str
```

(*End definition for* `\l_stex_modules_ns_str`. *This variable is documented on page* **??**.)

\stex_modules_current_namespace: Computes the current namespace based on the current MathHub repository (if existent) and the current file.

```
\cs_new_protected:Nn \stex_modules_current_namespace: {
  \str_clear:N \l_stex_modules_subpath_str
  \prop_get:NnNTF \l_stex_current_repository_prop { ns } \l_tmpa_str {
    \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
  }{
    % split off file extension
    \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
    \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
    \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
    \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
    \seq_put_right:No \l_tmpa_seq \l_tmpb_str
    \str_set:Nx \l_stex_modules_ns_str {
      file:/\stex_path_to_string:N \l_tmpa_seq
    }
  }
}
```

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

21.1 The module environment

`module` arguments:

```
\keys_define:nn { stex / module } {
  title         .str_set_x:N  = \l_stex_module_title_str ,
  ns            .str_set_x:N  = \l_stex_module_ns_str ,
  lang          .str_set_x:N  = \l_stex_module_lang_str ,
  sig           .str_set_x:N  = \l_stex_module_sig_str ,
  creators      .str_set_x:N  = \l_stex_module_creators_str ,
  contributors  .str_set_x:N  = \l_stex_module_contributors_str ,
  meta          .str_set_x:N  = \l_stex_module_meta_str ,
  srccite       .str_set_x:N  = \l_stex_module_srccite_str
}

\cs_new_protected:Nn \__stex_modules_args:n {
  \str_clear:N \l_stex_module_title_str
  \str_clear:N \l_stex_module_ns_str
  \str_clear:N \l_stex_module_lang_str
  \str_clear:N \l_stex_module_sig_str
  \str_clear:N \l_stex_module_creators_str
  \str_clear:N \l_stex_module_contributors_str
  \str_clear:N \l_stex_module_meta_str
  \str_clear:N \l_stex_module_srccite_str
  \keys_set:nn { stex / module } { #1 }
}

% module parameters here? In the body?

```

`\stex_module_setup:nn` Sets up a new module property list:

```
\cs_new_protected:Nn \stex_module_setup:nn {
  \str_set:Nx \l_stex_module_name_str { #2 }
  \__stex_modules_args:n { #1 }
```

First, we set up the name and namespace of the module.
Are we in a nested module?

```
  \stex_if_in_module:TF {
    % Nested module
    \prop_get:NnN \l_stex_current_module_prop
      { ns } \l_stex_module_ns_str
    \str_set:Nx \l_stex_module_name_str {
      \prop_item:Nn \l_stex_current_module_prop
```

```
        { name } / \l_stex_module_name_str
    }
  }{
    % not nested:
    \str_if_empty:NT \l_stex_module_ns_str {
      \stex_modules_current_namespace:
      \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
      \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
          / {\l_stex_module_ns_str}
      \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
      \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
        \str_set:Nx \l_stex_module_ns_str {
          \stex_path_to_string:N \l_tmpa_seq
        }
      }
    }
  }
```

Next, we determine the language of the module:

```
  \str_if_empty:NT \l_stex_module_lang_str {
    \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
    \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
    \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
    \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
    \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
      \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
        inferred~from~file~name}
      \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
    }
  }

  \str_if_empty:NF \l_stex_module_lang_str {
    \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
      \l_tmpa_str {
        \ltx@ifpackageloaded{babel}{
          \exp_args:Nx \selectlanguage { \l_tmpa_str }
        }{}
      } {
        \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
      }
  }
```

We check if we need to extend a signature module, and set `\l_stex_current_module_prop` accordingly:

```
  \str_if_empty:NTF \l_stex_module_sig_str {
    \str_clear:N \l_tmpa_str
    \seq_clear:N \l_tmpa_seq
    \tl_clear:N \l_tmpa_tl
    \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
      name      = \l_stex_module_name_str ,
      ns        = \l_stex_module_ns_str ,
      imports   = \exp_not:o { \l_tmpa_seq } ,
      constants = \exp_not:o { \l_tmpa_seq } ,
      content   = \exp_not:o { \l_tmpa_tl }  ,
      file      = \exp_not:o { \g_stex_currentfile_seq } ,
      lang      = \l_stex_module_lang_str ,
      sig       = \l_stex_module_sig_str ,
      meta      = \l_stex_module_meta_str
    }
  }{
    \str_if_empty:NT \l_stex_module_lang_str {
      \msg_error:nnxx{stex}{error/siglanguage}{
        \l_stex_module_ns_str?\l_stex_module_name_str
      }{\l_stex_module_sig_str}
    }

    \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
    \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
    \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
    \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
    \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
    \str_set:Nx \l_tmpa_str {
      \stex_path_to_string:N \l_tmpa_seq /
      \l_tmpa_str . \l_stex_module_sig_str .tex
    }
    \IfFileExists \l_tmpa_str {
      \exp_args:No \stex_in_smsmode:nn { \l_tmpa_str } {
        \seq_clear:N \l_stex_all_modules_seq
        \prop_clear:N \l_stex_current_module_prop
        \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
        \input { \l_tmpa_str }
      }
    }{
      \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
    }
    \stex_activate_module:n {
      \l_stex_module_ns_str ? \l_stex_module_name_str
    }
    \prop_set_eq:Nc \l_stex_current_module_prop {
      c_stex_module_
      \l_stex_module_ns_str ?
      \l_stex_module_name_str
      _prop
    }
  }
```

We load the metatheory:

```
  \str_if_empty:NT \l_stex_module_meta_str {
    \str_set:Nx \l_stex_module_meta_str {
      \c_stex_metatheory_ns_str ? Metatheory
    }
  }
  \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
    \exp_args:Nx \stex_add_to_current_module:n {
      \stex_activate_module:n {\l_stex_module_meta_str}
    }
    \stex_activate_module:n {\l_stex_module_meta_str}
  }
```

```
}
```

(*End definition for* `\stex_module_setup:nn`. *This function is documented on page 17.*)

module The `module` environment.

`\__stex_modules_begin_module:nn` implements `\begin{module}`

```
\cs_new_protected:Nn \__stex_modules_begin_module:nn {
  \stex_reactivate_macro:N \STEXexport
  \stex_reactivate_macro:N \importmodule
  \stex_reactivate_macro:N \symdecl
  \stex_reactivate_macro:N \notation
  \stex_reactivate_macro:N \symdef
  \stex_module_setup:nn{#1}{#2}

  \stex_debug:nn{modules}{
    New~module:\\
    Namespace:~\l_stex_module_ns_str\\
    Name:~\l_stex_module_name_str\\
    Language:~\l_stex_module_lang_str\\
    Signature:~\l_stex_module_sig_str\\
    Metatheory:~\l_stex_module_meta_str\\
    File:~\stex_path_to_string:N \g_stex_currentfile_seq
  }

  \seq_put_right:Nx \l_stex_all_modules_seq {
    \l_stex_module_ns_str ? \l_stex_module_name_str
  }

  \seq_gput_right:Nx  \g_stex_modules_in_file_seq
      { \l_stex_module_ns_str ? \l_stex_module_name_str }

  \stex_if_smsmode:TF {
    \stex_smsmode_set_codes:
  } {
    \begin{stex_annotate_env} {theory} {
      \l_stex_module_ns_str ? \l_stex_module_name_str
    }

    \stex_annotate_invisible:nnn{header}{} {
      \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
      \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
      \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
        \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
      }
    }
  }
  % TODO: Inherit metatheory for nested modules?
}
\iffalse \end{stex_annotate_env} \fi %^^A make syntax highlighting work again
```

(*End definition for* `\__stex_modules_begin_module:nn`.)

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

```
\cs_new_protected:Nn \__stex_modules_end_module: {
  \str_set:Nx \l_tmpa_str {
    c_stex_module_
    \prop_item:Nn \l_stex_current_module_prop { ns } ?
    \prop_item:Nn \l_stex_current_module_prop { name }
    _prop
  }
  %^^A \prop_new:c { \l_tmpa_str }
  \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
  \stex_debug:nn{modules}{Closing~module~\prop_item:Nn \l_stex_current_module_prop { name }}
}
```

(*End definition for* `\__stex_modules_end_module:`*.*)

@module The core environment, with no header

```
\iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
\NewDocumentEnvironment { @module } { O{} m } {
  \par
  \__stex_modules_begin_module:nn{#1}{#2}
} {
  \__stex_modules_end_module:
  \stex_if_smsmode:TF {
    \exp_args:Nx \stex_add_to_sms:n {
      \prop_gset_from_keyval:cn {
        c_stex_module_
        \prop_item:Nn \l_stex_current_module_prop { ns } ?
        \prop_item:Nn \l_stex_current_module_prop { name }
        _prop
      } {
        name      = \prop_item:cn { \l_tmpa_str } { name } ,
        ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
        imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
        constants = \prop_item:cn { \l_tmpa_str } { constants } ,
        content   = \prop_item:cn { \l_tmpa_str } { content } ,
        file      = \prop_item:cn { \l_tmpa_str } { file } ,
        lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
        sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
        meta      = \prop_item:cn { \l_tmpa_str } { meta }
      }
    }
  }{
    \end{stex_annotate_env}
  }
}
```

\stex_modules_heading: Code for document headers

```
\cs_if_exist:NTF \thesection {
  \newcounter{module}[section]
}{
  \newcounter{module}
}

\bool_if:NT \c_stex_showmods_bool {
  \latexml_if:F { \RequirePackage{mdframed} }
}

\cs_new_protected:Nn \stex_modules_heading: {
  \stepcounter{module}
  \par
  \bool_if:NT \c_stex_showmods_bool {
    \noindent{\textbf{Module} ~
      \cs_if_exist:NT \thesection {\thesection.}
      \themodule ~ [\l_stex_module_name_str]
    }
    \str_if_empty:NTF \l_stex_module_title_str {
    }{
      \quad(\l_stex_module_title_str)\hfill
    }\par
  }
  \edef\@currentlabel{Module~\thesection.\themodule~[\l_stex_module_name_str]}
  % TODO
  \stex_ref_new_doc_target:n \l_stex_module_name_str
}
```

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

Finally:

```
\NewDocumentEnvironment { module } { O{} m } {
  \bool_if:NT \c_stex_showmods_bool {
    \begin{mdframed}
  }
  \begin{@module}[#1]{#2}
  \stex_modules_heading:
}{
  \end{@module}
  \bool_if:NT \c_stex_showmods_bool {
    \end{mdframed}
  }
}
```

21.2 Invoking modules

`\STEXModule`
`\stex_invoke_module:n`

```
\NewDocumentCommand \STEXModule { m } {
  \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
  \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
  \tl_set:Nn \l_tmpa_tl {
    \msg_error:nnx{stex}{error/unknownmodule}{#1}
  }
  \seq_map_inline:Nn \l_stex_all_modules_seq {
    \str_set:Nn \l_tmpb_str { ##1 }
    \str_if_eq:eeT { \l_tmpa_str } {
      \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
    } {
      \seq_map_break:n {
        \tl_set:Nn \l_tmpa_tl {
          \stex_invoke_module:n { ##1 }
        }
      }
    }
  }
  \l_tmpa_tl
}

\cs_new_protected:Nn \stex_invoke_module:n {
  \stex_debug:nn{modules}{Invoking~module~#1}
  \peek_charcode_remove:NTF ! {
    \__stex_modules_invoke_uri:nN { #1 }
  } {
    \peek_charcode_remove:NTF ? {
      \__stex_modules_invoke_symbol:nn { #1 }
    } {
      \msg_error:nnx{stex}{error/syntax}{
        ?~or~!~expected~after~
        \c_backslash_str STEXModule{#1}
      }
    }
  }
}

\cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
  \str_set:Nn #2 { #1 }
}

\cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
  \stex_invoke_symbol:n{#1?#2}
}
```

(*End definition for* `\STEXModule` *and* `\stex_invoke_module:n`*. These functions are documented on page 18.*)

`\stex_activate_module:n`

```
\cs_new_protected:Nn \stex_activate_module:n {
  \stex_debug:nn{modules}{Activating~module~#1}
  \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
    \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
    \prop_item:cn { c_stex_module_#1_prop } { content }
  }
}
```

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

⟨/package⟩

Chapter 22

sTeX -Module Inheritance Implementation

```
⟨*package⟩

%%%%%%%%%%%%%   inheritance.dtx   %%%%%%%%%%%%%
```

22.1 SMS Mode

```
⟨@@=stex_smsmode⟩
```

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

```
\tl_new:N \g_stex_smsmode_allowedmacros_tl
\tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
\seq_new:N \g_stex_smsmode_allowedenvs_seq

\tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
  \makeatletter
  \makeatother
  \ExplSyntaxOn
  \ExplSyntaxOff
}

\tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
  \symdef
  \importmodule
  \notation
  \symdecl
  \STEXexport
}

\exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
  \tl_to_str:n {
    module,
    @module
  }
}
```

(*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 20.*)

`\stex_if_smsmode_p:`
`\stex_if_smsmode:TF`

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

(*End definition for* `\stex_if_smsmode:TF`*. This function is documented on page 20.*)

`\__stex_smsmode_if_catcodes_p:`
`__stex_smsmode_if_catcodes:TF`

Checks whether the SMS mode category code scheme is active.

```
\bool_new:N \g__stex_smsmode_catcode_bool
\bool_set_false:N \g__stex_smsmode_catcode_bool
\prg_new_conditional:Nnn \__stex_smsmode_if_catcodes: { p, T, F, TF } {
  \bool_if:NTF \g__stex_smsmode_catcode_bool
    \prg_return_true: \prg_return_false:
}
```

(*End definition for* `\__stex_smsmode_if_catcodes:TF`*.*)

`\stex_smsmode_set_codes:`

```
\cs_new_protected:Nn \stex_smsmode_set_codes: {
  \stex_if_smsmode:T {
    \__stex_smsmode_if_catcodes:F {
      \bool_gset_true:N \g__stex_smsmode_catcode_bool
      \exp_after:wN \char_gset_active_eq:NN
        \c_backslash_str \__stex_smsmode_cs:
      \tex_global:D \char_set_catcode_active:N \\
      \tex_global:D \char_set_catcode_other:N $
      \tex_global:D \char_set_catcode_other:N ^
      \tex_global:D \char_set_catcode_other:N _
      \tex_global:D \char_set_catcode_other:N &
      \tex_global:D \char_set_catcode_other:N ##
    }
  }
} \iffalse $ \fi % to make syntax highlighting work again
```

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

`\__stex_smsmode_unset_codes:`

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

```
\cs_new_protected:Nn \__stex_smsmode_unset_codes: {
  \__stex_smsmode_if_catcodes:T {
    \bool_gset_false:N \g__stex_smsmode_catcode_bool
    \exp_after:wN \tex_global:D \exp_after:wN
      \char_set_catcode_escape:N \c_backslash_str
    \tex_global:D \char_set_catcode_math_toggle:N $
    \tex_global:D \char_set_catcode_math_superscript:N ^
    \tex_global:D \char_set_catcode_math_subscript:N _
    \tex_global:D \char_set_catcode_alignment:N &
    \tex_global:D \char_set_catcode_parameter:N ##
  }
} \iffalse $ \fi % to make syntax highlighting work again
```

(*End definition for* `\__stex_smsmode_unset_codes:`.)

`\stex_in_smsmode:nn`

```
\cs_new_protected:Nn \stex_in_smsmode:nn {
  \vbox_set:Nn \l_tmpa_box {
    \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
    \bool_gset_true:N \g__stex_smsmode_bool
    \stex_smsmode_set_codes:
    #2
    \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
    \stex_if_smsmode:F {
      \__stex_smsmode_unset_codes:
    }
  }
  \box_clear:N \l_tmpa_box
}
```

(*End definition for* `\stex_in_smsmode:nn`. *This function is documented on page 21.*)

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

```
\cs_new_protected:Nn \__stex_smsmode_cs: {
  \str_clear:N \l_tmpa_str
  \peek_analysis_map_inline:n {
    % #1: token (one expansion)
    % #2: charcode
    % #3 catcode
    \token_if_eq_charcode:NNTF ##3 B {
      % token is a letter
      \exp_args:NNo \str_put_right:Nn \l_tmpa_str { ##1 }
    } {
      \str_if_empty:NTF \l_tmpa_str {
        % we don't allow (or need) single non-letter CSs
        % for now
        \peek_analysis_map_break:
      }{
        \str_if_eq:onTF \l_tmpa_str { begin } {
          \peek_analysis_map_break:n {
            \exp_after:wN \__stex_smsmode_checkbegin:n ##1
          }
        } {
          \str_if_eq:onTF \l_tmpa_str { end } {
            \peek_analysis_map_break:n {
              \exp_after:wN \__stex_smsmode_checkend:n ##1
            }
          } {
          \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
          \exp_args:NNNo \exp_args:NNo \tl_if_in:NnTF
            \g_stex_smsmode_allowedmacros_tl
              { \use:c{\l_tmpa_str} } {
              \stex_debug:nn{modules}{Executing~1:~\l_tmpa_str}
              \peek_analysis_map_break:n {
                \exp_after:wN \l_tmpa_tl ##1
              }
            } {
              \exp_args:NNNo \exp_args:NNo \tl_if_in:NnTF
              \g_stex_smsmode_allowedmacros_escape_tl
                { \use:c{\l_tmpa_str} } {
                \__stex_smsmode_unset_codes:
                \stex_debug:nn{modules}{Executing~2:~\l_tmpa_str}
                % TODO \__stex_smsmode_rescan_cs:
%                 \int_compare:nNnTF {##2} = {92} {
%                   \peek_analysis_map_break:n {
%                     \__stex_smsmode_unset_codes:
%                     \__stex_smsmode_rescan_cs:
%                   }
%                 } {
                  \peek_analysis_map_break:n {
                    \exp_after:wN \l_tmpa_tl ##1
                  }
%                 }
              } {
                  \int_compare:nNnTF {##2} = {92} {
                    \peek_analysis_map_break:n { \__stex_smsmode_cs: }
                  }{
                    \peek_analysis_map_break:n { \exp_after:wN\relax ##1 }
                  }
              }
            }
          }
        }
      }
    }
  }
}
```

(*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:

```
\cs_new_protected:Nn \__stex_smsmode_rescan_cs: {
  \str_clear:N \l_tmpb_str
  \peek_analysis_map_inline:n {
    \token_if_eq_charcode:NNTF ##3 B {
      % token is a letter
      \exp_args:NNo \str_put_right:Nn \l_tmpb_str { ##1 }
    } {
      \peek_analysis_map_break:n {
        \exp_after:wN \use:c \exp_after:wN {
          \exp_after:wN \l_tmpa_str\exp_after:wN
        } \use:c { \l_tmpb_str \exp_after:wN } ##1
      }
    }
  }
}
```

(*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.

```
\cs_new_protected:Nn \__stex_smsmode_checkbegin:n {
  \str_set:Nn \l_tmpa_str { #1 }
  \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
    \__stex_smsmode_unset_codes:
    \begin{#1}
  }
}
```

(*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.

```
\cs_new_protected:Nn \__stex_smsmode_checkend:n {
  \str_set:Nn \l_tmpa_str { #1 }
  \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
    \end{#1}
  }
}
```

(*End definition for* __*stex_smsmode_checkend:n*.)

22.2 Inheritance

```
⟨@@=stex_importmodule⟩
```

\stex_import_module_uri:nn

```
\cs_new_protected:Nn \stex_import_module_uri:nn {
  \str_set:Nx \l__stex_importmodule_archive_str { #1 }
  \str_set:Nn \l__stex_importmodule_path_str { #2 }

  \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l__stex_importmodule_path_str }
  \seq_pop_right:NN \l_tmpb_seq \l__stex_importmodule_name_str
  \str_set:Nx \l__stex_importmodule_path_str { \seq_use:Nn \l_tmpb_seq ? }

  \stex_modules_current_namespace:
  \bool_lazy_all:nTF {
    {\str_if_empty_p:N \l__stex_importmodule_archive_str}
    {\str_if_empty_p:N \l__stex_importmodule_path_str}
    {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } }
  }{
    \str_set_eq:NN \l__stex_importmodule_path_str \l_stex_modules_subpath_str
    \str_set_eq:NN \l_stex_module_ns
  }{
    \str_if_empty:NT \l__stex_importmodule_archive_str {
      \prop_if_empty:NF \l_stex_current_repository_prop {
        \prop_get:NnN \l_stex_current_repository_prop { id } \l__stex_importmodule_archive_s
      }
    }
    \str_if_empty:NTF \l__stex_importmodule_archive_str {
      \str_if_empty:NF \l__stex_importmodule_path_str {
        \str_set:Nx \l_stex_module_ns_str {
          \l_stex_module_ns_str / \l__stex_importmodule_path_str
        }
      }
    }{
      \stex_require_repository:n \l__stex_importmodule_archive_str
      \prop_get:cnN { c_stex_mathhub_\l__stex_importmodule_archive_str _manifest_prop } { ns
        \l_stex_module_ns_str
      \str_if_empty:NF \l__stex_importmodule_path_str {
        \str_set:Nx \l_stex_module_ns_str {
          \l_stex_module_ns_str / \l__stex_importmodule_path_str
        }
      }
    }
  }
}
```

(*End definition for* `\stex_import_module_uri:nn`. *This function is documented on page 23.*)

\l__stex_importmodule_name_str
\l__stex_importmodule_archive_str
\l__stex_importmodule_path_str
\l__stex_importmodule_file_str

Store the return values of `\stex_import_module_uri:nn`.

```
\str_new:N \l__stex_importmodule_name_str
\str_new:N \l__stex_importmodule_archive_str
\str_new:N \l__stex_importmodule_path_str
\str_new:N \g__stex_importmodule_file_str
```

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

\stex_import_require_module:nnnn

{⟨*ns*⟩} {⟨*archive-ID*⟩} {⟨*path*⟩} {⟨*name*⟩}

```
\cs_new_protected:Nn \stex_import_require_module:nnnn {
  \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {

    % archive
    \str_set:Nx \l_tmpa_str { #2 }
    \str_if_empty:NTF \l_tmpa_str {
      \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
    } {
      \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
      \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
      \seq_put_right:Nn \l_tmpa_seq { source }
    }

    % path
    \str_set:Nx \l_tmpb_str { #3 }
    \str_if_empty:NTF \l_tmpb_str {
      \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }

      \ltx@ifpackageloaded{babel} {
        \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
            { \languagename } \l_tmpb_str {
              \msg_error:nnx{stex}{error/unknownlanguage}{\languagename}
            }
      } {
        \str_clear:N \l_tmpb_str
      }

      \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
      \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
        \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
      }{
        \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
        \IfFileExists{ \l_tmpa_str.tex }{
          \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
        }{
          % try english as default
          \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
          \IfFileExists{ \l_tmpa_str.en.tex }{
            \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
          }{
            \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
          }
        }
      }

    } {
      \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
      \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq

      \ltx@ifpackageloaded{babel} {
        \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
            { \languagename } \l_tmpb_str {
              \msg_error:nnx{stex}{error/unknownlanguage}{\languagename}
            }
      } {
        \str_clear:N \l_tmpb_str
      }

      \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str

      \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
      \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
        \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
      }{
        \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
        \IfFileExists{ \l_tmpa_str/#4.tex }{
          \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
        }{
          % try english as default
          \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
          \IfFileExists{ \l_tmpa_str/#4.en.tex }{
            \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
          }{
            \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
            \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
              \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
            }{
              \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
              \IfFileExists{ \l_tmpa_str.tex }{
                \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
              }{
                % try english as default
                \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
                \IfFileExists{ \l_tmpa_str.en.tex }{
                    \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
                  }{
                    \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
                  }
                }
              }
            }
          }
        }
      }

    \seq_set_eq:NN \l_tmpa_seq \g_stex_modules_in_file_seq
    \seq_clear:N \g_stex_modules_in_file_seq
%    \exp_args:Nnx \use:nn {
      \exp_args:No \stex_in_smsmode:nn { \g__stex_importmodule_file_str } {
        \seq_clear:N \l_stex_all_modules_seq
        \prop_clear:N \l_stex_current_module_prop
        \str_set:Nx \l_tmpb_str { #2 }
        \str_if_empty:NF \l_tmpb_str {
          \stex_set_current_repository:n { #2 }
        }
        \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
        \input { \g__stex_importmodule_file_str }
      }
%    }{

%    }
    \prop_gput:Noo \g_stex_module_files_prop
    \g__stex_importmodule_file_str \g_stex_modules_in_file_seq
    \seq_set_eq:NN \g_stex_modules_in_file_seq \l_tmpa_seq

    \stex_if_module_exists:nF { #1 ? #4 } {
      \msg_error:nnx{stex}{error/unknownmodule}{
        #1?#4~(in~file~\g__stex_importmodule_file_str)
      }
    }
  }
  \stex_activate_module:n { #1 ? #4 }
}
```

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

`\importmodule`

```
\NewDocumentCommand \importmodule { O{} m } {
  \stex_import_module_uri:nn { #1 } { #2 }
  \stex_debug:nn{modules}{Importing~module:~
    \l_stex_module_ns_str ? \l__stex_importmodule_name_str
  }
  \stex_if_smsmode:F {
    \stex_import_require_module:nnnn
    { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
    { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
    \stex_annotate_invisible:nnn
      {import} {\l_stex_module_ns_str ? \l__stex_importmodule_name_str} {}
  }
  \exp_args:Nx \stex_add_to_current_module:n {
    \stex_import_require_module:nnnn
    { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
    { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
  }
  \exp_args:Nx \stex_add_import_to_current_module:n {
    \l_stex_module_ns_str ? \l__stex_importmodule_name_str
  }
  \stex_smsmode_set_codes:
}
\stex_deactivate_macro:Nn \importmodule {module~environments}
```

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

`\usemodule`

```
\NewDocumentCommand \usemodule { O{} m } {
  \stex_if_smsmode:F {
    \stex_import_module_uri:nn { #1 } { #2 }
    \stex_import_require_module:nnnn
    { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
    { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
    \stex_annotate_invisible:nnn
      {usemodule} {\l_stex_module_ns_str ? \l__stex_importmodule_name_str} {}
  }
  \stex_smsmode_set_codes:
}
```

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

```
⟨/package⟩
```

Chapter 23

sTEX -Symbols Implementation

```
⟨*package⟩

%%%%%%%%%%%%%   symbols.dtx   %%%%%%%%%%%%%

```

Warnings and error messages

23.1 Symbol Declarations

```
⟨@@=stex_symdecl⟩
```

\l_stex_all_symbols_seq Stores all available symbols

```
\seq_new:N \l_stex_all_symbols_seq
```

(*End definition for* \l_stex_all_symbols_seq*. This variable is documented on page 25.*)

\STEXsymbol

```
\NewDocumentCommand \STEXsymbol { m } {
  \stex_get_symbol:n { #1 }
  \exp_args:No
  \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
}
```

(*End definition for* \STEXsymbol*. This function is documented on page 27.*)

symdecl arguments:

```
\keys_define:nn { stex / symdecl } {
  name        .str_set_x:N  = \l_stex_symdecl_name_str ,
  local       .bool_set:N  = \l_stex_symdecl_local_bool ,
  args        .str_set_x:N  = \l_stex_symdecl_args_str ,
  type        .tl_set:N    = \l_stex_symdecl_type_tl ,
  align       .str_set:N    = \l_stex_symdecl_align_str , % TODO(?)
  gfc         .str_set:N    = \l_stex_symdecl_gfc_str , % TODO(?)
  specializes .str_set:N    = \l_stex_symdecl_specializes_str , % TODO(?)
  def         .tl_set:N    = \l_stex_symdecl_definiens_tl
}
```

```
\bool_new:N \l_stex_symdecl_make_macro_bool

\cs_new_protected:Nn \__stex_symdecl_args:n {
  \str_clear:N \l_stex_symdecl_name_str
  \str_clear:N \l_stex_symdecl_args_str
  \bool_set_false:N \l_stex_symdecl_local_bool
  \tl_clear:N \l_stex_symdecl_type_tl
  \tl_clear:N \l_stex_symdecl_definiens_tl

  \keys_set:nn { stex / symdecl } { #1 }
}
```

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

```
\NewDocumentCommand \symdecl { s O{} m } {
  \__stex_symdecl_args:n { #2 }
  \IfBooleanTF #1 {
    \bool_set_false:N \l_stex_symdecl_make_macro_bool
  } {
    \bool_set_true:N \l_stex_symdecl_make_macro_bool
  }
  \stex_symdecl_do:n { #3 }
  \stex_smsmode_set_codes:
}
\stex_deactivate_macro:Nn \symdecl {module~environments}
```

(*End definition for* `\symdecl`*. This function is documented on page 24.*)

`\stex_symdecl_do:n`

```
\cs_new_protected:Nn \stex_symdecl_do:n {
  \stex_if_in_module:F {
    % TODO throw error? some default namespace?
  }

  \str_if_empty:NT \l_stex_symdecl_name_str {
    \str_set:Nx \l_stex_symdecl_name_str { #1 }
  }

  \prop_if_exist:cT { g_stex_symdecl_
    \prop_item:Nn \l_stex_current_module_prop {ns} ?
    \prop_item:Nn \l_stex_current_module_prop {name} ?
      \l_stex_symdecl_name_str
    _prop
  }{
    % TODO throw error (beware of circular dependencies)
  }

  \prop_clear:N \l_tmpa_prop
  \prop_put:Nnx \l_tmpa_prop { module } {
    \prop_item:Nn \l_stex_current_module_prop {ns} ?
    \prop_item:Nn \l_stex_current_module_prop {name}
  }
```

```
\seq_clear:N \l_tmpa_seq
\prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
\prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
\prop_put:Nno \l_tmpa_prop { local } \l_stex_symdecl_local_bool
\prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl

\exp_args:No \stex_add_constant_to_current_module:n {
  \l_stex_symdecl_name_str
}

% arity/args
\int_zero:N \l_tmpb_int

\bool_set_true:N \l_tmpa_bool
\str_map_inline:Nn \l_stex_symdecl_args_str {
  \token_case_meaning:NnF ##1 {
    0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
    {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
    {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
    {\tl_to_str:n a} {
      \bool_set_false:N \l_tmpa_bool
      \int_incr:N \l_tmpb_int
    }
    {\tl_to_str:n B} {
      \bool_set_false:N \l_tmpa_bool
      \int_incr:N \l_tmpb_int
    }
  }{
    \msg_set:nnn{stex}{error/wrongargs}{
      args~value~in~symbol~declaration~for~
      \prop_item:Nn \l_stex_current_module_prop {ns} ?
      \prop_item:Nn \l_stex_current_module_prop {name} ?
      \l_stex_symdecl_name_str ~
      needs~to~be~
      i,~a,~b~or~B,~but~##1~given
    }
    \msg_error:nn{stex}{error/wrongargs}
  }
}
\bool_if:NTF \l_tmpa_bool {
  % possibly numeric
  \str_if_empty:NTF \l_stex_symdecl_args_str {
    \prop_put:Nnn \l_tmpa_prop { args } {}
    \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
  }{
    \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
    \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
    \str_clear:N \l_tmpa_str
    \int_step_inline:nn \l_tmpa_int {
      \str_put_right:Nn \l_tmpa_str i
    }
    \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
  }
} {
  \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
  \prop_put:Nnx \l_tmpa_prop { arity }
    { \str_count:N \l_stex_symdecl_args_str }
}
\prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }


% semantic macro

\bool_if:NT \l_stex_symdecl_make_macro_bool {
  \tl_set:cx { #1 } { \stex_invoke_symbol:n {
    \prop_item:Nn \l_tmpa_prop { module } ?
      \prop_item:Nn \l_tmpa_prop { name }
  } }

  \bool_if:NF \l_stex_symdecl_local_bool {
    \exp_args:Nx \stex_add_to_current_module:n {
      \tl_set:cx { #1 } { \stex_invoke_symbol:n {
        \prop_item:Nn \l_tmpa_prop { module } ?
          \prop_item:Nn \l_tmpa_prop { name }
      } }
    }
  }
}

% add to all symbols

\bool_if:NF \l_stex_symdecl_local_bool {
  \exp_args:Nx \stex_add_to_current_module:n {
    \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
      \prop_item:Nn \l_tmpa_prop { module } ?
      \prop_item:Nn \l_tmpa_prop { name }
    }
  }
}

\stex_debug:nn{symbols}{New~symbol:~
  \prop_item:Nn \l_tmpa_prop { module } ?
    \prop_item:Nn \l_tmpa_prop { name }^^J
  Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
  Args:~\prop_item:Nn \l_tmpa_prop { args }
}

% circular dependencies require this:

\prop_if_exist:cF {
  g_stex_symdecl_
  \prop_item:Nn \l_tmpa_prop { module } ?
  \prop_item:Nn \l_tmpa_prop { name }
  _prop
} {
  \prop_gset_eq:cN {
    g_stex_symdecl_
    \prop_item:Nn \l_tmpa_prop { module } ?
    \prop_item:Nn \l_tmpa_prop { name }
    _prop
  } \l_tmpa_prop
}

\stex_if_smsmode:TF {
  \bool_if:NF \l_stex_symdecl_local_bool {
    \exp_args:Nx \stex_add_to_sms:n {
      \prop_gset_from_keyval:cn {
        g_stex_symdecl_
        \prop_item:Nn \l_tmpa_prop { module } ?
        \prop_item:Nn \l_tmpa_prop { name }
        _prop
      } {
        name      = \prop_item:Nn \l_tmpa_prop { name }       ,
        module    = \prop_item:Nn \l_tmpa_prop { module }     ,
        notations = \prop_item:Nn \l_tmpa_prop { notations }  ,
        local     = \prop_item:Nn \l_tmpa_prop { local }      ,
        type      = \prop_item:Nn \l_tmpa_prop { type }       ,
        args      = \prop_item:Nn \l_tmpa_prop { args }       ,
        arity     = \prop_item:Nn \l_tmpa_prop { arity }      ,
        assocs    = \prop_item:Nn \l_tmpa_prop { assocs }
      }
      \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
        \prop_item:Nn \l_tmpa_prop { module } ?
        \prop_item:Nn \l_tmpa_prop { name }
      }
    }
  }
}{
  \exp_args:NNx \seq_put_right:Nn \l_stex_all_symbols_seq {
    \prop_item:Nn \l_tmpa_prop { module } ?
    \prop_item:Nn \l_tmpa_prop { name }
  }
  \stex_if_do_html:T {
    \stex_annotate_invisible:nnn {symdecl} {
      \prop_item:Nn \l_tmpa_prop { module } ?
      \prop_item:Nn \l_tmpa_prop { name }
    } {
      \stex_annotate_invisible:nnn{type}{}{$\l_stex_symdecl_type_tl$}
      \stex_annotate_invisible:nnn{args}{}{
        \prop_item:Nn \l_tmpa_prop { args }
      }
      \stex_annotate_invisible:nnn{macroname}{}{#1}
      \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
        \stex_annotate_invisible:nnn{definiens}{}
          {$\l_stex_symdecl_definiens_tl$}
      }
    }
  }
}
}
```

(*End definition for* `\stex_symdecl_do:n`*. This function is documented on page 25.*)

\stex_get_symbol:n

```
\str_new:N \l_stex_get_symbol_uri_str

\cs_new_protected:Nn \stex_get_symbol:n {
  \tl_if_head_eq_catcode:nNTF { #1 } \relax {
    \__stex_symdecl_get_symbol_from_cs:n { #1 }
  }{
    % argument is a string
    % is it a command name?
    \cs_if_exist:cTF { #1 }{
      \cs_set_eq:Nc \l_tmpa_tl { #1 }
      \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
      \str_if_empty:NTF \l_tmpa_str {
        \exp_args:Nx \cs_if_eq:NNTF {
          \tl_head:N \l_tmpa_tl
        } \stex_invoke_symbol:n {
          \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
        }{
          \__stex_symdecl_get_symbol_from_string:n { #1 }
        }
      } {
        \__stex_symdecl_get_symbol_from_string:n { #1 }
      }
    }{
      % argument is not a command name
      \__stex_symdecl_get_symbol_from_string:n { #1 }
      % \l_stex_all_symbols_seq
    }
  }
}

\cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
  \str_set:Nn \l_tmpa_str { #1 }
  \bool_set_false:N \l_tmpa_bool
  \stex_if_in_module:T {
    \prop_get:NnN \l_stex_current_module_prop
    { constants } \l_tmpa_seq
    \exp_args:NNo \seq_if_in:NnT \l_tmpa_seq { \l_tmpa_str } {
      \bool_set_true:N \l_tmpa_bool
      \str_set:Nx \l_stex_get_symbol_uri_str {
        \prop_item:Nn \l_stex_current_module_prop { ns } ?
        \prop_item:Nn \l_stex_current_module_prop { name } ? #1
      }
    }
  }
  \bool_if:NF \l_tmpa_bool {
    \tl_set:Nn \l_tmpa_tl {
      \msg_set:nnn{stex}{error/unknownsymbol}{
        No~symbol~#1~found!
      }
      \msg_error:nn{stex}{error/unknownsymbol}
    }
    \str_set:Nn \l_tmpa_str { #1 }
    \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
    \seq_map_inline:Nn \l_stex_all_symbols_seq {
      \str_set:Nn \l_tmpb_str { ##1 }
      \str_if_eq:eeT { \l_tmpa_str } {
        \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
      } {
        \seq_map_break:n {
          \tl_set:Nn \l_tmpa_tl {
            \str_set:Nn \l_stex_get_symbol_uri_str {
              ##1
            }
          }
        }
      }
    }
    \l_tmpa_tl
  }
}

\cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
  \exp_args:NNx \tl_set:Nn \l_tmpa_tl
    { \tl_tail:N \l_tmpa_tl }
  \tl_if_single:NTF \l_tmpa_tl {
    \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
      \exp_after:wN \str_set:Nn \exp_after:wN
        \l_stex_get_symbol_uri_str \l_tmpa_tl
    }{
      % TODO
      % tail is not a single group
    }
  }{
    % TODO
    % tail is not a single group
  }
}
```

(*End definition for* `\stex_get_symbol:n`*. This function is documented on page 25.*)

23.2 Notations

```
⟨@@=stex_notation⟩
```

`notation` arguments:

```
\keys_define:nn { stex / notation } {
  lang    .tl_set_x:N = \l__stex_notation_lang_str ,
  variant .tl_set_x:N = \l__stex_notation_variant_str ,
  prec    .str_set_x:N = \l__stex_notation_prec_str ,
  op      .tl_set:N   = \l__stex_notation_op_tl ,
  unknown .code:n     = \str_set:Nx
      \l__stex_notation_variant_str \l_keys_key_str
}

\cs_new_protected:Nn \__stex_notation_args:n {
  \str_clear:N \l__stex_notation_lang_str
  \str_clear:N \l__stex_notation_variant_str
  \str_clear:N \l__stex_notation_prec_str
  \tl_clear:N \l__stex_notation_op_tl

  \keys_set:nn { stex / notation } { #1 }
}
```

\notation

```
\NewDocumentCommand \notation { O{} m } {
  \__stex_notation_args:n { #1 }
  \tl_clear:N \l_stex_symdecl_definiens_tl
  \stex_get_symbol:n { #2 }
  \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
}
\stex_deactivate_macro:Nn \notation {module~environments}
```

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

\stex_notation_do:nn

```
\cs_new_protected:Nn \stex_notation_do:nn {
  \prop_set_eq:Nc \l_tmpa_prop {
    g_stex_symdecl_ #1 _prop
  }

  \prop_clear:N \l_tmpb_prop
  \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
  \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
  \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str

  % precedences
  \seq_clear:N \l_tmpb_seq
  \exp_args:NNno
  \str_if_empty:NTF \l__stex_notation_prec_str {
    \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
    \int_compare:nNnTF \l_tmpa_str = 0 {
      \exp_args:NNnx
      \prop_put:Nno \l_tmpb_prop { opprec }
        { \neginfprec }
    }{
      \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
    }
  } {
    \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
      \exp_args:NNnx
      \prop_put:Nno \l_tmpb_prop { opprec }
        { \neginfprec }
      \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
      \int_step_inline:nn { \l_tmpa_str } {
        \exp_args:NNx
        \seq_put_right:Nn \l_tmpb_seq { \infprec }
      }
    }{
      \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
      \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
        \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
        \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
```

```
        \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
          \l_tmpa_seq {\tl_to_str:n{x} } { \l_tmpa_str }
        \seq_map_inline:Nn \l_tmpa_seq {
          \seq_put_right:Nn \l_tmpb_seq { ##1 }
        }
      }
      \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
    }{
      \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
      \int_compare:nNnTF \l_tmpa_str = 0 {
        \exp_args:NNnx
        \prop_put:Nno \l_tmpb_prop { opprec }
          { \infprec }
      }{
        \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
      }
    }
  }
}

\seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
\int_step_inline:nn { \l_tmpa_str } {
  \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
    \exp_args:NNx
    \seq_put_right:Nn \l_tmpb_seq {
      \prop_item:Nn \l_tmpb_prop { opprec }
    }
  }
}

\prop_put:Nno \l_tmpb_prop { argprecs } \l_tmpb_seq
\tl_clear:N \l_tmpa_tl

\int_compare:nNnTF \l_tmpa_str = 0 {
  \exp_args:NNe
  \cs_set:Npn \l__stex_notation_macrocode_cs {
    \_stex_term_math_oms:nnnn { #1 }
      { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
      { \prop_item:Nn \l_tmpb_prop { opprec } }
      { \exp_not:n { #2 } }
  }
  \__stex_notation_final:
}{
  \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
  \str_if_in:NnTF \l_tmpb_str b {
    \exp_args:Nne \use:nn
    {
    \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
    \cs_set:Npn \l_tmpa_str } { {
      \_stex_term_math_omb:nnnn { #1 }
        { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
        { \prop_item:Nn \l_tmpb_prop { opprec } }
        { \exp_not:n { #2 } }
    }}
    }{
      \str_if_in:NnTF \l_tmpb_str B {
        \exp_args:Nne \use:nn
        {
        \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
        \cs_set:Npn \l_tmpa_str } { {
          \_stex_term_math_omb:nnnn { #1 }
            { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
            { \prop_item:Nn \l_tmpb_prop { opprec } }
            { \exp_not:n { #2 } }
        } }
      }{
        \exp_args:Nne \use:nn
        {
        \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
        \cs_set:Npn \l_tmpa_str } { {
          \_stex_term_math_oma:nnnn { #1 }
            { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
            { \prop_item:Nn \l_tmpb_prop { opprec } }
            { \exp_not:n { #2 } }
        } }
      }
    }

    \int_zero:N \l_tmpa_int
    \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
    \prop_get:NnN \l_tmpb_prop { argprecs } \l_tmpa_seq
    \__stex_notation_arguments:
  }
}
```

(*End definition for* `\stex_notation_do:nn`. *This function is documented on page 26.*)

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

```
\cs_new_protected:Nn \__stex_notation_arguments: {
  \int_incr:N \l_tmpa_int
  \str_if_empty:NTF \l_tmpa_str {
    \__stex_notation_final:
  }{
    \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
    \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
    \str_if_eq:VnTF \l_tmpb_str a {
      \__stex_notation_argument_assoc:n
    }{
      \str_if_eq:VnTF \l_tmpb_str B {
        \__stex_notation_argument_assoc:n
      }{
        \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
        \tl_put_right:Nx \l_tmpa_tl {
          { \_stex_term_math_arg:nnn
            { \int_use:N \l_tmpa_int }
            { \l_tmpb_str }
            { ####\int_use:N \l_tmpa_int }
          }
        }
        \__stex_notation_arguments:
      }
    }
  }
}
```

(*End definition for* `\__stex_notation_arguments:`.)

`\__stex_notation_argument_assoc:n`

```
\cs_new_protected:Nn \__stex_notation_argument_assoc:n {
  \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
  \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
  \tl_put_right:Nx \l_tmpa_tl {
    { \_stex_term_math_assoc_arg:nnnn
      { \int_use:N \l_tmpa_int }
      { \l_tmpb_str }
      \exp_args:No \exp_not:n
      {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
      { ####\int_use:N \l_tmpa_int }
    }
  }
  \__stex_notation_arguments:
}
```

(*End definition for* `\__stex_notation_argument_assoc:n`.)

`\__stex_notation_final:` Called after processing all notation arguments

```
\cs_new_protected:Nn \__stex_notation_final: {
  \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
  \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
  \prop_get:NnN \l_tmpb_prop { argprecs } \l_tmpa_seq
  \exp_args:Nne \use:nn
  {
  \cs_generate_from_arg_count:cNnn {
      stex_notation_ \l_tmpa_str \c_hash_str
      \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
      _cs
    }
    \cs_gset:Npn \l_tmpb_str } { {
      \exp_after:wN \exp_after:wN \exp_after:wN
      \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
      { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
  } }

  \tl_if_empty:NF \l__stex_notation_op_tl {
    \cs_gset:cpx {
      stex_op_notation_ \l_tmpa_str \c_hash_str
      \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
      _cs
    } {
      \_stex_term_oms:nnn {
        \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
        \l__stex_notation_lang_str
      }{
        \l_tmpa_str
      }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
    }
  }



  \stex_debug:nn{symbols}{
    Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
    ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
    Operator~precedence:~
      \prop_item:Nn \l_tmpb_prop { opprec }^^J
    Argument~precedences:~
      \seq_use:Nn \l_tmpa_seq {,~}^^J
    Notation: \cs_meaning:c {
      stex_notation_ \l_tmpa_str \c_hash_str
      \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
      _cs
    }
  }

  \prop_gset_eq:cN {
    g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
      \c_hash_str \l__stex_notation_lang_str _prop
  } \l_tmpb_prop

  \exp_args:Nx
  \stex_add_to_current_module:n {
    \prop_get:cnN {
      g_stex_symdecl_
        \prop_item:Nn \l_tmpb_prop { symbol }
      _prop
    } { notations } \exp_not:N \l_tmpa_seq
    \seq_put_right:Nn \exp_not:N \l_tmpa_seq {
      \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
    }
    \prop_put:cno {
      g_stex_symdecl_
        \prop_item:Nn \l_tmpb_prop { symbol }
      _prop
    } { notations } \exp_not:N \l_tmpa_seq
  }

  \stex_if_smsmode:TF {
    \stex_smsmode_set_codes:
    \exp_args:Nx \stex_add_to_sms:n {
      \prop_gset_from_keyval:cn {
        g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
          \c_hash_str \l__stex_notation_lang_str _prop
      } {
        symbol    = \prop_item:Nn \l_tmpb_prop { symbol }     ,
        language  = \prop_item:Nn \l_tmpb_prop { language }   ,
        variant   = \prop_item:Nn \l_tmpb_prop { variant }    ,
        opprec    = \prop_item:Nn \l_tmpb_prop { opprec }      ,
        argprecs  = \prop_item:Nn \l_tmpb_prop { argprecs }    ,
      }
    }
  }{
    \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
    \seq_put_right:Nx \l_tmpa_seq {
      \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
    }
    \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
    \prop_set_eq:cN {
      g_stex_symdecl_ \l_tmpa_str _prop
    } \l_tmpa_prop

    % HTML annotations
    \stex_if_do_html:T {
      \stex_annotate_invisible:nnn { notation }
      { \prop_item:Nn \l_tmpb_prop { symbol } } {
        \stex_annotate_invisible:nnn { notationfragment }
          { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
        \prop_get:NnN \l_tmpb_prop { argprecs } \l_tmpa_seq
        \stex_annotate_invisible:nnn { precedence }
          { \prop_item:Nn \l_tmpb_prop { opprec };
            \seq_use:Nn \l_tmpa_seq { x }
          }{}

        \int_zero:N \l_tmpa_int
        \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
        \tl_clear:N \l_tmpa_tl
        \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
          \int_incr:N \l_tmpa_int
          \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
          \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
          \str_if_eq:VnTF \l_tmpb_str a {
            \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
              \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
              \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
            } }
          }{
            \str_if_eq:VnTF \l_tmpb_str B {
              \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
                \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
                \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
              } }
            }{
              \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
                \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
              } }
            }
          }
        }
        \stex_annotate_invisible:nnn { notationcomp }{}{
          $ \exp_args:Nno \use:nn { \use:c {
            stex_notation_ \prop_item:Nn \l_tmpb_prop { symbol }
```

```
            \c_hash_str \l__stex_notation_variant_str
            \c_hash_str \l__stex_notation_lang_str _cs
          } } { \l_tmpa_tl } $
        }
      }
    }
  }
}
```

(*End definition for* `\__stex_notation_final:`.)

\symdef

```
\keys_define:nn { stex / symdef } {
  name    .str_set_x:N = \l_stex_symdecl_name_str ,
  local   .bool_set:N  = \l_stex_symdecl_local_bool ,
  args    .str_set_x:N = \l_stex_symdecl_args_str ,
  type    .tl_set:N    = \l_stex_symdecl_type_tl ,
  def     .tl_set:N    = \l_stex_symdecl_definiens_tl ,
  op      .tl_set:N    = \l__stex_notation_op_tl ,
  lang    .str_set_x:N = \l__stex_notation_lang_str ,
  variant .str_set_x:N = \l__stex_notation_variant_str ,
  prec    .str_set_x:N = \l__stex_notation_prec_str ,
  unknown .code:n      = \str_set:Nx
      \l__stex_notation_variant_str \l_keys_key_str
}

\cs_new_protected:Nn \__stex_notation_symdef_args:n {
  \str_clear:N \l_stex_symdecl_name_str
  \str_clear:N \l_stex_symdecl_args_str
  \bool_set_false:N \l_stex_symdecl_local_bool
  \tl_clear:N \l_stex_symdecl_type_tl
  \tl_clear:N \l_stex_symdecl_definiens_tl
  \str_clear:N \l__stex_notation_lang_str
  \str_clear:N \l__stex_notation_variant_str
  \str_clear:N \l__stex_notation_prec_str
  \tl_clear:N \l__stex_notation_op_tl

  \keys_set:nn { stex / symdef } { #1 }
}

\NewDocumentCommand \symdef { O{} m } {
  \__stex_notation_symdef_args:n { #1 }
  \bool_set_true:N \l_stex_symdecl_make_macro_bool
  \stex_symdecl_do:n { #2 }
  \exp_args:Nx \stex_notation_do:nn {
    \prop_item:Nn \l_tmpa_prop { module } ?
    \prop_item:Nn \l_tmpa_prop { name }
  }
}
\stex_deactivate_macro:Nn \symdef {module~environments}
```

(*End definition for* `\symdef`. *This function is documented on page 26.*)

```
⟨/package⟩
```

Chapter 24

sTeX -Terms Implementation

```
⟨*package⟩

%%%%%%%%%%%%%   terms.dtx   %%%%%%%%%%%%%

⟨@@=stex_terms⟩
```

Warnings and error messages

```
\msg_new:nnn{stex}{error/nonotation}{
  Symbol~#1~invoked,~but~has~no~notation#2!
}
\msg_new:nnn{stex}{error/notationarg}{
  Error~in~parsing~notation~#1
}
\msg_new:nnn{stex}{error/noop}{
  Symbol~#1~has~no~operator~notation~for~notation~#2
}

```

24.1 Symbol Invokations

Arguments:

```
\keys_define:nn { stex / terms } {
  lang    .tl_set_x:N = \l__stex_terms_lang_str ,
  variant .tl_set_x:N = \l__stex_terms_variant_str ,
  unknown .code:n     = \str_set:Nx
      \l__stex_terms_variant_str \l_keys_key_str
}

\cs_new_protected:Nn \__stex_terms_args:n {
  \str_clear:N \l__stex_terms_lang_str
  \str_clear:N \l__stex_terms_variant_str
  \str_clear:N \l__stex_terms_prec_str
  \tl_clear:N \l__stex_terms_op_tl

  \keys_set:nn { stex / terms } { #1 }
}
```

\stex_invoke_symbol:n Invokes a semantic macro

```
\cs_new_protected:Nn \stex_invoke_symbol:n {
  \if_mode_math:
    \exp_after:wN \__stex_terms_invoke_math:n
  \else:
    \exp_after:wN \__stex_terms_invoke_text:n
  \fi: { #1 }
}
```

(*End definition for* \stex_invoke_symbol:n*. This function is documented on page 27.*)

__stex_terms_invoke_math:n

```
\cs_new_protected:Nn \__stex_terms_invoke_math:n {
  \peek_charcode_remove:NTF ! {
    \peek_charcode:NTF [ {
      \__stex_terms_invoke_op:nw { #1 }
    }{
      \peek_charcode_remove:NTF ! {
        \peek_charcode:NTF [ {
          \__stex_terms_invoke_op_custom:nw
        }{
          % TODO throw error
        }
      }{
        \__stex_terms_invoke_op:nw { #1 } []
      }
    }
  }{
    \peek_charcode_remove:NTF * {
      \__stex_terms_invoke_text:n { #1 }
    }{
      \peek_charcode:NTF [ {
        \__stex_terms_invoke_math:nw { #1 }
      }{
        \__stex_terms_invoke_math:nw { #1 } []
      }
    }
  }
}
```

(*End definition for* __stex_terms_invoke_math:n*.*)

__stex_terms_invoke_op_custom:nw

```
\cs_new_protected:Npn \__stex_terms_invoke_op_custom:nw  #1 [#2] {
  \_stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
    \stex_highlight_term:nn{#1}{#2}
  }
}
```

(*End definition for* __stex_terms_invoke_op_custom:nw*.*)

__stex_terms_invoke_op:nw

```
\cs_new_protected:Npn \__stex_terms_invoke_op:nw  #1 [#2] {
  \__stex_terms_args:n { #2 }
  \cs_if_exist:cTF {
    stex_op_notation_ #1 \c_hash_str
    \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
  }{
    \csname stex_op_notation_ #1 \c_hash_str
      \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
    \endcsname
  }{
    \msg_error:nnxx{stex}{error/noop}{#1}{\l__stex_terms_variant_str \c_hash_str \l__stex_te
  }
}
```

(*End definition for* `\__stex_terms_invoke_op:nw`.)

__stex_terms_invoke_math:nw

```
\cs_new_protected:Npn \__stex_terms_invoke_math:nw  #1 [#2] {
  \__stex_terms_args:n { #2 }
  \prop_set_eq:Nc \l_tmpa_prop {
    g_stex_symdecl_ #1 _prop
  }
  \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
  \seq_if_empty:NTF \l_tmpa_seq {
    \msg_error:nnxn{stex}{error/nonotation}{#1}{s}
  } {
    \seq_if_in:NxTF \l_tmpa_seq
      { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
      \use:c{
        stex_notation_ #1 \c_hash_str
        \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
        _cs
      }
    }{
      \str_if_empty:NTF \l__stex_terms_variant_str {
        \str_if_empty:NTF \l__stex_terms_lang_str {
          \seq_get_left:NN \l_tmpa_seq \l_tmpa_str
          \use:c{
            stex_notation_ #1 \c_hash_str \l_tmpa_str
            _cs
          }
        }{
          \msg_error:nnxx{stex}{error/nonotation}{#1}{
            ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
          }
        }
      }{
        \msg_error:nnxx{stex}{error/nonotation}{#1}{
          ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
        }
      }
    }
  }
}
```

(*End definition for* `\__stex_terms_invoke_math:nw`*.*)

`\__stex_terms_invoke_text:n`

```
\cs_new_protected:Nn \__stex_terms_invoke_text:n {
  \peek_charcode_remove:NTF ! {
    \stex_term_custom:nn { #1 } { }
  }{
    \prop_set_eq:Nc \l_tmpa_prop {
      g_stex_symdecl_ #1 _prop
    }
    \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
    \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
  }
}
```

(*End definition for* `\__stex_terms_invoke_text:n`*.*)

24.2 Terms

Precedences:

`\infprec`
`\neginfprec`
`\l__stex_terms_downprec`

```
\tl_const:Nx \infprec {\int_use:N \c_max_int}
\tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
\int_new:N \l__stex_terms_downprec
\int_set_eq:NN \l__stex_terms_downprec \infprec
```

(*End definition for* `\infprec` *,* `\neginfprec` *, and* `\l__stex_terms_downprec`*. These variables are documented on page 28.*)

Bracketing:

`\l__stex_terms_left_bracket_str`
`\l__stex_terms_right_bracket_str`

```
\tl_set:Nn \l__stex_terms_left_bracket_str (
\tl_set:Nn \l__stex_terms_right_bracket_str )
```

(*End definition for* `\l__stex_terms_left_bracket_str` *and* `\l__stex_terms_right_bracket_str`*.*)

`\__stex_terms_maybe_brackets:nn`

Compares precedences and insert brackets accordingly

```
\cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
  \bool_if:NTF \l__stex_terms_brackets_done_bool {
    \bool_set_false:N \l__stex_terms_brackets_done_bool
    #2
  } {
    \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
      \bool_if:NTF \l_stex_inparray_bool { #2 }{
        \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
        \dobrackets { #2 }
      }
    }{ #2 }
  }
}
```

(*End definition for* `\__stex_terms_maybe_brackets:nn`*.*)

\dobrackets

```
\bool_new:N \l__stex_terms_brackets_done_bool
%\RequirePackage{scalerel}
\cs_new_protected:Npn \dobrackets #1 {
  %\ThisStyle{\if D\m@switch
  %     \exp_args:Nnx \use:nn
  %     { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
  %     { \exp_not:N\right\l__stex_terms_right_bracket_str }
  %   \else
      \exp_args:Nnx \use:nn
      {
        \bool_set_true:N \l__stex_terms_brackets_done_bool
        \int_set:Nn \l__stex_terms_downprec \infprec
        \l__stex_terms_left_bracket_str
        #1
      }
      {
        \bool_set_false:N \l__stex_terms_brackets_done_bool
        \l__stex_terms_right_bracket_str
        \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
      }
  %\fi}
}
```

(*End definition for* `\dobrackets`. *This function is documented on page 28.*)

\withbrackets

```
\cs_new_protected:Npn \withbrackets #1 #2 #3 {
  \exp_args:Nnx \use:nn
  {
    \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
    \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
    #3
  }
  {
    \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
      {\l__stex_terms_left_bracket_str}
    \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
      {\l__stex_terms_right_bracket_str}
  }
}
```

(*End definition for* `\withbrackets`. *This function is documented on page 28.*)

\STEXinvisible

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

(*End definition for* `\STEXinvisible`. *This function is documented on page 29.*)

OMDoc terms:

_stex_term_math_oms:nnnn

```
\cs_new_protected:Nn \_stex_term_oms:nnn {
  \stex_annotate:nnn{ OMID }{ #2 }{
    \stex_highlight_term:nn { #1 } { #3 }
  }
}

\cs_new_protected:Nn \_stex_term_math_oms:nnnn {
  \__stex_terms_maybe_brackets:nn { #3 }{
    \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
  }
}
```

(*End definition for* `\_stex_term_math_oms:nnnn`. *This function is documented on page 27.*)

_stex_term_math_oma:nnnn

```
\cs_new_protected:Nn \_stex_term_oma:nnn {
  \stex_annotate:nnn{ OMA }{ #2 }{
    \stex_highlight_term:nn { #1 } { #3 }
  }
}

\cs_new_protected:Nn \_stex_term_math_oma:nnnn {
  \__stex_terms_maybe_brackets:nn { #3 }{
    \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
  }
}
```

(*End definition for* `\_stex_term_math_oma:nnnn`. *This function is documented on page 27.*)

_stex_term_math_omb:nnnn

```
\cs_new_protected:Nn \_stex_term_ombind:nnn {
  \stex_annotate:nnn{ OMBIND }{ #2 }{
    \stex_highlight_term:nn { #1 } { #3 }
  }
}

\cs_new_protected:Nn \_stex_term_math_omb:nnnn {
  \__stex_terms_maybe_brackets:nn { #3 }{
    \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
  }
}
```

(*End definition for* `\_stex_term_math_omb:nnnn`. *This function is documented on page 27.*)

_stex_term_math_arg:nnn

```
\cs_new_protected:Nn \_stex_term_arg:nn {
  \stex_unhighlight_term:n {
    \stex_annotate:nnn{ arg }{ #1 }{ #2 }
  }
}
\cs_new_protected:Nn \_stex_term_math_arg:nnn {
  \exp_args:Nnx \use:nn
    { \int_set:Nn \l__stex_terms_downprec { #2 }
```

```
        \_stex_term_arg:nn { #1 }{ #3 }
    }
    { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
}
```

(*End definition for* `\_stex_term_math_arg:nnn`*. This function is documented on page 27.*)

_stex_term_math_assoc_arg:nnnn

```
\cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
  \clist_set:Nn \l_tmpa_clist{ #4 }
  \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
    \tl_set:Nn \l_tmpa_tl { #4 }
  }{
    \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
    \clist_reverse:N \l_tmpa_clist
    \clist_pop:NN \l_tmpa_clist \l_tmpa_tl

    \clist_map_inline:Nn \l_tmpa_clist {
      \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
        \exp_args:Nno
        \l_tmpa_cs { ##1 } \l_tmpa_tl
      }
    }

  }
  \exp_args:Nnno
  \_stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
}
```

(*End definition for* `\_stex_term_math_assoc_arg:nnnn`*. This function is documented on page 27.*)

\stex_term_custom:nn

```
\cs_new_protected:Nn \stex_term_custom:nn {
  \str_set:Nn \l__stex_terms_custom_uri { #1 }
  \str_set:Nn \l_tmpa_str { #2 }
  \tl_clear:N \l_tmpa_tl
  \int_zero:N \l_tmpa_int
  \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
  \__stex_terms_custom_loop:
}
```

(*End definition for* `\stex_term_custom:nn`*. This function is documented on page 29.*)

__stex_terms_custom_loop:

```
\cs_new_protected:Nn \__stex_terms_custom_loop: {
  \bool_set_false:N \l_tmpa_bool
  \bool_while_do:nn {
    \str_if_eq_p:ee X {
      \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
    }
  }{
    \int_incr:N \l_tmpa_int
  }

  \peek_charcode:NTF [ {
    % notation/text component
    \__stex_terms_custom_component:w
  } {
    \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
      % all arguments read => finish
      \__stex_terms_custom_final:
    } {
      % arguments missing
      \peek_charcode_remove:NTF * {
        % invisible, specific argument position or both
        \peek_charcode:NTF [ {
          % visible specific argument position
          \__stex_terms_custom_arg:wn
        } {
          % invisible
          \peek_charcode_remove:NTF * {
            % invisible specific argument position
            \__stex_terms_custom_arg_inv:wn
          } {
            % invisible next argument
            \__stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
          }
        }
      } {
        % next normal argument
        \__stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
      }
    }
  }
}
```

(*End definition for* `\__stex_terms_custom_loop:`.)

`\__stex_terms_custom_arg_inv:wn`

```
\cs_new_protected:Npn \__stex_terms_custom_arg_inv:wn [ #1 ] #2 {
  \bool_set_true:N \l_tmpa_bool
  \__stex_terms_custom_arg:wn [ #1 ] { #2 }
}
```

(*End definition for* `\__stex_terms_custom_arg_inv:wn`.)

`\__stex_terms_custom_arg:wn`

```
\cs_new_protected:Npn \__stex_terms_custom_arg:wn [ #1 ] #2 {
  \str_set:Nx \l_tmpb_str {
    \str_item:Nn \l_tmpa_str { #1 }
  }
  \str_case:VnTF \l_tmpb_str {
    { X } {
      \msg_error:nnx{stex}{error/notationarg}{\l__stex_terms_custom_uri}
    }
    { i } { \__stex_terms_custom_set_X:n { #1 } }
    { b } { \__stex_terms_custom_set_X:n { #1 } }
    { a } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
    { B } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
  }{}{
    \msg_error:nnx{stex}{error/notationarg}{\l__stex_terms_custom_uri}
  }

  \bool_if:nTF \l_tmpa_bool {
    \tl_put_right:Nx \l_tmpa_tl {
      \stex_annotate_invisible:n {
        \_stex_term_arg:nn { \int_eval:n { #1 } }
          \exp_not:n { { #2 } }
      }
    }
  } {
    \tl_put_right:Nx \l_tmpa_tl {
      \_stex_term_arg:nn { \int_eval:n { #1 } }
        \exp_not:n { { #2 } }
    }
  }

  \__stex_terms_custom_loop:
}
```

(*End definition for* `\__stex_terms_custom_arg:wn`.)

`\__stex_terms_custom_set_X:n`

```
  \cs_new_protected:Nn \__stex_terms_custom_set_X:n {
    \str_set:Nx \l_tmpa_str {
      \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
      X
      \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
    }
  }
```

(*End definition for* `\__stex_terms_custom_set_X:n`.)

`\__stex_terms_custom_component:`

```
  \cs_new_protected:Npn \__stex_terms_custom_component:w [ #1 ] {
    \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
    \__stex_terms_custom_loop:
  }
```

(*End definition for* `\__stex_terms_custom_component:`.)

`\__stex_terms_custom_final:`

```
  \cs_new_protected:Nn \__stex_terms_custom_final: {
    \int_compare:nNnTF \l_tmpb_int = 0 {
      \exp_args:Nnno \_stex_term_oms:nnn
    }{
      \str_if_in:NnTF \l_tmpa_str {b} {
        \exp_args:Nnno \_stex_term_ombind:nnn
      } {
        \exp_args:Nnno \_stex_term_oma:nnn
      }
    }
    { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
  }
```

(*End definition for* `\__stex_terms_custom_final:`.)

\symref
\symname

```
\NewDocumentCommand \symref { m m }{
  \let\compemph_uri_prev:\compemph@uri
  \let\compemph@uri\symrefemph@uri
  \STEXsymbol{#1}![#2]
  \let\compemph@uri\compemph_uri_prev:
}

\keys_define:nn { stex / symname } {
  post    .str_set_x:N   = \l_stex_symname_post_str
}

\cs_new_protected:Nn \stex_symname_args:n {
  \str_clear:N \l_stex_symname_post_str
  \keys_set:nn { stex / symname } { #1 }
}

\NewDocumentCommand \symname { O{} m }{
  \stex_symname_args:n { #1 }
  \stex_get_symbol:n { #2 }
  \str_set:Nx \l_tmpa_str {
    \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
  }
  \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}

  \let\compemph_uri_prev:\compemph@uri
  \let\compemph@uri\symrefemph@uri
  \exp_args:NNx \use:nn
  \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }![
    \l_tmpa_str \l_stex_symname_post_str
  ] }
  \let\compemph@uri\compemph_uri_prev:
}
```

(*End definition for* `\symref` *and* `\symname`*. These functions are documented on page 27.*)

24.3 Notation Components

```
⟨@@=stex_notationcomps⟩
```

\stex_highlight_term:nn

```

\str_new:N \l__stex_notationcomps_highlight_uri_str
\cs_new_protected:Nn \stex_highlight_term:nn {
  \exp_args:Nnx
  \use:nn {
    \str_set:Nx \l__stex_notationcomps_highlight_uri_str { #1 }
    #2
  } {
    \str_set:Nx \exp_not:N \l__stex_notationcomps_highlight_uri_str
      { \l__stex_notationcomps_highlight_uri_str }
  }
}

\cs_new_protected:Nn \stex_unhighlight_term:n {
%  \latexml_if:TF {
%    #1
%  } {
%    \rustex_if:TF {
%      #1
%    } {
       #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
%    }
%  }
}
```

(*End definition for* `\stex_highlight_term:nn`*. This function is documented on page 29.*)

\comp
\compemph@uri
\compemph
\defemph
\defemph@uri
\symrefemph
\symrefemph@uri

```
\cs_new_protected:Npn \comp #1 {
  \str_if_empty:NF \l__stex_notationcomps_highlight_uri_str {
    \rustex_if:TF {
      \stex_annotate:nnn { comp }{ \l__stex_notationcomps_highlight_uri_str }{ #1 }
    }{
      \exp_args:Nnx \compemph@uri { #1 } { \l__stex_notationcomps_highlight_uri_str }
    }
  }
}

\cs_new_protected:Npn \compemph@uri #1 #2 {
    \compemph{ #1 }
}


\cs_new_protected:Npn \compemph #1 {
    \textcolor{blue}{#1}
}

\cs_new_protected:Npn \defemph@uri #1 #2 {
    \defemph{#1}
}

\cs_new_protected:Npn \defemph #1 {
    \textbf{#1}
}

\cs_new_protected:Npn \symrefemph@uri #1 #2 {
    \symrefemph{#1}
}

\cs_new_protected:Npn \symrefemph #1 {
    \textbf{#1}
}
```

(*End definition for* `\comp` *and others. These functions are documented on page 29.*)

\ellipses

```
\NewDocumentCommand \ellipses {} { \ldots }
```

(*End definition for* `\ellipses`*. This function is documented on page 29.*)

\parray
\prmatrix
\parrayline
\parraylineh
\parraycell

```
\bool_new:N \l_stex_inparray_bool
\bool_set_false:N \l_stex_inparray_bool
\NewDocumentCommand \parray { m m } {
  \begingroup
  \bool_set_true:N \l_stex_inparray_bool
  \begin{array}{#1}
    #2
  \end{array}
  \endgroup
}

\NewDocumentCommand \prmatrix { m } {
  \begingroup
  \bool_set_true:N \l_stex_inparray_bool
  \begin{matrix}
    #1
  \end{matrix}
  \endgroup
}

\def \maybephline {
  \bool_if:NT \l_stex_inparray_bool {\hline}
}

\def \parrayline #1 #2 {
  #1 #2 \bool_if:NT \l_stex_inparray_bool {\\}
}

\def \parraylineh #1 #2 {
  #1 #2 \bool_if:NT \l_stex_inparray_bool {\\\hline}
}

\def \parraycell #1 {
  #1 \bool_if:NT \l_stex_inparray_bool {&}
}
```

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

```
⟨/package⟩
```

Chapter 25

sTeX -Structural Features Implementation

```
⟨*package⟩

%%%%%%%%%%%%%   features.dtx   %%%%%%%%%%%%%

⟨@@=stex_features⟩
```

Warnings and error messages

25.1 The feature environment

structural@feature

```

\NewDocumentEnvironment{structural@feature}{ m m m }{
  \stex_if_in_module:F {
    \msg_set:nnn{stex}{error/nomodule}{
      Structural~Feature~has~to~occur~in~a~module:\\
      Feature~#2~of~type~#1\\
      In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
    }
    \msg_error:nn{stex}{error/nomodule}
  }

  \str_set:Nx \l_stex_module_name_str {
    \prop_item:Nn \l_stex_current_module_prop
      { name } / #2 - feature
  }

  \str_set:Nx \l_stex_module_ns_str {
    \prop_item:Nn \l_stex_current_module_prop
      { ns }
  }

  \str_clear:N \l_tmpa_str
  \seq_clear:N \l_tmpa_seq
  \tl_clear:N \l_tmpa_tl
  \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
    origname  = #2,
    name      = \l_stex_module_name_str ,
    ns        = \l_stex_module_ns_str ,
    imports   = \exp_not:o { \l_tmpa_seq } ,
    constants = \exp_not:o { \l_tmpa_seq } ,
    content   = \exp_not:o { \l_tmpa_tl }  ,
    file      = \exp_not:o { \g_stex_currentfile_seq } ,
    lang      = \l_stex_module_lang_str ,
    sig       = \l_tmpa_str ,
    meta      = \l_tmpa_str ,
    feature   = #1 ,
  }

  \stex_if_smsmode:TF {
    \stex_smsmode_set_codes:
  } {
    \begin{stex_annotate_env}{ feature:#1 }{}
      \stex_annotate_invisible:nnn{header}{}{ #3 }
  }
}{
  \str_set:Nx \l_tmpa_str {
    c_stex_feature_
    \prop_item:Nn \l_stex_current_module_prop { ns } ?
    \prop_item:Nn \l_stex_current_module_prop { name }
    _prop
  }
  \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
  \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
  \stex_if_smsmode:TF {
    \exp_args:Nx \stex_add_to_sms:n {
      \prop_gset_from_keyval:cn {
        c_stex_feature_
        \prop_item:Nn \l_stex_current_module_prop { ns } ?
        \prop_item:Nn \l_stex_current_module_prop { name }
        _prop
      } {
        origname  = #2,
        name      = \prop_item:cn { \l_tmpa_str } { name } ,
        ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
        imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
        constants = \prop_item:cn { \l_tmpa_str } { constants } ,
        content   = \prop_item:cn { \l_tmpa_str } { content } ,
        file      = \prop_item:cn { \l_tmpa_str } { file } ,
        lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
        sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
        meta      = \prop_item:cn { \l_tmpa_str } { meta } ,
        feature   = \prop_item:cn { \l_tmpa_str } { feature }
      }
    }
  } {
      \end{stex_annotate_env}
  }
}

```

25.2 Features

structure

```

\prop_new:N \l_stex_all_structures_prop

\keys_define:nn { stex / features / structure } {
  name          .str_set_x:N  = \l__stex_features_structure_name_str ,
}

\cs_new_protected:Nn \__stex_features_structure_args:n {
  \str_clear:N \l__stex_features_structure_name_str
  \keys_set:nn { stex / features / structure } { #1 }
}

%\stex_new_feature:nnnn { structure } { O{} m } {
%  \__stex_features_structure_args:n { ##1 }
%  \str_if_empty:NT \l__stex_features_structure_name_str {
%    \str_set:Nx \l__stex_features_structure_name_str { ##2 }
%  }
%} {
%
%}

\NewDocumentEnvironment{mathstructure}{ O{} m }{
  \__stex_features_structure_args:n { #1 }
  \str_if_empty:NT \l__stex_features_structure_name_str {
    \str_set:Nx \l__stex_features_structure_name_str { #2 }
  }
  \exp_args:Nnnx
  \begin{structural@feature}{ structure }
    { \l__stex_features_structure_name_str }{}
    \seq_clear:N \l_tmpa_seq
    \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq

}{
    \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
    \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
    \str_set:Nx \l_tmpa_str {
      \prop_item:Nn \l_stex_current_module_prop { ns } ?
      \prop_item:Nn \l_stex_current_module_prop { name }
    }
    \seq_map_inline:Nn \l_tmpa_seq {
      \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
    }
    \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
    \exp_args:Nnx
    \AddToHookNext { env / mathstructure / after }{
      \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}{
        \_stex_term_math_oms:nnnn { \l_tmpa_str }{}{0}{}
      }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
      \STEXexport {
        \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
          {\prop_item:Nn \l_stex_current_module_prop { origname }}
          {\l_tmpa_str}
          \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
            {#2}{\l_tmpa_str}
%        \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
%          \prop_item:Nn \l_stex_current_module_prop { origname },
%          \l_tmpa_str
%        }
%        \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
%          #2,\l_tmpa_str
%        }
%        \tl_set:cx { #2 } {
%          \stex_invoke_structure:n { \l_tmpa_str }
      }
    }

  \end{structural@feature}
  % \g_stex_last_feature_prop
}
```

\instantiate

```
\seq_new:N \l__stex_features_structure_field_seq
\str_new:N \l__stex_features_structure_field_str
\str_new:N \l__stex_features_structure_def_tl
\prop_new:N \l__stex_features_structure_prop
\NewDocumentCommand \instantiate { m O{} m }{
  \stex_smsmode_set_codes:
  \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
  \prop_set_eq:Nc \l__stex_features_structure_prop {
    c_stex_feature_\l_tmpa_str _prop
  }
  \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
  \seq_map_inline:Nn \l__stex_features_structure_field_seq {
    \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
    \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
      \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
      \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
        {!} \l_tmpa_tl
      \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
        \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
        \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
        \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
      }{
        \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
        \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
        \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
          \l_tmpa_tl
        \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
```

```
        \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
        \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
      }{
        \tl_clear:N \l_tmpb_tl
      }
    }
  }{
    \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
    \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
      \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
      \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
      \tl_clear:N \l_tmpa_tl
    }{
      % TODO throw error
    }
  }
  % \l_tmpa_str: name
  % \l_tmpa_tl: definiens
  % \l_tmpb_tl: notation
  \tl_if_empty:NT \l__stex_features_structure_field_str {
    % TODO throw error
  }
  \str_clear:N \l_tmpb_str

  \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
  \seq_map_inline:Nn \l_tmpa_seq {
    \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
    \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
    \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
      \seq_map_break:n {
        \str_set:Nn \l_tmpb_str { ####1 }
      }
    }
  }
  \prop_get:cnN { g_stex_symdecl_ \l_tmpb_str _prop } {args}
    \l_tmpb_str

  \tl_if_empty:NTF \l_tmpb_tl {
    \tl_if_empty:NF \l_tmpa_tl {
      \exp_args:Nx \use:n {
        \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fe
      }
    }
  }{
    \tl_if_empty:NTF \l_tmpa_tl {
      \exp_args:Nx \use:n {
        \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\e
      }

    }{
      \exp_args:Nx \use:n {
        \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
        \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
      }
      }
    }
%     \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
%     \prop_item:Nn \l_stex_current_module_prop {name} ?
%     #3/\l__stex_features_structure_field_str
%     \par
%     \expandafter\present\csname
%       g_stex_symdecl_
%       \prop_item:Nn \l_stex_current_module_prop {ns} ?
%       \prop_item:Nn \l_stex_current_module_prop {name} ?
%       #3/\l__stex_features_structure_field_str
%       _prop
%     \endcsname
  }

  \tl_clear:N \l__stex_features_structure_def_tl

  \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
  \seq_map_inline:Nn \l_tmpa_seq {
    \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
    \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
    \exp_args:Nx \use:n {
      \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {

      }
    }

    \prop_if_exist:cF {
      g_stex_symdecl_
      \prop_item:Nn \l_stex_current_module_prop {ns} ?
      \prop_item:Nn \l_stex_current_module_prop {name} ?
      #3/\l_tmpa_str
      _prop
    }{
      \prop_get:cnN { g_stex_symdecl_ ##1 _prop } {args}
        \l_tmpb_str
      \exp_args:Nx \use:n {
        \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
      }
    }
  }

  \symdecl*[type={\STEXsymbol{module-type}{
    \_stex_term_math_oms:nnnn {
      \prop_item:Nn \l__stex_features_structure_prop {ns} ?
      \prop_item:Nn \l__stex_features_structure_prop {name}
      }{}{0}{}
  }}]{#3}

  % TODO: -> sms file

  \tl_set:cx{ #3 }{
    \stex_invoke_structure:nnn {
      \prop_item:Nn \l_stex_current_module_prop {ns} ?
      \prop_item:Nn \l_stex_current_module_prop {name} ? #3
    } {
      \prop_item:Nn \l__stex_features_structure_prop {ns} ?
      \prop_item:Nn \l__stex_features_structure_prop {name}
    }
  }

}
```

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

`\stex_invoke_structure:nnn`

```
% #1: URI of the instance
% #2: URI of the instantiated module
\cs_new_protected:Nn \stex_invoke_structure:nnn {
  \tl_if_empty:nTF{ #3 }{
    \prop_set_eq:Nc \l__stex_features_structure_prop {
      c_stex_feature_ #2 _prop
    }
    \tl_clear:N \l_tmpa_tl
    \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
    \seq_map_inline:Nn \l_tmpa_seq {
      \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
      \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
      \cs_if_exist:cT {
        stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
      }{
        \tl_if_empty:NF \l_tmpa_tl {
          \tl_put_right:Nn \l_tmpa_tl {,}
        }
        \tl_put_right:Nx \l_tmpa_tl {
          \stex_invoke_symbol:n {#1/\l_tmpa_str}!
        }
      }
    }
    \exp_args:No \mathstruct \l_tmpa_tl
  }{
    \stex_invoke_symbol:n{#1/#3}
  }
}
```

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

⟨/package⟩

Chapter 26

sTeX -Statements Implementation

```
⟨*package⟩

%%%%%%%%%%%%%   features.dtx   %%%%%%%%%%%%%

\protected\def\ignorespacesandpars{
  \begingroup\catcode13=10\relax
  \@ifnextchar\par{
    \endgroup\expandafter\ignorespacesandpars\@gobble
  }{
    \endgroup
  }
}

⟨@@=stex_statements⟩
```

Warnings and error messages

```

\def\titleemph#1{\textbf{#1}}
```

symboldoc

```
\NewDocumentEnvironment{symboldoc}{ m }{
  \seq_set_split:Nnn \l_tmpa_seq , { #1 }
  \seq_clear:N \l_tmpb_seq
  \seq_map_inline:Nn \l_tmpa_seq {
    \str_if_eq:nnF{ ##1 }{}{
      \stex_get_symbol:n { ##1 }
      \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
        \l_stex_get_symbol_uri_str
      }
    }
  }
  \par
  \exp_args:Nnnx
  \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
}{
  \end{stex_annotate_env}
}
\seq_new:N \g_stex_statements_patched_seq

\cs_new_protected:Nn \stex_statements_set_patched:n {
  \seq_put_right:Nn \g_stex_statements_patched_seq {#1}
}

\cs_new_protected:Nn \stex_statements_patch:nn {
  \seq_if_in:NnF \g_stex_statements_patched_seq {#1} {
    \AddToHook{begindocument}{
      \cs_if_exist:cTF{end#1}{
        \AddToHook{env/#1/before}[stex]{\use:c{__stex_statements_#2_begin:n}{}}
        \AddToHook{env/#1/after}[stex]{\use:c{__stex_statements_#2_end:}}
      }{
        \NewDocumentEnvironment{#1}{O{}}{
          \use:c{__stex_statements_#2_begin:n}{}
        }{
          \use:c{__stex_statements_#2_end:}
        }
      }
    }
  }
}
```

26.1 Definitions

definition

```
\keys_define:nn {stex / definiendum }{
  post    .tl_set:N     = \l__stex_statements_definiendum_post_tl,
  root    .str_set_x:N  = \l__stex_statements_definiendum_root_str
}
\cs_new_protected:Nn \__stex_statements_definiendum_args:n {
  \str_clear:N \l__stex_statements_definiendum_root_str
  \tl_clear:N \l__stex_statements_definiendum_post_tl
  \keys_set:nn { stex / definiendum }{ #1 }
}
\NewDocumentCommand \definiendum { O{} m m} {
  \__stex_statements_definiendum_args:n { #1 }
  \stex_get_symbol:n { #2 }
  \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
  \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
    \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
      \tl_set:Nn \l_tmpa_tl { #3 }
    } {
      \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
      \tl_set:Nn \l_tmpa_tl {
        \l__stex_statements_definiendum_root_str\l__stex_statements_definiendum_post_tl
       }
    }
  } {
    \tl_set:Nn \l_tmpa_tl { #3 }
  }

  % TODO root
  \rustex_if:TF {
    \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
  } {
    \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
  }
}
\stex_deactivate_macro:Nn \definiendum {definition~environments}

\NewDocumentCommand \definame { O{} m } {
  \__stex_statements_definiendum_args:n { #1 }
  % TODO: root
  \stex_get_symbol:n { #2 }
  \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
  \str_set:Nx \l_tmpa_str {
    \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
  }
  \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
  \rustex_if:TF {
    \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
      \l_tmpa_str\l__stex_statements_definiendum_post_tl
      }
  } {
    \defemph@uri {
      \l_tmpa_str\l__stex_statements_definiendum_post_tl
    } { \l_stex_get_symbol_uri_str }
  }
}
\stex_deactivate_macro:Nn \definame {definition~environments}

\cs_new_protected:Nn \__stex_statements_defi_begin:n {
  \stex_reactivate_macro:N \definiendum
  \stex_reactivate_macro:N \definame
  \seq_set_split:Nnn \l_tmpa_seq , { #1 }
  \seq_clear:N \l_tmpb_seq
  \seq_map_inline:Nn \l_tmpa_seq {
    \str_if_eq:nnF{ ##1 }{}{
      \stex_get_symbol:n { ##1 }
      \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
        \l_stex_get_symbol_uri_str
      }
    }
  }
  \stex_smsmode_set_codes:
  \exp_args:Nnnx
  \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
}

\cs_new_protected:Nn \__stex_statements_defi_end: {
  \end{stex_annotate_env}
}
```

Hook:

```
\stex_statements_patch:nn{definition}{defi}
```

inline:

```
\NewDocumentCommand \inlinedef { m } {
  \begingroup
  \stex_reactivate_macro:N \definiendum
  \stex_reactivate_macro:N \definame
  \stex_ref_new_doc_target:n{}
  #1
  \endgroup
}
```

26.2 Assertions

assertion

```
\cs_new_protected:Nn \__stex_statements_assertion_begin:n {
  \seq_set_split:Nnn \l_tmpa_seq , { #1 }
  \seq_clear:N \l_tmpb_seq
  \seq_map_inline:Nn \l_tmpa_seq {
    \str_if_eq:nnF{ ##1 }{}{
      \stex_get_symbol:n { ##1 }
      \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
        \l_stex_get_symbol_uri_str
      }
    }
  }
  \titleemph{Assertion}~
  \stex_smsmode_set_codes:
  \exp_args:Nnnx
  \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
}

\cs_new_protected:Nn \__stex_statements_assertion_end: {
  \end{stex_annotate_env}
}
```

Hook:

```
\stex_statements_patch:nn{assertion}{assertion}
```

inline:

```
\NewDocumentCommand \inlineass { m } {
  \begingroup
  \stex_ref_new_doc_target:n{}
  #1
  \endgroup
}
```

theorem

```
\cs_new_protected:Nn \__stex_statements_theorem_begin:n {
  \seq_set_split:Nnn \l_tmpa_seq , { #1 }
  \seq_clear:N \l_tmpb_seq
  \seq_map_inline:Nn \l_tmpa_seq {
    \str_if_eq:nnF{ ##1 }{}{
      \stex_get_symbol:n { ##1 }
      \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
        \l_stex_get_symbol_uri_str
      }
    }
  }
  \titleemph{Theorem}~
  \stex_smsmode_set_codes:
  \exp_args:Nnnx
  \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
}

\cs_new_protected:Nn \__stex_statements_theorem_end: {
  \end{stex_annotate_env}
}
```

Hook:

```
\stex_statements_patch:nn{theorem}{theorem}
```

`lemma`

```
\cs_new_protected:Nn \__stex_statements_lemma_begin:n {
  \seq_set_split:Nnn \l_tmpa_seq , { #1 }
  \seq_clear:N \l_tmpb_seq
  \seq_map_inline:Nn \l_tmpa_seq {
\str_if_eq:nnF{ ##1 }{}{
      \stex_get_symbol:n { ##1 }
      \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
        \l_stex_get_symbol_uri_str
      }
    }
  }
  \titleemph{Lemma}~
  \stex_smsmode_set_codes:
  \exp_args:Nnnx
  \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
}

\cs_new_protected:Nn \__stex_statements_lemma_end: {
  \end{stex_annotate_env}
}
```

Hook:

```
\stex_statements_patch:nn{lemma}{lemma}
```

`axiom`

```
\cs_new_protected:Nn \__stex_statements_axiom_begin:n {
  \seq_set_split:Nnn \l_tmpa_seq , { #1 }
  \seq_clear:N \l_tmpb_seq
  \seq_map_inline:Nn \l_tmpa_seq {
    \str_if_eq:nnF{ ##1 }{}{
      \stex_get_symbol:n { ##1 }
      \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
        \l_stex_get_symbol_uri_str
      }
    }
  }
  \titleemph{Axiom}~
  \stex_smsmode_set_codes:
  \exp_args:Nnnx
  \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
}

\cs_new_protected:Nn \__stex_statements_axiom_end: {
  \end{stex_annotate_env}
}
```

Hook:

```
\stex_statements_patch:nn{axiom}{axiom}
```

26.3 Examples

example

```
\cs_new_protected:Nn \__stex_statements_example_begin:n {
  \seq_set_split:Nnn \l_tmpa_seq , { #1 }
  \seq_clear:N \l_tmpb_seq
  \seq_map_inline:Nn \l_tmpa_seq {
   \str_if_eq:nnF{ ##1 }{}{
      \stex_get_symbol:n { ##1 }
      \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
        \l_stex_get_symbol_uri_str
      }
    }
  }
  \titleemph{Example}~
  \stex_smsmode_set_codes:
  \exp_args:Nnnx
  \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpb_seq {,}}
}

\cs_new_protected:Nn \__stex_statements_example_end: {
  \end{stex_annotate_env}
}
```

Hook:

```
\stex_statements_patch:nn{example}{example}
```

inline:

```
\NewDocumentCommand \inlineex { m } {
  \begingroup
  \stex_ref_new_doc_target:n{}
  #1
  \endgroup
}
```

26.4 OMText

```
\keys_define:nn { stex / omtext} {
  id      .str_set_x:N   = \l_stex_omtext_id_str ,
  title   .tl_set:N   = \l_stex_omtext_title_tl ,
  type    .tl_set_x:N   = \l_stex_omtext_type_tl ,
  for     .tl_set_x:N   = \l_stex_omtext_for_tl ,
  from    .tl_set_x:N   = \l_stex_omtext_from_tl ,
  start   .tl_set:N   = \l_stex_omtext_start_tl ,
}
\cs_new_protected:Nn \stex_omtext_args:n {
  \tl_clear:N \l_stex_omtext_title_tl
  \tl_clear:N \l_stex_omtext_start_tl
  \keys_set:nn { stex / omtext }{ #1 }
}
\newif\if@in@omtext\@in@omtextfalse
\NewDocumentEnvironment {omtext} { O{} } {
  \stex_omtext_args:n { #1 }
  \tl_if_empty:NTF \l_stex_omtext_start_tl {
    \tl_if_empty:NF \l_stex_omtext_title_tl {
      \titleemph{\l_stex_omtext_title_tl}:~
    }
  }{
    \titleemph{\l_stex_omtext_start_tl}~
  }
  \@in@omtexttrue

  \stex_ref_new_doc_target:n \l_stex_omtext_id_str
  \stex_smsmode_set_codes:
  \ignorespacesandpars
}{}
```

⟨/package⟩

Chapter 27

The Implementation

27.1 Package Options

We declare some switches which will modify the behavior according to the package options. Generally, an option xxx will just set the appropriate switches to true (otherwise they stay false).[10] EdN:10

```
⟨*package⟩
⟨@@=stex_sproof⟩

%%%%%%%%%%%%%   sproof.dtx   %%%%%%%%%%%%%

```

27.2 Proofs

We first define some keys for the proof environment.

```
\keys_define:nn { stex / spf } {
  id          .str_set_x:N  = \l__stex_sproof_spf_id_str,
  display     .tl_set:N     = \l__stex_sproof_spf_display_tl,
  for         .tl_set:N     = \l__stex_sproof_spf_for_tl ,
  from        .tl_set:N     = \l__stex_sproof_spf_from_tl ,
  proofend    .tl_set:N     = \l__stex_sproof_spf_proofend_tl,
  type        .tl_set:N     = \l__stex_sproof_spf_type_tl,
  title       .tl_set:N     = \l__stex_sproof_spf_title_tl,
  continues   .tl_set:N     = \l__stex_sproof_spf_continues_tl,
  functions   .tl_set:N     = \l__stex_sproof_spf_functions_tl,
  method      .tl_set:N     = \l__stex_sproof_spf_method_tl
}
\cs_new_protected:Nn \__stex_sproof_spf_args:n {
\str_clear:N \l__stex_sproof_spf_id_str
\tl_clear:N \l__stex_sproof_spf_display_tl
\tl_clear:N \l__stex_sproof_spf_for_tl
\tl_clear:N \l__stex_sproof_spf_from_tl
\tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
\tl_clear:N \l__stex_sproof_spf_type_tl
\tl_clear:N \l__stex_sproof_spf_title_tl
```

[10] EdNote: need an implementation for LaTeXML

```
\tl_clear:N \l__stex_sproof_spf_continues_tl
\tl_clear:N \l__stex_sproof_spf_functions_tl
\tl_clear:N \l__stex_sproof_spf_method_tl
\keys_set:nn { stex / spf }{ #1 }
}
```

`\spf@flow` We define this macro, so that we can test whether the `display` key has the value `flow`

```
\def\spf@flow{flow}
```

(*End definition for* `\spf@flow`*. This function is documented on page* **??**.)

For proofs, we will have to have deeply nested structures of enumerated list-like environments. However, LaTeX only allows `enumerate` environments up to nesting depth 4 and general list environments up to listing depth 6. This is not enough for us. Therefore we have decided to go along the route proposed by Leslie Lamport to use a single top-level list with dotted sequences of numbers to identify the position in the proof tree. Unfortunately, we could not use his `pf.sty` package directly, since it does not do automatic numbering, and we have to add keyword arguments all over the place, to accomodate semantic information.

`pst@with@label` This environment manages[6] the path labeling of the proof steps in the description environment of the outermost `proof` environment. The argument is the label prefix up to now; which we cache in `\pst@label` (we need evaluate it first, since are in the right place now!). Then we increment the proof depth which is stored in `\count10` (lower counters are used by TeX for page numbering) and initialize the next level counter `\count\count10` with 1. In the end call for this environment, we just decrease the proof depth counter by 1 again.

```
\newcount\count_ten
\newenvironment{pst@with@label}[1]{
  \edef\pst@label{#1}
  \advance\count_ten by 1\relax
  \count_ten=1
}{
  \advance\count_ten by -1\relax
}
```

`\the@pst@label` `\the@pst@label` evaluates to the current step label.

```
\def\the@pst@label{
  \pst@make@label\pst@label{\number\count_ten}\l__stex_sproof_pstlabel_postfix_tl
}
```

(*End definition for* `\the@pst@label`*. This function is documented on page* **??**.)

`\setpstlabelstyle` `\setpstlabelstyle`{metaKey-Val pairs} makes the labeling style customizable. `\setpstlabelstyle{pr`
will change the labeling style from **P.1.2.3** to **Pr-1-2-3†**. `\setpstlabelstyledefault` will set the labeling style back to default.

```
\keys_define:nn { stex / pstlabel }{
  prefix      .tl_set:N   = \l__stex_sproof_pstlabel_prefix_tl,
  delimiter   .tl_set:N   = \l__stex_sproof_pstlabel_delimiter_tl,
  postfix     .tl_set:N   = \l__stex_sproof_pstlabel_postfix_tl
}
\cs_new_protected:Nn \__stex_sproof_pstlabel_args:n {
```

[6]This gets the labeling right but only works 8 levels deep

```
  \tl_set:Nn \l__stex_sproof_pstlabel_prefix_tl {P}
  \tl_set:Nn \l__stex_sproof_pstlabel_delimiter_tl {.}
  \tl_clear:N \l__stex_sproof_pstlabel_postfix_tl
}
\__stex_sproof_pstlabel_args:n {}
\newcommand\setpstlabelstyle[1]{
  \__stex_sproof_pstlabel_args:n {#1}
}
\newcommand\setpstlabelstyledefault{%
  \__stex_sproof_pstlabel_args:n{prefix=P,delimiter=.,postfix={}}
}
```

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

`\pstlabelstyle` `\pstlabelstyle` just sets the `\pst@make@label` macro according to the style.

```
\ExplSyntaxOff
\def\pst@make@label@long#1#2{\@for\@I:=#1\do{\expandafter\expandafter\expandafter\@I\csname
\def\pst@make@label@angles#1#2{\ensuremath{\@for\@I:=#1\do{\rangle}}#2}
\def\pst@make@label@short#1#2{#2}
\def\pst@make@label@empty#1#2{}
\ExplSyntaxOn
\def\pstlabelstyle#1{%
  \def\pst@make@label{\use:c{pst@make@label@#1}}%
}%
\pstlabelstyle{long}%
```

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

`\next@pst@label` `\next@pst@label` increments the step label at the current level.

```
\def\next@pst@label{%
  \global\advance\count\count10 by 1%
}%
```

(*End definition for* `\next@pst@label`*. This function is documented on page* **??**.)

`\sproofend` This macro places a little box at the end of the line if there is space, or at the end of the next line if there isn't

```
\def\sproof@box{
  \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
}
\def\spf@proofend{\sproof@box}
\def\sproofend{
  \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
    \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
  }
}
\def\sProofEndSymbol#1{\def\sproof@box{#1}}
```

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

`spf@*@kw`

```
\def\spf@proofsketch@kw{Proof Sketch}
\def\spf@proof@kw{Proof}
\def\spf@step@kw{Step}
```

(*End definition for* `spf@*@kw`*. This function is documented on page* **??**.)

For the other languages, we set up triggers

```
\cs_if_exist:NT \bbl@loaded {
  \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
  \clist_if_in:NnT \l_tmpa_clist {ngerman}{
    \input{sproof-ngerman.ldf}
  }
  \clist_if_in:NnT \l_tmpa_clist {finnish}{
    \input{sproof-finnish.ldf}
  }
  \clist_if_in:NnT \l_tmpa_clist {french}{
    \input{sproof-french.ldf}
  }
  \clist_if_in:NnT \l_tmpa_clist {russian}{
    \input{sproof-russian.ldf}
  }
}

```

`spfsketch`

```
\newcommand\spfsketch[2][]{
  \__stex_sproof_spf_args:n{#1}
  \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
    \titleemph{
      \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {
        \spf@proofsketch@kw
      }{
        \l__stex_sproof_spf_type_tl
      }
    }:
  }
  {~#2}
  %\sref@label@id{this \ifx\spf@type\@empty\spf@proofsketch@kw\else\spf@type\fi}
  \sproofend
}
```

(*End definition for* `spfsketch`*. This function is documented on page* **??**.)

EdN:11
EdN:12

`spfeq` This is very similar to `\spfsketch`, but uses a computation array[11][12]

```
\newenvironment{spfeq}[2][]{
  \__stex_sproof_spf_args:n{#1}
  %\sref@target
  \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
    \titleemph{
      \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {
        \spf@proof@kw
      }{
        \l__stex_sproof_spf_type_tl
      }
    }:
```

[11] EdNote: This should really be more like a tabular with an ensuremath in it. or invoke text on the last column

[12] EdNote: document above

```
  }
  {~#2}
  \begin{displaymath}\begin{array}{rcll}
}{
  \end{array}\end{displaymath}
}
```

(*End definition for* `spfeq`. *This function is documented on page* **??**.)

`sproof` In this environment, we initialize the proof depth counter `\count10` to 10, and set up the description environment that will take the proof steps. At the end of the proof, we position the proof end into the last line.

```
\newenvironment{spf@proof}[2][]{
  \__stex_sproof_spf_args:n{#1}
  %\sref@target
  \count_ten=10
  \par\noindent
  \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
    \titleemph{
      \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {
        \spf@proof@kw
      }{
        \l__stex_sproof_spf_type_tl
      }
    }:
  }
  {~#2}
  %\sref@label@id{this \ifx\spf@type\@empty\spf@proof@kw\else\spf@type\fi}
  \def\pst@label{}
  \newcount\pst@count% initialize the labeling mechanism
  \begin{description}\begin{pst@with@label}{\l__stex_sproof_pstlabel_prefix_tl}
}{
  \end{pst@with@label}\end{description}
}
\newenvironment{sproof}[2][]{\begin{spf@proof}[#1]{#2}}{\sproofend\end{spf@proof}}
\newenvironment{sProof}[2][]{\begin{spf@proof}[#1]{#2}}{\end{spf@proof}}
```

`\spfidea`

```
\newcommand\spfidea[2][]{
  \__stex_sproof_spf_args:n{#1}
  \titleemph{
    \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {Proof~Idea}{
      \l__stex_sproof_spf_type_tl
    }:
  }~#2
  \sproofend
}
```

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

The next two environments (proof steps) and comments, are mostly semantical, they take `KeyVal` arguments that specify their semantic role. In draft mode, they read these values and show them. If the surrounding proof had `display=flow`, then no new `\item` is generated, otherwise it is. In any case, the proof step number (at the current level) is incremented.

EdN:13 spfstep [13]

```
\newenvironment{spfstep}[1][]{
  \__stex_sproof_spf_args:n{#1}
  \@in@omtexttrue
  \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
    \item[\the@pst@label]
  }
  \tl_if_empty:NF \l__stex_sproof_spf_title_tl {
    {(\titleemph{\l__stex_sproof_spf_title_tl})\enspace}
  }
  %\sref@label@id{\pst@label}
  \ignorespacesandpars
}{
  \next@pst@label\ignorespacesandpars
}
```

sproofcomment

```
\newenvironment{sproofcomment}[1][]{
  \__stex_sproof_spf_args:n{#1}
  \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
    \item[\the@pst@label]
  }
}{
  \next@pst@label
}
```

The next two environments also take a `KeyVal` argument, but also a regular one, which contains a start text. Both environments start a new numbered proof level.

subproof In the `subproof` environment, a new (lower-level) proproofof environment is started.

```
\newenvironment{subproof}[2][]{
  \__stex_sproof_spf_args:n{#1}
  \def\@test{#2}
  \ifx\@test\empty\else
    \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
      \item[\the@pst@label]
    }{#2}
  \fi
  \begin{pst@with@label}{\pst@label,\number\count_ten}
}{
  \end{pst@with@label}\next@pst@label
}
```

spfcases In the `pfcases` environment, the start text is displayed as the first comment of the proof.

```
\newenvironment{spfcases}[2][]{
  \def\@test{#1}
  \ifx\@test\empty
    \begin{subproof}[method=by-cases]{#2}
  \else
    \begin{subproof}[#1,method=by-cases]{#2}
  \fi
}{
```

[13] EdNote: MK: labeling of steps does not work yet.

```
  \end{subproof}
}
```

spfcase In the pfcase environment, the start text is displayed specification of the case after the \item

```
\newenvironment{spfcase}[2][]{
  \__stex_sproof_spf_args:n{#1}
  \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
    \item[\the@pst@label]
  }
  \def\@test{#2}
  \ifx\@test\@empty
  \else
    {\titleemph{#2}:~}
  \fi
  \begin{pst@with@label}{\pst@label,\number\count_ten}
}{
  \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
    \sproofend
  }
  \end{pst@with@label}
  \next@pst@label
}
```

spfcase similar to spfcase, takes a third argument.

```
\newcommand\spfcasesketch[3][]{
  \__stex_sproof_spf_args:n{#1}
  \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
    \item[\the@pst@label]
  }
  \def\@test{#2}
  \ifx\@test\@empty
  \else
    {\titleemph{#2}:~}
  \fi#3
  \next@pst@label
}%
```

27.3 Justifications

We define the actions that are undertaken, when the keys for justifications are encountered. Here this is very simple, we just define an internal macro with the value, so that we can use it later.

```
\keys_define:nn { stex / just }{
  id        .str_set_x:N  = \l__stex_sproof_just_id_str,
  method    .tl_set:N     = \l__stex_sproof_just_method_tl,
  premises  .tl_set:N     = \l__stex_sproof_just_premises_tl,
  args      .tl_set:N     = \l__stex_sproof_just_args_tl
}
```

EdN:14 The next three environments and macros are purely semantic, so we ignore the keyval arguments for now and only display the content.[14]

[14]EdNote: need to do something about the premise in draft mode.

`justification`

```
\newenvironment{justification}[1][]{}{}
```

`\premise`

```
\newcommand\premise[2][]{#2}
```

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

`\justarg` the `\justarg` macro is purely semantic, so we ignore the keyval arguments for now and only display the content.

```
\newcommand\justarg[2][]{#2}
⟨/package⟩
```

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

Some auxiliary code, and clean up to be executed at the end of the package.

Chapter 28

sTeX -Others Implementation

```
⟨*package⟩

%%%%%%%%%%%%%   others.dtx   %%%%%%%%%%%%%

⟨@@=stex_others⟩
```

Warnings and error messages

```
  % None
```

\MSC Math subject classifier

```
\NewDocumentCommand \MSC {m} {
  % TODO
}
```

(*End definition for* \MSC*. This function is documented on page 10.*)

Patching tikzinput, if loaded

```
\@ifpackageloaded{tikzinput}{
  \RequirePackage{stex-tikzinput}
}{}
⟨/package⟩
```

Chapter 29

sTeX -Metatheory Implementation

```
⟨*package⟩
⟨@@=stex_modules⟩

%%%%%%%%%%%%%   metatheory.dtx   %%%%%%%%%%%%%

\str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
\begingroup
\stex_module_setup:nn{
  ns=\c_stex_metatheory_ns_str,
  meta=NONE
}{Metatheory}
\stex_reactivate_macro:N \symdecl
\stex_reactivate_macro:N \notation
\stex_reactivate_macro:N \symdef
\ExplSyntaxOff
\csname stex_suppress_html:n\endcsname{
  % is-a (a:A, a \in A, a is an A, etc.)
  \symdecl[args=ai]{isa}
  \notation[typed]{isa}{#1 \comp{:} #2}{#1 \comp, #2}
  \notation[in]{isa}{#1 \comp\in #2}{#1 \comp, #2}
  \notation[pred]{isa}{#2\comp(#1 \comp)}{#1 \comp, #2}

  % bind (\forall, \Pi, \lambda etc.)
  \symdecl[args=Bi]{bind}
  \notation[forall]{bind}{\comp\forall #1.\;#2}{#1 \comp, #2}
  \notation[Pi]{bind}{\comp\prod_{#1}#2}{#1 \comp, #2}
  \notation[depfun]{bind}{\comp( #1 \comp{)\;\to\;} #2}{#1 \comp, #2}

  % dummy variable
  \symdecl{dummyvar}
  \notation[underscore]{dummyvar}{\comp\_}
  \notation[dot]{dummyvar}{\comp\cdot}
  \notation[dash]{dummyvar}{\comp{{\rm --}}}

  %fromto (function space, Hom-set, implication etc.)
  \symdecl[args=ai]{fromto}
  \notation[xarrow]{fromto}{#1 \comp\to #2}{#1 \comp\times #2}
  \notation[arrow]{fromto}{#1 \comp\to #2}{#1 \comp\to #2}

  % mapto (lambda etc.)
  %\symdecl[args=Bi]{mapto}
  %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
  %\notation[lambda]{mapto}{\comp\lambda #1 \comp.\; #2}{#1 \comp, #2}
  %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp.\; #2}{#1 \comp, #2}

  % function/operator application
  \symdecl[args=ia]{apply}
  \notation[prec=0;0x\infprec,parens]{apply}{#1 \comp( #2 \comp)}{#1 \comp, #2}
  \notation[prec=0;0x\infprec,lambda]{apply}{#1 \; #2 }{#1 \; #2}

  % ‘‘type’’ of all collections (sets,classes,types,kinds)
  \symdecl{collection}
  \notation[U]{collection}{\comp{\mathcal{U}}}
  \notation[set]{collection}{\comp{\textsf{Set}}}

  % sequences
  \symdecl[args=1]{seqtype}
  \notation[kleene]{seqtype}{#1^{\comp\ast}}

  \symdef[args=2,li]{sequence-index}{#1_{#2}}
  \notation[ui]{sequence-index}{#1^{#2}}

  %\symdef[args=3,li]{sequence-from-to}{#1_{#2}\comp{,\ellipses,}#1_{#3}}
  %\notation[ui]{sequence-from-to}{#1^{#2}\comp{,\ellipses,}#1^{#3}}
  % ^ superceded by \aseqfromto and \livar/\uivar

  \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{,\ellipses}}{#1\comp,#2}
  \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{,\ellipses,}#2}{#1\comp,#2}
  \symdef[args=aii,prec=nobrackets]{aseqfromtovia}{#1\comp{,\ellipses,}#2\comp{,\ellipses,}#

  % letin (‘‘let’’, local definitions, variable substitution)
  \symdecl[args=bii]{letin}
  \notation[let]{letin}{\comp{{\rm let}}\;#1\comp{=}#2\;\comp{{\rm in}}\;#3}
  \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
  \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}

  % structures
  \symdecl*[args=1]{module-type}
  \notation{module-type}{\mathtt{MOD} #1}
  \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
  \notation[angle,prec=nobrackets]{mathstruct}{\comp\langle #1 \comp\rangle}{#1 \comp, #2}

}
  \ExplSyntaxOn
  \stex_add_to_current_module:n{
    \let\nappa\apply
    \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
    \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
    \def\livar{\csname sequence-index\endcsname[li]}
    \def\uivar{\csname sequence-index\endcsname[ui]}
    \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
    \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
    \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
  }
\__stex_modules_end_module:
\endgroup
```
⟨/package⟩

Chapter 30

Tikzinput Implementation

```
⟨*package⟩

%%%%%%%%%%%%   tikzinput.dtx   %%%%%%%%%%%%

\ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
\RequirePackage{l3keys2e}

\keys_define:nn { tikzinput } {
  image   .bool_set:N   = \c_tikzinput_image_bool,
  image   .default:n    = false ,
  unknown   .code:n       = {}
}

\ProcessKeysOptions { tikzinput }

\bool_if:NTF \c_tikzinput_image_bool {
  \RequirePackage{graphicx}

  \providecommand\usetikzlibrary[]{}
  \newcommand\tikzinput[2][]{\includegraphics[#1]{#2}}
}{
  \RequirePackage{tikz}
  \RequirePackage{standalone}

  \newcommand \tikzinput [2] [] {
    \setkeys{Gin}{#1}
    \ifx \Gin@ewidth \Gin@exclamation
      \ifx \Gin@eheight \Gin@exclamation
        \input { #2 }
      \else
        \resizebox{!}{ \Gin@eheight }{
          \input { #2 }
        }
      \fi
    \else
      \ifx \Gin@eheight \Gin@exclamation
        \resizebox{ \Gin@ewidth }{!}{
          \input { #2 }
        }
      \else
        \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
          \input { #2 }
        }
      \fi
    \fi
  }
}

\newcommand \ctikzinput [2] [] {
  \begin{center}
    \tikzinput [#1] {#2}
  \end{center}
}

\@ifpackageloaded{stex}{
  \RequirePackage{stex-tikzinput}
}{}

⟨/package⟩
⟨*stex⟩
\ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
\RequirePackage{stex}
\RequirePackage{tikzinput}

\newcommand\mhtikzinput[2][]{%
  \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
  \stex_in_repository:nn\Gin@mhrepos{
    \tikzinput[#1]{\mhpath{##1}{#2}}
  }
}
\newcommand\cmhtikzinput[2][]{\begin{center}\mhtikzinput[#1]{#2}\end{center}}
⟨/stex⟩
```

LocalWords: bibfolder jobname.dtx tikzinput.dtx usetikzlibrary Gin@ewidth Gin@eheight
LocalWords: resizebox ctikzinput mhtikzinput Gin@mhrepos mhpath

Chapter 31

document-structure.sty Implementation

31.1 The OMDoc Class

The functionality is spread over the `omdoc` class and package. The class provides the `document` environment and the `omdoc` element corresponds to it, whereas the package provides the concrete functionality.

```
⟨*cls⟩
⟨@@=document_structure⟩
\ProvidesExplClass{omdoc}{2020/10/19}{1.4}{OMDoc Documents}
\RequirePackage{l3keys2e,expl-keystr-compat}
```

31.2 Class Options

`\omdoc@cls@class` To initialize the `omdoc` class, we declare and process the necessary options using the `kvoptions` package for key/value options handling. For `omdoc.cls` this is quite simple. We have options `report` and `book`, which set the `\omdoc@cls@class` macro and pass on the macro to `omdoc.sty` for further processing.

```
\keys_define:nn{ document-structure / pkg }{
  class       .str_set_x:N  = \c_document_structure_class_str,
  minimal     .bool_set:N   = \c_document_structure_minimal_bool,
  report      .code:n       = {
    \ClassWarning{omdoc}{the option 'report' is deprecated, use 'class=report', instead}
    \str_set:Nn \c_document_structure_class_str {report}
  },
  book        .code:n       = {
    \ClassWarning{omdoc}{the option 'book' is deprecated, use 'class=book', instead}
    \str_set:Nn \c_document_structure_class_str {book}
  },
  bookpart    .code:n       = {
    \ClassWarning{omdoc}{the option 'bookpart' is deprecated, use 'class=book,topsect=chapte
    \str_set:Nn \c_document_structure_class_str {book}
    \str_set:Nn \c_document_structure_topsect_str {chapter}
  },
  docopt      .str_set_x:N  = \c_document_structure_docopt_str,
  unknown     .code:n       = {
    \PassOptionsToPackage{ \CurrentOption }{ omdoc }
  }
}
\ProcessKeysOptions{ document-structure / pkg }
\str_if_empty:NT \c_document_structure_class_str {
  \str_set:Nn \c_document_structure_class_str {article}
}
\exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
  {\c_document_structure_class_str}

```

31.3 Beefing up the `document` environment

Now, – unless the option `minimal` is defined – we include the `stex` package

```
\RequirePackage{omdoc}
\bool_if:NF \c_document_structure_minimal_bool {
\RequirePackage{stex-compatibility}
```

And define the environments we need. The top-level one is the `document` environment, which we redefined so that we can provide keyval arguments.

`document` For the moment we do not use them on the LaTeX level, but the document identifier is picked up by LaTeXML.[15] EdN:15

```
\keys_define:nn { document-structure / document }{
  id .str_set_x:N = \c_document_structure_document_id_str
}
\let\__document_structure_orig_document=\document
\renewcommand{\document}[1][]{
  \keys_set:nn{ document-structure / document }{ #1 }
  \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
  \__document_structure_orig_document
}
```

Finally, we end the test for the `minimal` option.

```
}
⟨/cls⟩
```

31.4 Implementation: OMDoc Package

```
⟨*package⟩
\ProvidesExplPackage{omdoc}{2020/10/19}{1.4}{OMDoc document Structure}
\RequirePackage{expl-keystr-compat,l3keys2e}
```

31.5 Package Options

We declare some switches which will modify the behavior according to the package options. Generally, an option `xxx` will just set the appropriate switches to true (otherwise they stay false).

[15]EdNote: faking documentkeys for now. @HANG, please implement

```

\keys_define:nn{ document-structure / pkg }{
  class       .str_set_x:N  = \c_document_structure_class_str,
  topsect     .str_set_x:N  = \c_document_structure_topsect_str,
%  showignores .bool_set:N   = \c_document_structure_showignores_bool,
}
\ProcessKeysOptions{ document-structure / pkg }
\str_if_empty:NT \c_document_structure_class_str {
  \str_set:Nn \c_document_structure_class_str {article}
}
\str_if_empty:NT \c_document_structure_topsect_str {
  \str_set:Nn \c_document_structure_topsect_str {section}
}
```

Then we need to set up the packages by requiring the `sref` package to be loaded.

```
\RequirePackage{xspace}
\RequirePackage{comment}
\@ifpackageloaded{babel}{}{\RequirePackage[base]{babel}}
```

We set up triggers for the other languages, currently only German.

```
\@ifpackageloaded{babel}{
    \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
    \clist_if_in:NnT \l_tmpa_clist {ngerman}{
      \input{omdoc-ngerman.ldf}
    }
}{}
%\AfterBabelLanguage{ngerman}{\input{omdoc-ngerman.ldf}}
```

`\section@level` Finally, we set the `\section@level` macro that governs sectioning. The default is two (corresponding to the `article` class), then we set the defaults for the standard classes `book` and `report` and then we take care of the levels passed in via the `topsect` option.

```
\int_new:N \l_document_structure_section_level_int
\str_case:VnF \c_document_structure_topsect_str {
  {part}{
    \int_set:Nn \l_document_structure_section_level_int {0}
  }
  {chapter}{
    \int_set:Nn \l_document_structure_section_level_int {1}
  }
}{
  \str_case:VnF \c_document_structure_class_str {
    {book}{
      \int_set:Nn \l_document_structure_section_level_int {0}
    }
    {report}{
      \int_set:Nn \l_document_structure_section_level_int {0}
    }
  }{
    \int_set:Nn \l_document_structure_section_level_int {2}
  }
}
```

31.6 Document Structure

The structure of the document is given by the `omgroup` environment just like in OMDoc. The hierarchy is adjusted automatically according to the LaTeX class in effect.

`\currentsectionlevel` For the `\currentsectionlevel` and `\Currentsectionlevel` macros we use an internal macro `\current@section@level` that only contains the keyword (no markup). We initialize it with “document” as a default. In the generated OMDoc, we only generate a text element of class `omdoc_currentsectionlevel`, wich will be instantiated by CSS later.[16] EdN:16

```
\def\current@section@level{document}%
\newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
\newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
\cs_new_protected:Npn \skipomgroup {
  \ifcase\l_document_structure_section_level_int
  \or\stepcounter{part}
  \or\stepcounter{chapter}
  \or\stepcounter{section}
  \or\stepcounter{subsection}
  \or\stepcounter{subsubsection}
  \or\stepcounter{paragraph}
  \or\stepcounter{subparagraph}
  \fi
}
```

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

`blindomgroup`

```
\newcommand\at@begin@blindomgroup[1]{}
\newenvironment{blindomgroup}
{
  \int_incr:N\l_document_structure_section_level_int
  \at@begin@blindomgroup\l_document_structure_section_level_int
}{}
```

`\omgroup@nonum` convenience macro: `\omgroup@nonum{`⟨*level*⟩`}{`⟨*title*⟩`}` makes an unnumbered sectioning with title ⟨*title*⟩ at level ⟨*level*⟩.

```
\newcommand\omgroup@nonum[2]{
  \ifx\hyper@anchor\@undefined\else\phantomsection\fi
  \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
}
```

(*End definition for* `\omgroup@nonum`. *This function is documented on page* **??**.)

`\omgroup@num` convenience macro: `\omgroup@nonum{`⟨*level*⟩`}{`⟨*title*⟩`}` makes numbered sectioning with title ⟨*title*⟩ at level ⟨*level*⟩. We have to check the `short` key was given in the `omgroup` environment and – if it is use it. But how to do that depends on whether the `rdfmeta` package has been loaded. In the end we call `\sref@label@id` to enable crossreferencing.

```
\newcommand\omgroup@num[2]{
```

[16] EdNote: MK: we may have to experiment with the more powerful uppercasing macro from `mfirstuc.sty` once we internationalize.

```
  \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
    \@nameuse{#1}{#2}
  }{
    \cs_if_exist:NTF\rdfmeta@sectioning{
      \@nameuse{rdfmeta@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
    }{
      \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
    }
  }
%\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\omgroup@id
}
```

(*End definition for* `\omgroup@num`. *This function is documented on page* **??**.)

`omgroup`

```
\keys_define:nn { document-structure / omgroup }{
  id            .str_set_x:N = \l__document_structure_omgroup_id_str,
  date          .str_set_x:N = \l__document_structure_omgroup_date_str,
  creators      .clist_set:N = \l__document_structure_omgroup_creators_clist,
  contributors  .clist_set:N = \l__document_structure_omgroup_contributors_clist,
  srccite       .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
  type          .tl_set:N    = \l__document_structure_omgroup_type_tl,
  short         .tl_set:N    = \l__document_structure_omgroup_short_tl,
  display       .tl_set:N    = \l__document_structure_omgroup_display_tl,
  intro         .tl_set:N    = \l__document_structure_omgroup_intro_tl,
  loadmodules   .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
}
\cs_new_protected:Nn \__document_structure_omgroup_args:n {
  \str_clear:N \l__document_structure_omgroup_id_str
  \str_clear:N \l__document_structure_omgroup_date_str
  \clist_clear:N \l__document_structure_omgroup_creators_clist
  \clist_clear:N \l__document_structure_omgroup_contributors_clist
  \tl_clear:N \l__document_structure_omgroup_srccite_tl
  \tl_clear:N \l__document_structure_omgroup_type_tl
  \tl_clear:N \l__document_structure_omgroup_short_tl
  \tl_clear:N \l__document_structure_omgroup_display_tl
  \tl_clear:N \l__document_structure_omgroup_intro_tl
  \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
  \keys_set:nn { document-structure / omgroup } { #1 }
}
```

`\at@begin@omgroup` we define a switch for numbering lines and a hook for the beginning of groups: The `\at@begin@omgroup` macro allows customization. It is run at the beginning of the `omgroup`, i.e. after the section heading.

```
\newif\if@mainmatter\@mainmattertrue
\newcommand\at@begin@omgroup[3][]{}
```

Then we define a helper macro that takes care of the sectioning magic. It comes with its own key/value interface for customization.

```
\keys_define:nn { document-structure / sectioning }{
  name    .str_set_x:N  = \l__document_structure_sect_name_str   ,
  ref     .str_set_x:N  = \l__document_structure_sect_ref_str    ,
  clear   .bool_set:N   = \l__document_structure_sect_clear_bool ,
  num     .bool_set:N   = \l__document_structure_sect_num_bool   ,
}
```

```
\cs_new_protected:Nn \__document_structure_sect_args:n {
  \str_clear:N \l__document_structure_sect_name_str
  \str_clear:N \l__document_structure_sect_ref_str
  \bool_set_false:N \l__document_structure_sect_clear_bool
  \bool_set_false:N \l__document_structure_sect_num_bool
  \keys_set:nn { document-structure / sectioning } { #1 }
}
\newcommand\omdoc@sectioning[3][]{
  \__document_structure_sect_args:n {#1 }
  \let\omdoc@sect@name\l__document_structure_sect_name_str
  \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
  \if@mainmatter% numbering not overridden by frontmatter, etc.
    \bool_if:NTF \l__document_structure_sect_num_bool {
      \omgroup@num{#2}{#3}
    }{
      \omgroup@nonum{#2}{#3}
    }
    \def\current@section@level{\omdoc@sect@name}
  \else
    \omgroup@nonum{#2}{#3}
  \fi
}% if@mainmatter
```

and another one, if redefines the `\addtocontentsline` macro of LaTeX to import the respective macros. It takes as an argument a list of module names.

```
\newcommand\omgroup@redefine@addtocontents[1]{%
%\edef\__document_structureimport{#1}%
%\@for\@I:=\__document_structureimport\do{%
%\edef\@path{\csname module@\@I  @path\endcsname}%
%\@ifundefined{tf@toc}\relax%
%     {\protected@write\tf@toc{}{\string\@requiremodules{\@path}}}}
%\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
%\def\addcontentsline##1##2##3{%
%\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}
%\else% hyperref.sty not loaded
%\def\addcontentsline##1##2##3{%
%\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}{
%\fi
}% hypreref.sty loaded?
```

now the `omgroup` environment itself. This takes care of the table of contents via the helper macro above and then selects the appropriate sectioning command from `article.cls`. It also registeres the current level of omgroups in the `\omgroup@level` counter.

```
\int_new:N \l_document_structure_omgroup_level_int
\newenvironment{omgroup}[2][]% keys, title
{
  \__document_structure_omgroup_args:n { #1 }%\sref@target%
```

If the `loadmodules` key is set on `\begin{omgroup}`, we redefine the `\addcontetsline` macro that determines how the sectioning commands below construct the entries for the table of contents.

```
  \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
    \omgroup@redefine@addtocontents{
      %\@ifundefined{module@id}\used@modules%
      %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}
    }
  }
```

now we only need to construct the right sectioning depending on the value of `\section@level`.

```
  \int_incr:N \l_document_structure_omgroup_level_int
  \int_incr:N\l_document_structure_section_level_int
  \ifcase\l_document_structure_section_level_int
    \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
    \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
    \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
    \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
    \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
    \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#
    \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{paragr
  \fi
  \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
  \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
}% for customization
{}
```

and finally, we localize the sections

```
\newcommand\omdoc@part@kw{Part}
\newcommand\omdoc@chapter@kw{Chapter}
\newcommand\omdoc@section@kw{Section}
\newcommand\omdoc@subsection@kw{Subsection}
\newcommand\omdoc@subsubsection@kw{Subsubsection}
\newcommand\omdoc@paragraph@kw{paragraph}
\newcommand\omdoc@subparagraph@kw{subparagraph}
```

31.7 Front and Backmatter

Index markup is provided by the `omtext` package [Koh20c], so in the `omdoc` package we only need to supply the corresponding `\printindex` command, if it is not already defined

`\printindex`

```
\providecommand\printindex{\IfFileExists{\jobname.ind}{\input{\jobname.ind}}{}}
```

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

some classes (e.g. `book.cls`) already have `\frontmatter`, `\mainmatter`, and `\backmatter` macros. As we want to define `frontmatter` and `backmatter` environments, we save their behavior (possibly defining it) in `orig@*matter` macros and make them undefined (so that we can define the environments).

```
\cs_if_exist:NTF\frontmatter{
  \let\__document_structure_orig_frontmatter\frontmatter
  \let\frontmatter\relax
}{
  \tl_set:Nn\__document_structure_orig_frontmatter{
    \clearpage
    \@mainmatterfalse
    \pagenumbering{roman}
  }
}
\cs_if_exist:NTF\backmatter{
  \let\__document_structure_orig_backmatter\backmatter
  \let\backmatter\relax
}{
  \tl_set:Nn\__document_structure_orig_backmatter{
    \clearpage
    \@mainmatterfalse
    \pagenumbering{roman}
  }
}
```

Using these, we can now define the `frontmatter` and `backmatter` environments

`frontmatter` we use the `\orig@frontmatter` macro defined above and `\mainmatter` if it exists, otherwise we define it.

```
\newenvironment{frontmatter}{
  \__document_structure_orig_frontmatter
}{
  \cs_if_exist:NTF\mainmatter{
    \mainmatter
  }{
    \clearpage
    \@mainmattertrue
    \pagenumbering{arabic}
  }
}
```

`backmatter` As backmatter is at the end of the document, we do nothing for `\endbackmatter`.

```
\newenvironment{backmatter}{
  \__document_structure_orig_backmatter
}{
  \cs_if_exist:NTF\mainmatter{
    \mainmatter
  }{
    \clearpage
    \@mainmattertrue
    \pagenumbering{arabic}
  }
}
```

finally, we make sure that page numbering is arabic and we have main matter as the default

```
\@mainmattertrue\pagenumbering{arabic}
```

`\prematurestop` We initialize `\afterprematurestop`, and provide `\prematurestop@endomgroup` which looks up `\omgroup@level` and recursively ends enough `{omgroup}`s.

```
\newcommand\afterprematurestop{}
\def\prematurestop@endomgroup{
  \int_compare:nNnF \l_document_structure_omgroup_level_int = 0 {
    \end{omgroup}
    \int_decr:N \l_document_structure_omgroup_level_int
    \prematurestop@endomgroup
  }
}
\providecommand\prematurestop{
  \message{Stopping sTeX processing prematurely}
  \prematurestop@endomgroup
  \afterprematurestop
  \end{document}
}
```

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

31.8 Global Variables

`\setSGvar` set a global variable

```
\RequirePackage{etoolbox}
\newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}
```

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

`\useSGvar` use a global variable

```
\newrobustcmd\useSGvar[1]{%
  \@ifundefined{sTeX@Gvar@#1}
  {\PackageError{omdoc}
    {The sTeX Global variable #1 is undefined}
    {set it with \protect\setSGvar}}
\@nameuse{sTeX@Gvar@#1}}
```

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

`\ifSGvar` execute something conditionally based on the state of the global variable.

```
\newrobustcmd\ifSGvar[3]{\def\@test{#2}%
  \@ifundefined{sTeX@Gvar@#1}
  {\PackageError{omdoc}
    {The sTeX Global variable #1 is undefined}
    {set it with \protect\setSGvar}}
  {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}
```

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

Chapter 32

MiKoSlides – Implementation

32.1 Class and Package Options

We define some Package Options and switches for the mikoslides class and activate them by passing them on to beamer.cls and omdoc.cls and the mikoslides package. We pass the nontheorem option to the statements package when we are not in notes mode, since the beamer package has its own (overlay-aware) theorem environments.

```
⟨*cls⟩
⟨@@=mikoslides⟩
\ProvidesExplClass{mikoslides}{2020/12/06}{1.3}{MiKo slides Class}
\RequirePackage{l3keys2e,expl-keystr-compat}

\keys_define:nn{mikoslides / cls}{
  class   .code:n   = {
    \PassOptionsToClass{\CurrentOption}{omdoc}
    \str_if_eq:nnT{#1}{book}{
      \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
    }
    \str_if_eq:nnT{#1}{report}{
      \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
    }
  },
  notes   .bool_set:N  = \c__mikoslides_notes_bool ,
  slides  .code:n      = { \bool_set_false:N \c__mikoslides_notes_bool },
  unknown .code:n      = {
    \PassOptionsToClass{\CurrentOption}{omdoc}
    \PassOptionsToClass{\CurrentOption}{beamer}
    \PassOptionsToPackage{\CurrentOption}{mikoslides}
  }
}
\ProcessKeysOptions{ mikoslides / cls }
\bool_if:NTF \c__mikoslides_notes_bool {
  \PassOptionsToPackage{notes=true}{mikoslides}
}{
  \PassOptionsToPackage{notes=false}{mikoslides}
}
⟨/cls⟩
```

now we do the same for the `mikoslides` package.

```
⟨*package⟩
\ProvidesExplPackage{mikoslides}{2020/12/06}{1.3}{MiKo slides Package}
\RequirePackage{l3keys2e,expl-keystr-compat}

\keys_define:nn{mikoslides / pkg}{
  topsect         .str_set_x:N  = \c__mikoslides_topsect_str,
  defaulttopsect  .str_set_x:N  = \c__mikoslides_defaulttopsec_str,
  notes           .bool_set:N   = \c__mikoslides_notes_bool ,
  slides          .code:n       = { \bool_set_false:N \c__mikoslides_notes_bool },
  sectocframes    .bool_set:N   = \c__mikoslides_sectocframes_bool ,
  frameimages     .bool_set:N   = \c__mikoslides_frameimages_bool ,
  fiboxed         .bool_set:N   = \c__mikoslides_fiboxed_bool ,
  noproblems      .bool_set:N   = \c__mikoslides_noproblems_bool,
  unknown         .code:n       = {
    \PassOptionsToClass{\CurrentOption}{stex}
    \PassOptionsToClass{\CurrentOption}{tikzinput}
  }
}
\ProcessKeysOptions{ mikoslides / pkg }
\newif\ifnotes
\bool_if:NTF \c__mikoslides_notes_bool {
  \notestrue
}{
  \notesfalse
}

```

we give ourselves a macro `\@@topsect` that needs only be evaluated once, so that the `\ifdefstring` conditionals work below.

```
\str_if_empty:NTF \c__mikoslides_topsect_str {
  \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_defaulttopsec_str
}{
  \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_topsect_str
}
⟨/package⟩
```

Depending on the options, we either load the `article`-based `omdoc` or the `beamer` class (and set some counters).

```
⟨*cls⟩
\bool_if:NTF \c__mikoslides_notes_bool {
  \LoadClass{omdoc}
}{
  \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
  \newcounter{Item}
  \newcounter{paragraph}
  \newcounter{subparagraph}
  \newcounter{Hfootnote}
  \RequirePackage{omdoc}
}
```

now it only remains to load the `mikoslides` package that does all the rest.

```
\RequirePackage{mikoslides}
⟨/cls⟩
```

In `notes` mode, we also have to make the `beamer`-specific things available to `article` via the `beamerarticle` package. We use options to avoid loading theorem-like environments, since we want to use our own from the sTeX packages. The first batch of packages we want are loaded on `mikoslides.sty`. These are the general ones, we will load the sTeX-specific ones after we have done some work (e.g. defined the counters `m*`). Only the `stex-logo` package is already needed now for the default theme.

```
⟨*package⟩
\bool_if:NT \c__mikoslides_notes_bool {
  \RequirePackage{a4wide}
  \RequirePackage{marginnote}
  \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
  \RequirePackage{mdframed}
  \RequirePackage[noxcolor,noamsthm]{beamerarticle}
  \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
}
\RequirePackage{stex-compatibility}
\RequirePackage{stex-tikzinput}
\RequirePackage{etoolbox}
\RequirePackage{amssymb}
\RequirePackage{amsmath}
\RequirePackage{comment}
\RequirePackage{textcomp}
\RequirePackage{url}
\RequirePackage{graphicx}
\RequirePackage{pgf}
```

32.2 Notes and Slides

For the lecture notes cases, we also provide the `\usetheme` macro that would otherwise come from the the `beamer` class. While the latter loads `beamertheme`⟨*theme*⟩.sty, the notes version loads `beamernotestheme`⟨*theme*⟩`.sty`.[17] EdN:17

```
\bool_if:NT \c__mikoslides_notes_bool {
  \renewcommand\usetheme[2][]{\usepackage[#1]{beamernotestheme#2}}
}
```

We define the sizes of slides in the notes. Somehow, we cannot get by with the same here.

```
\newcounter{slide}
\newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
\newlength{\slideheight}\setlength{\slideheight}{9cm}
```

`note` The `note` environment is used to leave out text in the `slides` mode. It does not have a counterpart in OMDoc. So for course notes, we define the `note` environment to be a no-operation otherwise we declare the `note` environment as a comment via the `comment` package.

```
\bool_if:NTF \c__mikoslides_notes_bool {
  \renewenvironment{note}{\ignorespaces}{}
}{
  \excludecomment{note}
}
```

[17] EdNote: MK: This is not ideal, but I am not sure that I want to be able to provide the full theme functionality there.

We first set up the slide boxes in `article` mode. We set up sizes and provide a box register for the frames and a counter for the slides.

```
\bool_if:NT \c__mikoslides_notes_bool {
  \newlength{\slideframewidth}
  \setlength{\slideframewidth}{1.5pt}
```

`frame` We first define the keys.

```
  \cs_new_protected:Nn \__mikoslides_do_yes_param:Nn {
    \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
      \bool_set_true:N #1
    }{
      \bool_set_false:N #1
    }
  }
  \keys_define:nn{mikoslides / frame}{
    label               .str_set_x:N  = \l__mikoslides_frame_label_str,
    allowframebreaks    .code:n       = {
      \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowframebreaks_bool { #1 }
    },
    allowdisplaybreaks  .code:n       = {
      \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowdisplaybreaks_bool { #1 }
    },
    fragile             .code:n       = {
      \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_fragile_bool { #1 }
    },
    shrink              .code:n       = {
      \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_shrink_bool { #1 }
    },
    squeeze             .code:n       = {
      \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_squeeze_bool { #1 }
    },
    t                   .code:n       = {
      \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_t_bool { #1 }
    },
  }
  \cs_new_protected:Nn \__mikoslides_frame_args:n {
    \str_clear:N \l__mikoslides_frame_label_str
    \bool_set_true:N \l__mikoslides_frame_allowframebreaks_bool
    \bool_set_true:N \l__mikoslides_frame_allowdisplaybreaks_bool
    \bool_set_true:N \l__mikoslides_frame_fragile_bool
    \bool_set_true:N \l__mikoslides_frame_shrink_bool
    \bool_set_true:N \l__mikoslides_frame_squeeze_bool
    \bool_set_true:N \l__mikoslides_frame_t_bool
    \keys_set:nn { mikoslides / frame }{ #1 }
  }
```

We define the environment, read them, and construct the slide number and label.

```
  \renewenvironment{frame}[1][]{
    \__mikoslides_frame_args:n{#1}
    \sffamily
    \stepcounter{slide}
    \def\@currentlabel{\theslide}
    \str_if_empty:NF \l__mikoslides_frame_label_str {
      \label{\l__mikoslides_frame_label_str}
      }
```

We redefine the itemize environment so that it looks more like the one in beamer.

```
      \def\itemize@level{outer}
      \def\itemize@outer{outer}
      \def\itemize@inner{inner}
      \renewcommand\newpage{\addtocounter{framenumber}{1}}
      \newcommand\metakeys@show@keys[2]{\marginnote{{\scriptsize ##2}}}
      \renewenvironment{itemize}{
        \ifx\itemize@level\itemize@outer
          \def\itemize@label{$\rhd$}
        \fi
        \ifx\itemize@level\itemize@inner
          \def\itemize@label{$\scriptstyle\rhd$}
        \fi
        \begin{list}
        {\itemize@label}
        {\setlength{\labelsep}{.3em}
         \setlength{\labelwidth}{.5em}
         \setlength{\leftmargin}{1.5em}
        }
        \edef\itemize@level{\itemize@inner}
      }{
        \end{list}
      }
```

We create the box with the mdframed environment from the equinymous package.

```
      \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth
    }{
      \medskip\miko@slidelabel\end{mdframed}
    }
```

Now, we need to redefine the frametitle (we are still in course notes mode).

\frametitle

```
    \renewcommand{\frametitle}[1]{{\Large\bf\sf\color{blue}{#1}}\medskip}
  }
```

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

EdN:18 \pause [18]

```
  \bool_if:NT \c__mikoslides_notes_bool {
    \newcommand\pause{}
  }
```

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

nomtext

```
  \bool_if:NTF \c__mikoslides_notes_bool {
    \newenvironment{nomtext}[1][]{\begin{omtext}[#1]}{\end{omtext}}
  }{
    \excludecomment{nomtext}
  }
```

[18] EdNote: MK: fake it in notes mode for now

nomgroup

```
\bool_if:NTF \c__mikoslides_notes_bool {
  \newenvironment{nomgroup}[2][]{\begin{omgroup}[#1]{#2}}{\end{omgroup}}
}{
  \excludecomment{nomgroup}
}
```

ndefinition

```
\bool_if:NTF \c__mikoslides_notes_bool {
  \newenvironment{ndefinition}[1][]{\begin{definition}[#1]}{\end{definition}}
}{
  \excludecomment{ndefinition}
}
```

nassertion

```
\bool_if:NTF \c__mikoslides_notes_bool {
  \newenvironment{nassertion}[1][]{\begin{assertion}[#1]}{\end{assertion}}
}{
  \excludecomment{nassertion}
}
```

nsproof

```
\bool_if:NTF \c__mikoslides_notes_bool {
  \newenvironment{nsproof}[2][]{\begin{sproof}[#1]{#2}}{\end{sproof}}
}{
  \excludecomment{nsproof}
}
```

nexample

```
\bool_if:NTF \c__mikoslides_notes_bool {
  \newenvironment{nexample}[1][]{\begin{example}[#1]}{\end{example}}
}{
  \excludecomment{nexample}
}
```

\inputref@*skip We customize the hooks for in \inputref.

```
\def\inputref@preskip{\smallskip}
\def\inputref@postskip{\medskip}
```

(*End definition for* \inputref@*skip*. This function is documented on page* **??**.)

\inputref*

```
\let\orig@inputref\inputref
\def\inputref{\@ifstar\ninputref\orig@inputref}
\newcommand\ninputref[2][]{
  \bool_if:NT \c__mikoslides_notes_bool {
    \orig@inputref[#1]{#2}
  }
}
```

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

32.3 Header and Footer Lines

Now, we set up the infrastructure for the footer line of the slides, we use boxes for the logos, so that they are only loaded once, that considerably speeds up processing.

`\setslidelogo` The default logo is the sTeX logo. Customization can be done by `\setslidelogo{`⟨*logo name*⟩`}`.

```
\newlength{\slidelogoheight}

\bool_if:NTF \c__mikoslides_notes_bool {
  \setlength{\slidelogoheight}{.4cm}
}{
  \setlength{\slidelogoheight}{1cm}
}
\newsavebox{\slidelogo}
\sbox{\slidelogo}{\sTeX}
\newrobustcmd{\setslidelogo}[1]{
  \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
}
```

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

`\setsource` `\source` stores the writer's name. By default it is *Michael Kohlhase* since he is the main user and designer of this package. `\setsource{`⟨*name*⟩`}` can change the writer's name.

```
\def\source{Michael Kohlhase}% customize locally
\newrobustcmd{\setsource}[1]{\def\source{#1}}
```

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

`\setlicensing` Now, we set up the copyright and licensing. By default we use the Creative Commons Attribuition-ShareAlike license to strengthen the public domain. If package `hyperref` is loaded, then we can attach a hyperlink to the license logo. `\setlicensing[`⟨*url*⟩`]{`⟨*logo name*⟩`}` is used for customization, where ⟨*url*⟩ is optional.

```
\def\copyrightnotice{\footnotesize\copyright :\hspace{.3ex}{\source}}
\newsavebox{\cclogo}
\sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
\newif\ifcchref\cchreffalse
\AtBeginDocument{
  \@ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
}
\def\licensing{
  \ifcchref
    \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
  \else
    {\usebox{\cclogo}}
  \fi
}
\newrobustcmd{\setlicensing}[2][]{
  \def\@url{#1}
  \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
  \ifx\@url\@empty
    \def\licensing{{\usebox{\cclogo}}}
  \else
    \def\licensing{
      \ifcchref
      \href{#1}{\usebox{\cclogo}}
      \else
      {\usebox{\cclogo}}
      \fi
    }
  \fi
}
```

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

EdN:19 `\slidelabel` Now, we set up the slide label for the `article` mode.[19]

```
\newrobustcmd\miko@slidelabel{
  \vbox to \slidelogoheight{
    \vss\hbox to \slidewidth
    {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
  }
}
```

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

32.4 Frame Images

`\frameimage` We have to make sure that the width is overwritten, for that we check the `\Gin@ewidth` macro from the `graphicx` package. We also add the `label` key.

```
\def\Gin@mhrepos{}
\define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
\define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
\newrobustcmd\frameimage[2][]{
  \stepcounter{slide}
  \bool_if:NT \c__mikoslides_frameimages_bool {
    \def\Gin@ewidth{}\setkeys{Gin}{#1}
    \bool_if:NF \c__mikoslides_notes_bool { \vfill }
    \begin{center}
      \bool_if:NTF \c__mikoslides_fiboxed_bool {
        \fbox{
          \ifx\Gin@ewidth\@empty
            \ifx\Gin@mhrepos\@empty
              \mhgraphics[width=\slidewidth,#1]{#2}
            \else
              \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
            \fi
          \else% Gin@ewidth empty
            \ifx\Gin@mhrepos\@empty
              \mhgraphics[#1]{#2}
            \else
              \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
            \fi
          \fi% Gin@ewidth empty
        }
      }{
        \ifx\Gin@ewidth\@empty
```

[19]EdNote: see that we can use the themes for the slides some day. This is all fake.

```
          \ifx\Gin@mhrepos\@empty
            \mhgraphics[width=\slidewidth,#1]{#2}
          \else
            \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
          \fi
          \ifx\Gin@mhrepos\@empty
            \mhgraphics[#1]{#2}
          \else
            \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
          \fi
        \fi% Gin@ewidth empty
      }
     \end{center}
    \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
    \bool_if:NF \c__mikoslides_notes_bool { \vfill }
  }
} % ifmks@sty@frameimages
```

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

32.5 Colors and Highlighting

We first specify sans serif fonts as the default.

```
\sffamily
```

Now, we set up an infrastructure for highlighting phrases in slides. Note that we use content-oriented macros for highlighting rather than directly using color markup. The first thing to to is to adapt the green so that it is dark enough for most beamers

```
\AddToHook{begindocument}{
  \definecolor{green}{rgb}{0,.5,0}
  \definecolor{purple}{cmyk}{.3,1,0,.17}
}
```

We customize the `\defemph`, `\symrefemph`, `\compemph`, and `\titleemph` macros with colors. Furthermore we customize the `\__omtextlec` macro for the appearance of line end comments in `\lec`.

```
% \def\STpresent#1{\textcolor{blue}{#1}}
\def\defemph#1{{\textcolor{magenta}{#1}}}
\def\symrefemph#1{{\textcolor{cyan}{#1}}}
\def\compemph#1{{\textcolor{blue}{#1}}}
\def\titleemph#1{{\textcolor{blue}{#1}}}
\def\__omtext_lec#1{(\textcolor{green}{#1})}
```

I like to use the dangerous bend symbol for warnings, so we provide it here.

`\textwarning` as the macro can be used quite often we put it into a box register, so that it is only loaded once.

```
\pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
\def\smalltextwarning{
  \pgfuseimage{miko@small@dbend}
  \xspace
}
\pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}
```

```
\newrobustcmd\textwarning{
  \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
  \xspace
}
\pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}
\newrobustcmd\bigtextwarning{
  \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
  \xspace
}
```

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

```
\newrobustcmd\putgraphicsat[3]{
  \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
}
\newrobustcmd\putat[2]{
  \begin{picture}(0,0)\put(#1){#2}\end{picture}
}
```

32.6 Sectioning

If the `sectocframes` option is set, then we make section frames. We first define counters for `part` and `chapter`, which `beamer.cls` does not have and we make the `section` counter which it does dependent on `chapter`.

```
\bool_if:NT \c__mikoslides_sectocframes_bool {
  \str_if_eq:VnTF \__mikoslidestopsect{part}{
    \newcounter{chapter}\counterwithin*{section}{chapter}
  }{
    \str_if_eq:VnT\__mikoslidestopsect{chapter}{
      \newcounter{chapter}\counterwithin*{section}{chapter}
    }
  }
}
```

`\section@level` We set the `\section@level` counter that governs sectioning according to the class options. We also introduce the sectioning counters accordingly.

`\section@level`

```
\def\part@prefix{}
\@ifpackageloaded{omdoc}{}{
  \str_case:VnF \__mikoslidestopsect {
    {part}{
      \int_set:Nn \l_document_structure_section_level_int {0}
      \def\thesection{\arabic{chapter}.\arabic{section}}
      \def\part@prefix{\arabic{chapter}.}
    }
    {chapter}{
      \int_set:Nn \l_document_structure_section_level_int {1}
      \def\thesection{\arabic{chapter}.\arabic{section}}
      \def\part@prefix{\arabic{chapter}.}
    }
  }{
    \int_set:Nn \l_document_structure_section_level_int {2}
    \def\part@prefix{}
  }
}

\bool_if:NF \c__mikoslides_notes_bool { % only in slides
```

(*End definition for* `\section@level`. *This function is documented on page* **??**.)

The new counters are used in the `omgroup` environment that choses the LaTeX sectioning macros according to `\section@level`.

`omgroup`

```
  \renewenvironment{omgroup}[2][]{
    \__document_structure_omgroup_args:n { #1 }
    \int_incr:N \l_document_structure_omgroup_level_int
    \int_incr:N \l_document_structure_section_level_int
    \bool_if:NT \c__mikoslides_sectocframes_bool {
      \stepcounter{slide}
      \begin{frame}[noframenumbering]
      \vfill\Large\centering
      \red{
        \ifcase\l_document_structure_section_level_int\or
          \stepcounter{part}
          \def\__mikoslideslabel{\omdoc@part@kw~\Roman{part}}
          \def\currentsectionlevel{\omdoc@part@kw}
        \or
          \stepcounter{chapter}
          \def\__mikoslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
          \def\currentsectionlevel{\omdoc@chapter@kw}
        \or
          \stepcounter{section}
          \def\__mikoslideslabel{\part@prefix\arabic{section}}
          \def\currentsectionlevel{\omdoc@section@kw}
        \or
          \stepcounter{subsection}
          \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
          \def\currentsectionlevel{\omdoc@subsection@kw}
        \or
          \stepcounter{subsubsection}
          \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{su
          \def\currentsectionlevel{\omdoc@subsubsection@kw}
        \or
          \stepcounter{mparagraph}
          \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{msubsection}.\arabic{s
          \def\currentsectionlevel{\omdoc@paragraph@kw}
        \fi% end ifcase
        \__mikoslideslabel%\sref@label@id\__mikoslideslabel
        \quad #2%
      }%
      \vfill%
      \end{frame}%
    }
    \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str%
  }{}
}
```

We set up a `beamer` template for theorems like ams style, but without a block environment.

```
\def\inserttheorembodyfont{\normalfont}
\bool_if:NF \c__mikoslides_notes_bool {
  \defbeamertemplate{theorem begin}{miko}
  {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
    \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
    \inserttheorempunctuation\inserttheorembodyfont\xspace}
  \defbeamertemplate{theorem end}{miko}{}
```

and we set it as the default one.

```
  \setbeamertemplate{theorems}[miko]
```

The following fixes an error I do not understand, this has something to do with beamer compatibility, which has similar definitions but only up to 1.

```
  \expandafter\def\csname Parent2\endcsname{}
}
\bool_if:NT \c__mikoslides_notes_bool {
  \renewenvironment{columns}[1][]{%
    \par\noindent%
    \begin{minipage}%
    \slidewidth\centering\leavevmode%
  }{%
    \end{minipage}\par\noindent%
  }%
  \newsavebox\columnbox%
  \renewenvironment<>{column}[2][]{%
    \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
  }{%
    \end{minipage}\end{lrbox}\usebox\columnbox%
  }%
}
```

```
\bool_if:NTF \c__mikoslides_noproblems_bool {
  \newenvironment{problems}{}{}
}{
  \excludecomment{problems}
}
```

32.7 Excursions

`\excursion` The excursion macros are very simple, we define a new internal macro `\excursionref` and use it in `\excursion`, which is just an `\inputref` that checks if the new macro is defined before formatting the file in the argument.

```
\gdef\printexcursions{}
\newcommand\excursionref[2]{% label, text
  \bool_if:NT \c__mikoslides_notes_bool {
    \begin{omtext}[title=Excursion]
      #2 \sref[fallback=the appendix]{#1}.
    \end{omtext}
  }
}
\newcommand\activate@excursion[2][]{
  \gappto\printexcursions{\inputref[#1]{#2}}
}
\newcommand\excursion[4][]{% repos, label, path, text
  \bool_if:NT \c__mikoslides_notes_bool {
    \activate@excursion[#1]{#3}\excursionref{#2}{#4}
  }
}
```

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

`\excursiongroup`

```
\keys_define:nn{mikoslides / excursiongroup }{
  id        .str_set_x:N  = \l__mikoslides_excursion_id_str,
  intro     .tl_set:N     = \l__mikoslides_excursion_intro_tl,
  mhrepos   .str_set_x:N  = \l__mikoslides_excursion_mhrepos_str
}
\cs_new_protected:Nn \__mikoslides_excursion_args:n {
  \tl_clear:N \l__mikoslides_excursion_intro_tl
  \str_clear:N \l__mikoslides_excursion_id_str
  \str_clear:N \l__mikoslides_excursion_mhrepos_str
  \keys_set:nn {mikoslides / excursiongroup }{ #1 }
}
\newcommand\excursiongroup[1][]{
  \__mikoslides_excursion_args:n{ #1 }
  \ifdefempty\printexcursions{}% only if there are excursions
  {\begin{note}
    \begin{omgroup}[#1]{Excursions}%
      \ifdefempty\l__mikoslides_excursion_intro_tl{}{
        \inputref[\l__mikoslides_excursion_mhrepos_str]{
          \l__mikoslides_excursion_intro_tl
        }
      }
      \printexcursions%
    \end{omgroup}
  \end{note}}
}
⟨/package⟩
```

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

Chapter 33

The Implementation

33.1 Package Options

The first step is to declare (a few) package options that handle whether certain information is printed or not. They all come with their own conditionals that are set by the options.

```
⟨*package⟩
⟨@@=problems⟩
\ProvidesExplPackage{problem}{2019/03/20}{1.3}{Semantic Markup for Problems}
\RequirePackage{l3keys2e,expl-keystr-compat}

\keys_define:nn { problem / pkg }{
  notes     .default:n    = { true },
  notes     .bool_set:N   = \c__problems_notes_bool,
  gnotes    .default:n    = { true },
  gnotes    .bool_set:N   = \c__problems_gnotes_bool,
  hints     .default:n    = { true },
  hints     .bool_set:N   = \c__problems_hints_bool,
  solutions .default:n    = { true },
  solutions .bool_set:N   = \c__problems_solutions_bool,
  pts       .default:n    = { true },
  pts       .bool_set:N   = \c__problems_pts_bool,
  min       .default:n    = { true },
  min       .bool_set:N   = \c__problems_min_bool,
  boxed     .default:n    = { true },
  boxed     .bool_set:N   = \c__problems_boxed_bool,
  unknown   .code:n       = {}
}
\def\solutionstrue{
  \bool_set_true:N \c__problems_solutions_bool
}
\def\solutionsfalse{
  \bool_set_false:N \c__problems_solutions_bool
}

\ProcessKeysOptions{ problem / pkg }
```

Then we make sure that the necessary packages are loaded (in the right versions).

```
\RequirePackage{stex-compatibility}
\RequirePackage{comment}
```

The next package relies on the LaTeX3 kernel, which LaTeXMLonly partially supports. As it is purely presentational, we only load it when the `boxed` option is given and we run LaTeXML.

```
\bool_if:NT \c__problems_boxed_bool { \RequirePackage{mdframed} }
```

`\prob@*@kw` For multilinguality, we define internal macros for keywords that can be specialized in `*.ldf` files.

```
\def\prob@problem@kw{Problem}
\def\prob@solution@kw{Solution}
\def\prob@hint@kw{Hint}
\def\prob@note@kw{Note}
\def\prob@gnote@kw{Grading}
\def\prob@pt@kw{pt}
\def\prob@min@kw{min}
```

(*End definition for* `\prob@*@kw`*. This function is documented on page* **??**.)

For the other languages, we set up triggers

```
\@ifpackageloaded{babel}{
    \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
    \clist_if_in:NnT \l_tmpa_clist {ngerman}{
      \input{problem-ngerman.ldf}
    }
    \clist_if_in:NnT \l_tmpa_clist {finnish}{
      \input{problem-finnish.ldf}
    }
    \clist_if_in:NnT \l_tmpa_clist {french}{
      \input{problem-french.ldf}
    }
    \clist_if_in:NnT \l_tmpa_clist {russian}{
      \input{problem-russian.ldf}
    }
}{}
```

33.2 Problems and Solutions

We now prepare the KeyVal support for problems. The key macros just set appropriate internal macros.

```
\keys_define:nn{ problem / problem }{
  id      .str_set_x:N  = \l__problems_prob_id_str,
  pts     .tl_set:N     = \l__problems_prob_pts_tl,
  min     .tl_set:N     = \l__problems_prob_min_tl,
  title   .tl_set:N     = \l__problems_prob_title_tl,
  refnum  .int_set:N    = \l__problems_prob_refnum_int
}
\cs_new_protected:Nn \__problems_prob_args:n {
  \str_clear:N \l__problems_prob_id_str
  \tl_clear:N \l__problems_prob_pts_tl
  \tl_clear:N \l__problems_prob_min_tl
  \tl_clear:N \l__problems_prob_title_tl
  \int_zero_new:N \l__problems_prob_refnum_int
  \keys_set:nn { problem / problem }{ #1 }
  \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
    \let\l__problems_inclprob_refnum_int\undefined
  }
}
```

Then we set up a counter for problems.

`\numberproblemsin`

```
\newcounter{problem}
\newcommand\numberproblemsin[1]{\@addtoreset{problem}{#1}}
```

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

`\prob@label` We provide the macro `\prob@label` to redefine later to get context involved.

```
\newcommand\prob@label[1]{#1}
```

(*End definition for* `\prob@label`*. This function is documented on page* ??.)

`\prob@number` We consolidate the problem number into a reusable internal macro

```
\newcommand\prob@number{
  \int_if_exist:NTF \l__problems_inclprob_refnum_int {
    \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
  }{
    \int_if_exist:NTF \l__problems_prob_refnum_int {
      \prob@label{\int_use:N \l__problems_prob_refnum_int }
    }{
        \prob@label\theproblem
    }
  }
}
```

(*End definition for* `\prob@number`*. This function is documented on page* ??.)

`\prob@title` We consolidate the problem title into a reusable internal macro as well. `\prob@title` takes three arguments the first is the fallback when no title is given at all, the second and third go around the title, if one is given.

```
\newcommand\prob@title[3]{%
  \tl_if_exist:NTF \l__problems_inclprob_title_tl {
    #2 \l__problems_inclprob_title_tl #3
  }{
    \tl_if_exist:NTF \l__problems_prob_title_tl {
      #2 \l__problems_prob_title_tl #3
    }{
      #1
    }
  }
}
```

(*End definition for* `\prob@title`*. This function is documented on page* ??.)

With these the problem header is a one-liner

`\prob@heading` We consolidate the problem header line into a separate internal macro that can be reused in various settings.

```
\def\prob@heading{
  \prob@problem@kw~\prob@number\prob@title{~}{~(}{)\strut}
  %\sref@label@id{\prob@problem@kw~\prob@number}{}
}
```

(*End definition for* `\prob@heading`. *This function is documented on page* **??**.)

With this in place, we can now define the `problem` environment. It comes in two shapes, depending on whether we are in boxed mode or not. In both cases we increment the problem number and output the points and minutes (depending) on whether the respective options are set.

`problem`

```
\newenvironment{problem}[1][]{
  \__problems_prob_args:n{#1}%\sref@target%
  \@in@omtexttrue% we are in a statement (for inline definitions)
  \stepcounter{problem}\record@problem
  \def\current@section@level{\prob@problem@kw}
  \par\noindent\textbf\prob@heading\show@pts\show@min\\\ignorespacesandpars
}%
{\smallskip}
\bool_if:NT \c__problems_boxed_bool {
  \surroundwithmdframed{problem}
}
```

`\record@problem` This macro records information about the problems in the `*.aux` file.

```
\def\record@problem{
  \protected@write\@auxout{}
  {
    \string\@problem{\prob@number}
    {
      \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
        \l__problems_inclprob_pts_tl
      }{
        \l__problems_prob_pts_tl
      }
    }%
    {
      \tl_if_exist:NTF \l__problems_inclprob_min_tl {
        \l__problems_inclprob_min_tl
      }{
        \l__problems_prob_min_tl
      }
    }
  }
}
```

(*End definition for* `\record@problem`. *This function is documented on page* **??**.)

`\@problem` This macro acts on a problem's record in the `*.aux` file. It does not have any functionality here, but can be redefined elsewhere (e.g. in the `assignment` package).

```
\def\@problem#1#2#3{}
```

(*End definition for* `\@problem`. *This function is documented on page* **??**.)

`solution` The `solution` environment is similar to the `problem` environment, only that it is independent of the boxed mode. It also has it's own keys that we need to define first.

```
\keys_define:nn { problem / solution }{
  id            .str_set_x:N  = \l__problems_solution_id_str ,
  for           .tl_set:N     = \l__problems_solution_for_tl ,
  height        .dim_set:N    = \l__problems_solution_height_dim ,
  creators      .clist_set:N  = \l__problems_solution_creators_clist ,
  contributors  .clist_set:N  = \l__problems_solution_contributors_clist ,
  srccite       .tl_set:N     = \l__problems_solution_srccite_tl
}
\cs_new_protected:Nn \__problems_solution_args:n {
  \str_clear:N \l__problems_solution_id_str
  \tl_clear:N \l__problems_solution_for_tl
  \tl_clear:N \l__problems_solution_srccite_tl
  \clist_clear:N \l__problems_solution_creators_clist
  \clist_clear:N \l__problems_solution_contributors_clist
  \dim_zero:N \l__problems_solution_height_dim
  \keys_set:nn { problem / solution }{ #1 }
}
```

the next step is to define a helper macro that does what is needed to start a solution.

```
\newcommand\@startsolution[1][]{
  \__problems_solution_args:n { #1 }
  \@in@omtexttrue% we are in a statement.
  \bool_if:NF \c__problems_boxed_bool { \hrule }
  \smallskip\noindent
  {\textbf\prob@solution@kw :\enspace}
  \begin{small}
  \def\current@section@level{\prob@solution@kw}
  \ignorespacesandpars
}
```

`\startsolutions` for the `\startsolutions` macro we use the `\specialcomment` macro from the `comment` package. Note that we use the `\@startsolution` macro in the start codes, that parses the optional argument.

```
\newcommand\startsolutions{
  \specialcomment{solution}{\@startsolution}{
    \bool_if:NF \c__problems_boxed_bool {
      \hrule\medskip
    }
    \end{small}%
  }
  \bool_if:NT \c__problems_boxed_bool {
    \surroundwithmdframed{solution}
  }
}
```

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

`\stopsolutions`

```
\newcommand\stopsolutions{\excludecomment{solution}}
```

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

so it only remains to start/stop solutions depending on what option was specified.

```
\bool_if:NTF \c__problems_solutions_bool {
  \startsolutions
}{
  \stopsolutions
}
```

`exnote`

```
\bool_if:NTF \c__problems_notes_bool {
  \newenvironment{exnote}[1][]{
    \par\smallskip\hrule\smallskip
    \noindent\textbf{\prob@note@kw : }\small
  }{
    \smallskip\hrule
  }
}{
  \excludecomment{exnote}
}
```

`hint`

```
\bool_if:NTF \c__problems_notes_bool {
  \newenvironment{hint}[1][]{
    \par\smallskip\hrule\smallskip
    \noindent\textbf{\prob@hint@kw :~ }\small
  }{
    \smallskip\hrule
  }
  \newenvironment{exhint}[1][]{
    \par\smallskip\hrule\smallskip
    \noindent\textbf{\prob@hint@kw :~ }\small
  }{
    \smallskip\hrule
  }
}{
  \excludecomment{hint}
  \excludecomment{exhint}
}
```

`gnote`

```
\bool_if:NTF \c__problems_notes_bool {
  \newenvironment{gnote}[1][]{
    \par\smallskip\hrule\smallskip
    \noindent\textbf{\prob@gnote@kw : }\small
  }{
    \smallskip\hrule
  }
}{
  \excludecomment{gnote}
}
```

33.3 Multiple Choice Blocks

EdN:20

mcb [20]

```
\newenvironment{mcb}{
  \begin{enumerate}
}{
  \end{enumerate}
}
```

we define the keys for the mcc macro

```
\cs_new_protected:Nn \__problems_do_yes_param:Nn {
  \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
    \bool_set_true:N #1
  }{
    \bool_set_false:N #1
  }
}
\keys_define:nn { problem / mcc }{
  id        .str_set_x:N  = \l__problems_mcc_id_str ,
  feedback  .tl_set:N     = \l__problems_mcc_feedback_tl ,
  T         .default:n    = { true } ,
  T         .bool_set:N   = \l__problems_mcc_t_bool ,
  F         .default:n    = { true } ,
  F         .bool_set:N   = \l__problems_mcc_f_bool ,
  Ttext     .code:n       = {
    \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
  } ,
  Ftext     .code:n       = {
    \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
  }
}
\cs_new_protected:Nn \l__problems_mcc_args:n {
  \str_clear:N \l__problems_mcc_id_str
  \tl_clear:N \l__problems_mcc_feedback_tl
  \bool_set_true:N \l__problems_mcc_t_bool
  \bool_set_true:N \l__problems_mcc_f_bool
  \bool_set_true:N \l__problems_mcc_Ttext_bool
  \bool_set_false:N \l__problems_mcc_Ftext_bool
  \keys_set:nn { problem / mcc }{ #1 }
}
```

\mcc

```
\newcommand\mcc[2][]{
  \l__problems_mcc_args:n{ #1 }
  \item #2
  \bool_if:NT \c__problems_solutions_bool {
    \\
    \bool_if:NT \l__problems_mcc_t_bool {
      % TODO!
      % \ifcsstring{mcc@T}{T}{}{\mcc@Ttext}%
    }
    \bool_if:NT \l__problems_mcc_f_bool {
```

[20]EdNote: MK: maybe import something better here from a dedicated MC package

```
      % TODO!
      % \ifcsstring{mcc@F}{F}{}{\mcc@Ftext}%
    }
    \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
      !
    }{
      \l__problems_mcc_feedback_tl
    }
  }
} %solutions
```

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

33.4 Including Problems

`\includeproblem` The `\includeproblem` command is essentially a glorified `\input` statement, it sets some internal macros first that overwrite the local points. Importantly, it resets the `inclprob` keys after the input.

```

\keys_define:nn{ problem / inclproblem }{
 % id      .str_set_x:N  = \l__problems_inclprob_id_str,
  pts     .tl_set:N     = \l__problems_inclprob_pts_tl,
  min     .tl_set:N     = \l__problems_inclprob_min_tl,
  title   .tl_set:N     = \l__problems_inclprob_title_tl,
  refnum  .int_set:N    = \l__problems_inclprob_refnum_int,
  mhrepos .str_set_x:N  = \l__problems_inclprob_mhrepos_str
}
\cs_new_protected:Nn \__problems_inclprob_args:n {
%  \str_clear:N \l__problems_prob_id_str
  \tl_clear:N \l__problems_inclprob_pts_tl
  \tl_clear:N \l__problems_inclprob_min_tl
  \tl_clear:N \l__problems_inclprob_title_tl
  \int_zero_new:N \l__problems_inclprob_refnum_int
  \str_clear:N \l__problems_inclprob_mhrepos_str
  \keys_set:nn { problem / inclproblem }{ #1 }
  \tl_if_empty:NT \l__problems_inclprob_pts_tl {
    \let\l__problems_inclprob_pts_tl\undefined
  }
  \tl_if_empty:NT \l__problems_inclprob_min_tl {
    \let\l__problems_inclprob_min_tl\undefined
  }
  \tl_if_empty:NT \l__problems_inclprob_title_tl {
    \let\l__problems_inclprob_title_tl\undefined
  }
  \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
    \let\l__problems_inclprob_refnum_int\undefined
  }
}

\cs_new_protected:Nn \__problems_inclprob_clear: {
%  \str_clear:N \l__problems_prob_id_str
  \let\l__problems_inclprob_pts_tl\undefined
  \let\l__problems_inclprob_min_tl\undefined
  \let\l__problems_inclprob_title_tl\undefined
  \let\l__problems_inclprob_refnum_int\undefined
  \let\l__problems_inclprob_mhrepos_str\undefined
}

\newcommand\includeproblem[2][]{
  \__problems_inclprob_args:n{ #1 }
  \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
    \input{#2}
  }{
    \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
      \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
    }
  }
  \__problems_inclprob_clear:
}
```

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

33.5 Reporting Metadata

For messages it is OK to have them in English as the whole documentation is, and we can therefore assume authors can deal with it.

```
\AddToHook{enddocument}{
  \bool_if:NT \c__problems_pts_bool {
    \message{Total:~\arabic{pts}~points}
  }
  \bool_if:NT \c__problems_min_bool {
    \message{Total:~\arabic{min}~minutes}
  }
}
```

The margin pars are reader-visible, so we need to translate

```
\def\pts#1{
  \bool_if:NT \c__problems_pts_bool {
    \marginpar{#1~\prob@pt@kw}
  }
}
\def\min#1{
  \bool_if:NT \c__problems_min_bool {
    \marginpar{#1~\prob@min@kw}
  }
}
```

`\show@pts` The `\show@pts` shows the points: if no points are given from the outside and also no points are given locally do nothing, else show and add. If there are outside points then we show them in the margin.

```
\newcounter{pts}
\def\show@pts{
  \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
    \bool_if:NT \c__problems_pts_bool {
      \marginpar{\l__problems_inclprob_pts_tl;\prob@pt@kw\smallskip}
      \addtocounter{pts}{\l__problems_inclprob_pts_tl}
    }
  }{
    \tl_if_exist:NT \l__problems_prob_pts_tl {
      \bool_if:NT \c__problems_pts_bool {
        \marginpar{\l__problems_prob_pts_tl;\prob@pt@kw\smallskip}
        \addtocounter{pts}{\l__problems_prob_pts_tl}
      }
    }
  }
}
```

(*End definition for* `\show@pts`. *This function is documented on page* **??**.)

and now the same for the minutes

`\show@min`

```
\newcounter{min}
\def\show@min{
  \tl_if_exist:NTF \l__problems_inclprob_min_tl {
    \bool_if:NT \c__problems_min_bool {
      \marginpar{\l__problems_inclprob_pts_tl;min}
      \addtocounter{min}{\l__problems_inclprob_min_tl}
    }
  }{
    \tl_if_exist:NT \l__problems_prob_min_tl {
      \bool_if:NT \c__problems_min_bool {
        \marginpar{\l__problems_prob_min_tl;min}
        \addtocounter{min}{\l__problems_prob_min_tl}
      }
    }
  }
}
⟨/package⟩
```

(*End definition for* `\show@min`. *This function is documented on page* **??**.)

Chapter 34

Implementation: The hwexam Class

The functionality is spread over the `hwexam` class and package. The class provides the `document` environment and pre-loads some convenience packages, whereas the package provides the concrete functionality.

34.1 Class Options

To initialize the `hwexam` class, we declare and process the necessary options by passing them to the respective packages and classes they come from.

```
⟨@@=hwexam⟩
⟨*cls⟩
\ProvidesExplClass{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
\RequirePackage{l3keys2e,expl-keystr-compat}
\DeclareOption*{
  \PassOptionsToClass{\CurrentOption}{omdoc}
  \PassOptionsToPackage{\CurrentOption}{stex}
  \PassOptionsToPackage{\CurrentOption}{hwexam}
  \PassOptionsToPackage{\CurrentOption}{tikzinput}
}
\ProcessOptions
```

We load `omdoc.cls`, and the desired packages. For the LaTeXML bindings, we make sure the right packages are loaded.

```
\LoadClass{omdoc}
\RequirePackage{stex}
\RequirePackage{hwexam}
\RequirePackage{tikzinput}
\RequirePackage{graphicx}
\RequirePackage{a4wide}
\RequirePackage{amssymb}
\RequirePackage{amstext}
\RequirePackage{amsmath}
```

Finally, we register another keyword for the `document` environment. We give a default assignment type to prevent errors

```
\newcommand\assig@default@type{\hwexam@assignment@kw}
\def\document@hwexamtype{\assig@default@type}
⟨@@=document_structure⟩
\keys_define:nn { document-structure / document }{
id .str_set_x:N = \c_document_structure_document_id_str,
hwexamtype .tl_set:N = \document@hwexamtype
}
⟨@@=hwexam⟩
⟨/cls⟩
```

Chapter 35

Implementation: The hwexam Package

35.1 Package Options

The first step is to declare (a few) package options that handle whether certain information is printed or not. Some come with their own conditionals that are set by the options, the rest is just passed on to the problems package.

```
⟨*package⟩
\ProvidesExplPackage{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
\RequirePackage{l3keys2e,expl-keystr-compat}

\newif\iftest\testfalse
\DeclareOption{test}{\testtrue}
\newif\ifmultiple\multiplefalse
\DeclareOption{multiple}{\multipletrue}
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
\ProcessOptions
```

Then we make sure that the necessary packages are loaded (in the right versions).

```
\RequirePackage{keyval}[1997/11/10]
\RequirePackage{problem}
```

\hwexam@*@kw For multilinguality, we define internal macros for keywords that can be specialized in *.ldf files.

```
\newcommand\hwexam@assignment@kw{Assignment}
\newcommand\hwexam@given@kw{Given}
\newcommand\hwexam@due@kw{Due}
\newcommand\hwexam@testemptypage@kw{This page was intentionally left blank for extra
  space}%
\newcommand\correction@probs@kw{prob.}%
\newcommand\correction@pts@kw{total}%
\newcommand\correction@reached@kw{reached}%
\newcommand\correction@sum@kw{Sum}%
\newcommand\correction@grade@kw{grade}%
\newcommand\correction@forgrading@kw{To be used for grading, do not write here}
```

(*End definition for* `\hwexam@*@kw`. *This function is documented on page* **??**.)

For the other languages, we set up triggers

```
\@ifpackageloaded{babel}{}{\RequirePackage[base]{babel}}

\clist_set:Nx \l_tmpa_clist {\bbl@loaded}
\clist_if_in:NnT \l_tmpa_clist {ngerman}{
  \input{hwexam-ngerman.ldf}
}
\clist_if_in:NnT \l_tmpa_clist {finnish}{
  \input{hwexam-finnish.ldf}
}
\clist_if_in:NnT \l_tmpa_clist {french}{
  \input{hwexam-french.ldf}
}
\clist_if_in:NnT \l_tmpa_clist {russian}{
  \input{hwexam-russian.ldf}
}
```

35.2 Assignments

Then we set up a counter for problems and make the problem counter inherited from `problem.sty` depend on it. Furthermore, we specialize the `\prob@label` macro to take the assignment counter into account.

```
\newcounter{assignment}
\numberproblemsin{assignment}
\renewcommand\prob@label[1]{\arabic{assignment}.#1}
```

We will prepare the keyval support for the `assignment` environment.

```
\keys_define:nn { hwexam / assignment } {
id  .str_set_x:N = \l__hwexam_assign_id_str,
number  .int_set:N  = \l__hwexam_assign_number_int,
title  .tl_set:N  = \l__hwexam_assign_title_tl,
type  .tl_set:N  = \l__hwexam_assign_type_tl,
given .tl_set:N  = \l__hwexam_assign_given_tl,
due .tl_set:N  = \l__hwexam_assign_due_tl,
loadmodules .code:n  = {
\bool_set_true:N \l__hwexam_assign_loadmodules_bool
}
}
\cs_new_protected:Nn \__hwexam_assignment_args:n {
\str_clear:N \l__hwexam_assign_id_str
\int_set:Nn \l__hwexam_assign_number_int {-1}
\tl_clear:N \l__hwexam_assign_title_tl
\tl_clear:N \l__hwexam_assign_type_tl
\tl_clear:N \l__hwexam_assign_given_tl
\tl_clear:N \l__hwexam_assign_due_tl
\bool_set_false:N \l__hwexam_assign_loadmodules_bool
\keys_set:nn { hwexam / assignment }{ #1 }
}
```

The next three macros are intermediate functions that handle the case gracefully, where the respective token registers are undefined.

The `\given@due` macro prints information about the given and due status of the assignment. Its arguments specify the brackets.

```
\newcommand\given@due[2]{
\bool_lazy_all:nF {
{\tl_if_empty_p:V \l__hwexam_inclassign_given_tl}
{\tl_if_empty_p:V \l__hwexam_assign_given_tl}
{\tl_if_empty_p:V \l__hwexam_inclassign_due_tl}
{\tl_if_empty_p:V \l__hwexam_assign_due_tl}
}{ #1 }

\tl_if_empty:NTF \l__hwexam_inclassign_given_tl {
\tl_if_empty:NF \l__hwexam_assign_given_tl {
\hwexam@given@kw\xspace\l__hwexam_assign_given_tl
}
}{
\hwexam@given@kw\xspace\l__hwexam_inclassign_given_tl
}

\bool_lazy_or:nnF {
\bool_lazy_and_p:nn {
\tl_if_empty_p:V \l__hwexam_inclassign_due_tl
}{
\tl_if_empty_p:V \l__hwexam_assign_due_tl
}
}{
\bool_lazy_and_p:nn {
\tl_if_empty_p:V \l__hwexam_inclassign_due_tl
}{
\tl_if_empty_p:V \l__hwexam_assign_due_tl
}
}{ ,~ }

\tl_if_empty:NTF \l__hwexam_inclassign_due_tl {
\tl_if_empty:NF \l__hwexam_assign_due_tl {
\hwexam@due@kw\xspace \l__hwexam_assign_due_tl
}
}{
\hwexam@due@kw\xspace \l__hwexam_inclassign_due_tl
}

\bool_lazy_all:nF {
{ \tl_if_empty_p:V \l__hwexam_inclassign_given_tl }
{ \tl_if_empty_p:V \l__hwexam_assign_given_tl }
{ \tl_if_empty_p:V \l__hwexam_inclassign_due_tl }
{ \tl_if_empty_p:V \l__hwexam_assign_due_tl }
}{ #2 }
}
```

`\assignment@title` This macro prints the title of an assignment, the local title is overwritten, if there is one from the `\inputassignment`. `\assignment@title` takes three arguments the first is the fallback when no title is given at all, the second and third go around the title, if one is given.

```
\newcommand\assignment@title[3]{
```

```
\tl_if_empty:NTF \l__hwexam_inclassign_title_tl {
\tl_if_empty:NTF \l__hwexam_assign_title_tl {
#1
}{
#2\l__hwexam_assign_title_tl#3
}
}{
#2\l__hwexam_inclassign_title_tl#3
}
}
```

(*End definition for* `\assignment@title`. *This function is documented on page* **??**.)

`\assignment@number` Like `\assignment@title` only for the number, and no around part.

```
\newcommand\assignment@number{
\int_compare:nNnTF \l__hwexam_inclassign_number_int = {-1} {
\int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
\int_use:N \l__hwexam_assign_number_int
}
}{
\int_use:N \l__hwexam_inclassign_number_int
}
}
```

(*End definition for* `\assignment@number`. *This function is documented on page* **??**.)

With them, we can define the central `assignment` environment. This has two forms (separated by `\ifmultiple`) in one we make a title block for an assignment sheet, and in the other we make a section heading and add it to the table of contents. We first define an assignment counter

`assignment` For the `assignment` environment we delegate the work to the `@assignment` environment that depends on whether `multiple` option is given.

```
\newenvironment{assignment}[1][]{
\__hwexam_assignment_args:n { #1 }
%\sref@target
\let\__hwexamnum\l__hwexam_assign_number_int
\int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
\stepcounter{assignment}
}{
\setcounter{assignment}{\int_use:N\__hwexamnum}
}
\setcounter{problem}{0}
\def\current@section@level{\document@hwexamtype}
%\sref@label@id{\document@hwexamtype \thesection}
\begin{@assignment}
}{
\end{@assignment}
}
```

In the multi-assignment case we just use the `omdoc` environment for suitable sectioning.

```
\def\__hwexamasstitle{
\protect\document@hwexamtype~\arabic{assignment}
\assignment@title{}{\;(}{)\;} -- \given@due{}{}
}
```

```
\ifmultiple
\newenvironment{@assignment}{
\bool_if:NTF \l__hwexam_assign_loadmodules_bool {
\begin{omgroup}[loadmodules]{\__hwexamasstitle}
}{
\begin{omgroup}{\__hwexamasstitle}
}
}{
\end{omgroup}
}
```

for the single-page case we make a title block from the same components.

```
\else
\newenvironment{@assignment}{
\begin{center}\bf
\Large\@title\strut\\
\document@hwexamtype~\arabic{assignment}\assignment@title{\;}{:\;}{\\}
\large\given@due{--\;}{\;--}
\end{center}
}{}
\fi% multiple
```

35.3 Including Assignments

`\in*assignment` This macro is essentially a glorified `\include` statement, it just sets some internal macros first that overwrite the local points Importantly, it resets the `inclassig` keys after the input.

```
\keys_define:nn { hwexam / inclassignment } {
%id  .str_set_x:N = \l__hwexam_assign_id_str,
number  .int_set:N  = \l__hwexam_inclassign_number_int,
title  .tl_set:N  = \l__hwexam_inclassign_title_tl,
type  .tl_set:N  = \l__hwexam_inclassign_type_tl,
given .tl_set:N  = \l__hwexam_inclassign_given_tl,
due .tl_set:N  = \l__hwexam_inclassign_due_tl,
mhrepos  .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
}
\cs_new_protected:Nn \__hwexam_inclassignment_args:n {
\int_set:Nn \l__hwexam_inclassign_number_int {-1}
\tl_clear:N \l__hwexam_inclassign_title_tl
\tl_clear:N \l__hwexam_inclassign_type_tl
\tl_clear:N \l__hwexam_inclassign_given_tl
\tl_clear:N \l__hwexam_inclassign_due_tl
\str_clear:N \l__hwexam_inclassign_mhrepos_str
\keys_set:nn { hwexam / inclassignment }{ #1 }
}
\__hwexam_inclassignment_args:n {}

\newcommand\inputassignment[2][]{
\__hwexam_inclassignment_args:n { #1 }
\str_if_empty:NTF \l__hwexam_inclassign_mhrepos_str {
\input{#2}
}{
\stex_in_repository:nn{\l__hwexam_inclassign_mhrepos_str}{
```

```
\input{\mhpath{\l__hwexam_inclassign_mhrepos_str}{#2}}
}
}
\__hwexam_inclassignment_args:n {}
}
\newcommand\includeassignment[2][]{
\newpage
\inputassignment[#1]{#2}
}
```

(*End definition for* `\in*assignment`. *This function is documented on page* **??**.)

35.4 Typesetting Exams

`\quizheading`

```
\ExplSyntaxOff
\newcommand\quizheading[1]{%
\def\@tas{#1}%
\large\noindent NAME: \hspace{8cm}  MAILBOX:\\[2ex]%
\ifx\@tas\@empty\else%
\noindent TA:~\@for\@I:=\@tas\do{{\Large$\Box$}\@I\hspace*{1em}}\\[2ex]%
\fi%
}
\ExplSyntaxOn
```

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

`\testheading`

```
\keys_define:nn { hwexam / testheading } {
min  .tl_set:N  = \l__hwexam_testheading_min_tl,
duration .tl_set:N  = \__hwexam_testheading_duration_tl,
reqpts .tl_set:N  = \l__hwexam_testheading_reqpts_tl
}
\cs_new_protected:Nn \__hwexam_testheading_args:n {
\tl_clear:N \l__hwexam_testheading_min_tl
\tl_clear:N \l__hwexam_testheading_duration_tl
\tl_clear:N \l__hwexam_testheading_reqpts_tl
\keys_set:nn { hwexam / testheading }{ #1 }
}
\newenvironment{testheading}[1][]{
\__hwexam_testheading_args:n{ #1 }
\noindent\large{}Name:~\hfill
Matriculation Number:\hspace*{2cm}\strut\\[1ex]
\begin{center}
\Large\textbf{\@title}\\[1ex]
\large\@date\\[3ex]
\end{center}
\textbf{You~have~
\tl_if_empty:NTF \l__hwexam_testheading_duration_tl {
\l__hwexam_testheading_min_tl~minutes
}{
\l__hwexam_testheading_duration_tl
}~
(sharp)~for~the~test
};\\
Write~the~solutions~to~the~sheet.
\par\noindent
\newcount\check@time\check@time=\l__hwexam_testheading_min_tl
\advance\check@time by -\theassignment@totalmin
The~estimated~time~for~solving~this~exam~is~
{\theassignment@totalmin}~minutes,~
leaving~you~{\the\check@time}~minutes~for~revising~
your~exam.

\par\noindent
\newcount\bonus@pts\bonus@pts=\theassignment@totalpts
\advance\bonus@pts by -\l__hwexam_testheading_reqpts_tl
You~can~reach~{\theassignment@totalpts}~points~if~you~
solve~all~problems.~You~will~only~need~
{\l__hwexam_testheading_reqpts_tl}~points~for~a~perfect~score,~
i.e.\ {\the\bonus@pts}~points~are~bonus~points.
\vfill
\begin{center}
   {
\Large\em You~have~ample~time,~so~take~it~slow~
   and~avoid~rushing~to~mistakes!\\[2ex]
   Different~problems~test~different~skills~and~
knowledge,~so~do~not~get~stuck~on~one~problem.
}
\vfill\par\resizebox{\textwidth}{!}{\correction@table}\\[3ex]
\end{center}
}{
\newpage
}
```

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

`\testspace`

```
\newcommand\testspace[1]{\iftest\vspace*{#1}\fi}
```

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

`\testnewpage`

```
\newcommand\testnewpage{\iftest\newpage\fi}
```

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

`\testemptypage`

```
\newcommand\testemptypage[1][]{\iftest\begin{center}\hwexam@testemptypage@kw\end{center}\vfi
```

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

`\@problem` This macro acts on a problem's record in the `*.aux` file. Here we redefine it (it was defined to do nothing in `problem.sty`) to generate the correction table.

```
⟨@@=problems⟩
\renewcommand\@problem[3]{
\stepcounter{assignment@probs}
\def\__problemspts{#2}
\ifx\__problemspts\@empty\else
\addtocounter{assignment@totalpts}{#2}
\fi
\def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\f
\xdef\correction@probs{\correction@probs & #1}%
\xdef\correction@pts{\correction@pts & #2}
\xdef\correction@reached{\correction@reached &}
}
⟨@@=hwexam⟩
```

(*End definition for* `\@problem`. *This function is documented on page* **??**.)

`\correction@table` This macro generates the correction table

```
\newcounter{assignment@probs}
\newcounter{assignment@totalpts}
\newcounter{assignment@totalmin}
\def\correction@probs{\correction@probs@kw}%
\def\correction@pts{\correction@pts@kw}%
\def\correction@reached{\correction@reached@kw}%
\def\after@correction@table{}%
\stepcounter{assignment@probs}
\newcommand\correction@table{
\resizebox{\textwidth}{!}{%
\begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
&\multicolumn{\theassignment@probs}{c||}%|
{\footnotesize\correction@forgrading@kw} &\\\hline
\correction@probs & \correction@sum@kw & \correction@grade@kw\\\hline
\correction@pts &\theassignment@totalpts & \\\hline
\correction@reached & & \\[.7cm]\hline
\end{tabular}}
\ifx\after@correction@table\@empty\else\strut\par\noindent\after@correction@table\fi}
⟨/package⟩
```

(*End definition for* `\correction@table`. *This function is documented on page* **??**.)

35.5 Leftovers

at some point, we may want to reactivate the logos font, then we use

```
here we define the logos that characterize the assignment
\font\bierfont=../assignments/bierglas
\font\denkerfont=../assignments/denker
\font\uhrfont=../assignments/uhr
\font\warnschildfont=../assignments/achtung

\newcommand\bierglas{{\bierfont\char65}}
\newcommand\denker{{\denkerfont\char65}}
\newcommand\uhr{{\uhrfont\char65}}
\newcommand\warnschild{{\warnschildfont\char 65}}
\newcommand\hardA{\warnschild}
\newcommand\longA{\uhr}
\newcommand\thinkA{\denker}
\newcommand\discussA{\bierglas}
```