

# The $\text{\TeX}$ 3 Package \*

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

2022-01-18

## **Abstract**

TODO

---

\*Version 3.0 (last revised 2022-01-18)

# Contents

<b>I</b>	<b>Manual</b>	<b>1</b>
<b>1</b>	<b>Stuff</b>	<b>2</b>
1.1	Modules . . . . .	2
1.1.1	Semantic Macros and Notations . . . . .	2
	Other Argument Types . . . . .	4
	Precedences . . . . .	6
1.1.2	Archives and Imports . . . . .	6
	Namespaces . . . . .	6
	Paths in Import-Statements . . . . .	7
<b>II</b>	<b>Documentation</b>	<b>8</b>
<b>2</b>	<b>sTeX-Basics</b>	<b>9</b>
2.1	Macros and Environments . . . . .	9
<b>3</b>	<b>sTeX-MathHub</b>	<b>11</b>
3.1	Macros and Environments . . . . .	11
3.1.1	Files, Paths, URIs . . . . .	11
3.1.2	MathHub Archives . . . . .	12
<b>4</b>	<b>sTeX-References</b>	<b>14</b>
4.1	Macros and Environments . . . . .	14
<b>5</b>	<b>sTeX-Modules</b>	<b>15</b>
5.1	Macros and Environments . . . . .	15
5.1.1	The module-environment . . . . .	17
<b>6</b>	<b>sTeX-Module Inheritance</b>	<b>20</b>
6.1	Macros and Environments . . . . .	20
6.1.1	SMS Mode . . . . .	20
6.1.2	Imports and Inheritance . . . . .	21
<b>7</b>	<b>sTeX-Symbols</b>	<b>24</b>
7.1	Macros and Environments . . . . .	24
<b>8</b>	<b>sTeX-Terms</b>	<b>27</b>
8.1	Macros and Environments . . . . .	27
<b>9</b>	<b>sTeX-Structural Features</b>	<b>30</b>
9.1	Macros and Environments . . . . .	30
9.1.1	Structures . . . . .	30
<b>10</b>	<b>sTeX-Statements</b>	<b>31</b>
10.1	Macros and Environments . . . . .	31

<b>11</b>	<b>STeX-Proofs: Structural Markup for Proofs</b>	<b>32</b>
11.1	Introduction . . . . .	34
11.2	The User Interface . . . . .	35
11.2.1	Package Options . . . . .	35
11.2.2	Proofs and Proof steps . . . . .	35
11.2.3	Justifications . . . . .	35
11.2.4	Proof Structure . . . . .	36
11.2.5	Proof End Markers . . . . .	37
11.2.6	Configuration of the Presentation . . . . .	37
11.3	Limitations . . . . .	37
<b>12</b>	<b>STeX-Metatheory</b>	<b>39</b>
12.1	Symbols . . . . .	39
<b>III</b>	<b>Extensions</b>	<b>40</b>
<b>13</b>	<b>Tikzinput</b>	<b>41</b>
13.1	Macros and Environments . . . . .	41
<b>14</b>	<b>document-structure.sty: Semantic Markup for Open Mathematical Documents in L<sup>A</sup>T<sub>E</sub>X</b>	<b>42</b>
14.1	Introduction . . . . .	42
14.2	The User Interface . . . . .	43
14.2.1	Package and Class Options . . . . .	43
14.2.2	Document Structure . . . . .	43
14.2.3	Ignoring Inputs . . . . .	44
14.2.4	Structure Sharing . . . . .	45
14.2.5	Global Variables . . . . .	45
14.2.6	Colors . . . . .	46
14.3	Limitations . . . . .	46
<b>15</b>	<b>Slides and Course Notes</b>	<b>47</b>
15.1	Introduction . . . . .	47
15.2	The User Interface . . . . .	47
15.2.1	Package Options . . . . .	47
15.2.2	Notes and Slides . . . . .	48
15.2.3	Header and Footer Lines of the Slides . . . . .	49
15.2.4	Frame Images . . . . .	49
15.2.5	Colors and Highlighting . . . . .	50
15.2.6	Front Matter, Titles, etc. . . . .	50
15.2.7	Excursions . . . . .	50
15.2.8	Miscellaneous . . . . .	50
15.3	Limitations . . . . .	50

<b>16</b>	<b>problem.sty: An Infrastructure for formatting Problems</b>	<b>51</b>
16.1	Introduction . . . . .	51
16.2	The User Interface . . . . .	51
16.2.1	Package Options . . . . .	51
16.2.2	Problems and Solutions . . . . .	52
16.2.3	Multiple Choice Blocks . . . . .	53
16.2.4	Including Problems . . . . .	53
16.2.5	Reporting Metadata . . . . .	53
16.3	Limitations . . . . .	53
<b>17</b>	<b>hwexam.sty/cls: An Infrastructure for formatting Assignments and Exams</b>	<b>55</b>
17.1	Introduction . . . . .	56
17.2	The User Interface . . . . .	56
17.2.1	Package and Class Options . . . . .	56
17.2.2	Assignments . . . . .	56
17.2.3	Typesetting Exams . . . . .	56
17.2.4	Including Assignments . . . . .	57
17.3	Limitations . . . . .	57
<b>IV</b>	<b>Implementation</b>	<b>59</b>
<b>18</b>	<b>gTeX-Basics Implementation</b>	<b>60</b>
18.1	The gTeXDocument Class . . . . .	60
18.2	Preliminaries . . . . .	60
18.3	Messages and logging . . . . .	61
18.4	Persistence . . . . .	62
18.5	HTML Annotations . . . . .	62
18.6	Languages . . . . .	65
18.7	Activating/Deactivating Macros . . . . .	66
<b>19</b>	<b>gTeX-MathHub Implementation</b>	<b>68</b>
19.1	Generic Path Handling . . . . .	68
19.2	PWD and kpsewhich . . . . .	70
19.3	File Hooks and Tracking . . . . .	71
19.4	MathHub Repositories . . . . .	72
<b>20</b>	<b>gTeX-References Implementation</b>	<b>78</b>
20.1	Document URIs and URLs . . . . .	78
20.2	Setting Reference Targets . . . . .	80
20.3	Using References . . . . .	81
<b>21</b>	<b>gTeX-Modules Implementation</b>	<b>83</b>
21.1	The module environment . . . . .	86
21.2	Invoking modules . . . . .	91
<b>22</b>	<b>gTeX-Module Inheritance Implementation</b>	<b>93</b>
22.1	SMS Mode . . . . .	93
22.2	Inheritance . . . . .	97

<b>23</b>	<b>STEX-Symbols Implementation</b>	<b>102</b>
23.1	Symbol Declarations . . . . .	102
23.2	Notations . . . . .	108
<b>24</b>	<b>STEX-Terms Implementation</b>	<b>116</b>
24.1	Symbol Invocations . . . . .	116
24.2	Terms . . . . .	119
24.3	Notation Components . . . . .	125
<b>25</b>	<b>STEX-Structural Features Implementation</b>	<b>128</b>
25.1	The feature environment . . . . .	128
25.2	Features . . . . .	130
<b>26</b>	<b>STEX-Statements Implementation</b>	<b>135</b>
26.1	Definitions . . . . .	136
26.2	Assertions . . . . .	138
26.3	Examples . . . . .	140
26.4	OMText . . . . .	141
<b>27</b>	<b>The Implementation</b>	<b>142</b>
27.1	Package Options . . . . .	142
27.2	Proofs . . . . .	142
27.3	Justifications . . . . .	148
<b>28</b>	<b>STEX-Others Implementation</b>	<b>150</b>
<b>29</b>	<b>STEX-Metatheory Implementation</b>	<b>151</b>
<b>30</b>	<b>Tikzinput Implementation</b>	<b>154</b>
<b>31</b>	<b>document-structure.sty Implementation</b>	<b>156</b>
31.1	The OMDoc Class . . . . .	156
31.2	Class Options . . . . .	156
31.3	Beefing up the document environment . . . . .	157
31.4	Implementation: OMDoc Package . . . . .	157
31.5	Package Options . . . . .	157
31.6	Document Structure . . . . .	159
31.7	Front and Backmatter . . . . .	162
31.8	Global Variables . . . . .	164
<b>32</b>	<b>MiKoSlides – Implementation</b>	<b>165</b>
32.1	Class and Package Options . . . . .	165
32.2	Notes and Slides . . . . .	167
32.3	Header and Footer Lines . . . . .	171
32.4	Frame Images . . . . .	172
32.5	Colors and Highlighting . . . . .	173
32.6	Sectioning . . . . .	174
32.7	Excursions . . . . .	176

<b>33 The Implementation</b>	<b>178</b>
33.1 Package Options . . . . .	178
33.2 Problems and Solutions . . . . .	179
33.3 Multiple Choice Blocks . . . . .	184
33.4 Including Problems . . . . .	185
33.5 Reporting Metadata . . . . .	186
<b>34 Implementation: The hwexam Class</b>	<b>188</b>
34.1 Class Options . . . . .	188
<b>35 Implementation: The hwexam Package</b>	<b>190</b>
35.1 Package Options . . . . .	190
35.2 Assignments . . . . .	191
35.3 Including Assignments . . . . .	194
35.4 Typesetting Exams . . . . .	195
35.5 Leftovers . . . . .	197

**Part I**  
**Manual**

# Chapter 1

## Stuff

### 1.1 Modules

---

`\sTeX`  
`\stex`

---

Both print this  $\text{\TeX}$  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<sup>1</sup>.

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

### Example 3

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

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

.

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

### Example 4

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

Multiplying again by  $b$  yields...

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

### Example 5

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

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

<sup>1</sup>EdNOTE: TODO

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

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

### Example 6

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

The operator  $\textcolor{teal}{+}$  adds two elements, as in  $a\textcolor{teal}{+}b$ .

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

### Example 7

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

$\textcolor{teal}{\textit{Multiplication}}$  (denoted by  $\cdot$ ) is defined by...

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

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

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

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

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

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

### Example 8

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

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

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

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

### Example 9

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

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

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

2 3

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

<sup>3</sup>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.

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

Notations for symbols of arity 0 have a default precedence of `\infprec`, i.e. by default, parentheses are never inserted around constants. Notations for symbols with arity  $> 0$  have a default operator precedence of 0. If no argument precedences are explicitly provided, then by default they are equal to the operator precedence.

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

For example:

### Example 10

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

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

## 1.1.2 Archives and Imports

### Namespaces

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

Unfortunately,  $\TeX$  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  $\TeX$  can resolve them accordingly. Largely, users need not concern themselves with namespaces at all, but for completeness sake, we describe how they are constructed:

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

In other words: outside of archives, the namespace corresponds to the file URI with the filename dropped iff it is equal to the module name, and ignoring the (optional) language suffix<sup>1</sup>.

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, exemplary `\importmodule`:

- `\importmodule{Foo}` outside of an archive refers to module `Foo` in the current namespace. Consequently, `Foo` must have been declared earlier in the same document or, if not, in a file `Foo[.<lang>].tex` in the same directory.
- The same statement *within* an archive refers to either the module `Foo` declared earlier in the same document, or otherwise to the module `Foo` in the archive's top-level namespace. In the latter case, it has to be declared in a file `Foo[.<lang>].tex` directly in the archive's `source`-folder.
- Similarly, in `\importmodule{some/path?Foo}` the path `some/path` refers to either the sub-directory and relative namespace path of the current directory and namespace outside of an archive, or relative to the current archive's top-level namespace and `source`-folder, respectively.

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

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

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

---

<sup>1</sup>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** ( $\langle log-prefix \rangle *$ ) Logs debugging information with the given prefixes to the terminal, or all if **all** is given.

**showmods** ( $\langle boolean \rangle$ ) Shows explicit module information at the document margins.

**lang** ( $\langle language \rangle *$ ) Languages to load with the **babel** package.

**mathhub** ( $\langle directory \rangle$ ) MathHub folder to search for repositories.

**sms** ( $\langle boolean \rangle$ ) use *persisted* mode (see ???).

**image** ( $\langle boolean \rangle$ ) passed on to tikzinput.

### 2.1 Macros and Environments

---

<code>\sTeX</code>	Both print this sTeX logo.
<code>\stex</code>	

---

---

<code>\stex_debug:nn</code>	<code>\stex_debug:nn {<math>\langle log-prefix \rangle</math>} {<math>\langle message \rangle</math>}</code>
-----------------------------	--

---

Logs  $\langle message \rangle$ , if the package option **debug** contains  $\langle log-prefix \rangle$ .

---

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

---

---

<code>\if@latexml</code>	L <sup>A</sup> T <sub>E</sub> X2e and L <sup>A</sup> T <sub>E</sub> X3 conditionals for L <sup>A</sup> T <sub>E</sub> XML.
<code>\latexml_if_p:</code>	
<code>\latexml_if:T</code>	
<code>\latexml_if:F</code>	
<code>\latexml_if:TF</code>	

---

We have four macros for annotating generated HTML (via L<sup>A</sup>T<sub>E</sub>XML or RusT<sub>E</sub>X) with attributes:

---

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

---

Annotates the HTML generated by  $\langle content \rangle$  with

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

---

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

---

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

---

<code>\stex_deactivate_macro:Nn</code>	<code>\stex_deactivate_macro:Nn⟨cs⟩{⟨environments⟩}</code>
<code>\stex_reactivate_macro:N</code>	

---

Makes the macro  $\langle cs \rangle$  throw an error, indicating that it is only allowed in the context of  $\langle environments \rangle$ .

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

---

<code>\MSC</code>	<code>\MSC{⟨msc⟩}</code>
-------------------	--------------------------

---

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

---

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

#### 3.1.1 Files, Paths, URIs

---

<code>\stex_path_from_string:Nn</code>	<code>\stex_path_from_string:Nn</code> $\langle path-variable \rangle$ $\{ \langle string \rangle \}$
<code>\stex_path_from_string:(NV cn cV)</code>	

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

---

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

---

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

---

<code>\stex_path_if_absolute_p:N</code>	$\star$
<code>\stex_path_if_absolute:NTF</code>	$\star$

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

---

<code>\c_stex_pwd_seq</code>	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 <code>\jobname</code> .
<code>\c_stex_pwd_str</code>	
<code>\c_stex_mainfile_seq</code>	
<code>\c_stex_mainfile_str</code>	

---

`\g_stex_currentfile_seq`

---

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

### Test 1

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

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

## 3.1.2 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).

<hr/> <hr/> <code>\stex_set_current_repository:n</code>	Sets the current repository to the one with the provided ID. calls <code>\__stex_mathhub_do_manifest:n</code> , so works whether this repository's MANIFEST.MF-file has already been read or not.
<hr/> <hr/> <code>\stex_require_repository:n</code>	Calls <code>\__stex_mathhub_do_manifest:n</code> iff the corresponding archive property list does not already exist, and adds a corresponding definition to the <code>.sms</code> -file.
<hr/> <hr/> <code>\stex_in_repository:nn</code>	<code>\stex_in_repository:nn{&lt;repository-name&gt;}{&lt;code&gt;}</code> Change the current repository to <code>{&lt;repository-name&gt;}</code> (or not, if <code>{&lt;repository-name&gt;}</code> is empty), and passes its ID on to <code>{&lt;code&gt;}</code> as #1. Switches back to the previous repository after executing <code>{&lt;code&gt;}</code> .
<hr/> <hr/> <code>\mhpath *</code>	<code>\mhpath{&lt;archive-ID&gt;}{&lt;filename&gt;}</code> Expands to the full path of file <code>&lt;filename&gt;</code> in repository <code>&lt;archive-ID&gt;</code> . Does not check whether the file or the repository exist.
<hr/> <hr/> <code>\inputref</code> <hr/> <code>\inputref:nn</code>	<code>\inputref[&lt;archive-ID&gt;]{&lt;filename&gt;}</code> <code>\inputs</code> the file <code>&lt;filename&gt;</code> in repository <code>&lt;archive-ID&gt;</code> .
<hr/> <hr/> <code>\libinput</code>	<code>\libinput{&lt;filename&gt;}</code> Inputs <code>&lt;filename&gt;.tex</code> from the <code>lib</code> folders in the current archive and the <code>meta-inf</code> -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 - `\inputs` are considered the same file).

---

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

---



---

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

---

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

---

```
\stex_add_to_current_module:n
\STEXexport
```

---

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

---

```
\stex_add_constant_to_current_module:n
```

---

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

---

```
\stex_add_import_to_current_module:n
```

---

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

---

```
\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}[\langle options \rangle]{\langle name \rangle}`  
 Opens a new module with name  $\langle name \rangle$ .  
 TODO document options.

---

`\stex_module_setup:nn`    `\stex_module_setup:nn{\langle params \rangle}{\langle name \rangle}`  
 Sets up a new module with name  $\langle name \rangle$  and optional parameters  $\langle params \rangle$ . 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}[\langle options \rangle]{\langle name \rangle}`  
 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}
```






---

`\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 T<sub>E</sub>X 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.  $\TeX$  determines the module and its containing file by passing its arguments on to `\stex_import_module_path:nn`.

### Test 8

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

**Module 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?Metatheory?collect, http://mathhub.info/sTeX?Metatheory?fromto, http://mathhub.info/sTeX?Metatheory?apply, http://mathhub.info/sTeX?Metatheory?collect, http://mathhub.info/sTeX?Metatheory?seqtype, http://mathhub.info/sTeX?Metatheory?sequence-index, http://mathhub.info/sTeX?Metatheory?aseqfromto, http://mathhub.info/sTeX?Metatheory?aseqfromtovia, 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

---

<code>\symdecl</code>	<code>\symdecl[⟨args⟩]{⟨macroname⟩}</code>
-----------------------	--

---

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  $\langle URI \rangle$  in the property list `\g_stex_symdecl_⟨URI⟩_prop` with fields:

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

### Test 11

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

```
Module 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  $\langle symbol \rangle$ , 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
```

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

<hr/> <hr/> <code>\STEXsymbol</code>	Uses <code>\stex_get_symbol:n</code> to find the symbol denoted by the first argument and passes the result on to <code>\stex_invoke_symbol:n</code>
<hr/> <hr/> <code>\symref</code>	<code>\symref{&lt;symbol&gt;}{&lt;text&gt;}</code> shortcut for <code>\STEXsymbol{&lt;symbol&gt;}! [ &lt;text&gt; ]</code>
<hr/> <hr/> <code>\stex_invoke_symbol:n</code>	Executes a semantic macro. Outside of math mode or if followed by <code>*</code> , it continues to <code>\stex_term_custom:nn</code> . In math mode, it uses the default or optionally provided notation of the associated symbol. If followed by <code>!</code> , it will invoke the symbol <i>itself</i> rather than its application (and continue to <code>\stex_term_custom:nn</code> ), i.e. it allows to refer to <code>\plus! [addition]</code> as an operation, rather than <code>\plus [addition of] {some} {terms}</code> .
<hr/> <hr/> <code>\_stex_term_math_oms:nnnn</code> <code>\_stex_term_math_oma:nnnn</code> <code>\_stex_term_math_omb:nnnn</code>	<code>&lt;URI&gt;&lt;fragment&gt;&lt;precedence&gt;&lt;body&gt;</code> Annotates <code>&lt;body&gt;</code> as an OMDOC-term (OMID, OMA or OMBIND, respectively) with head symbol <code>&lt;URI&gt;</code> , generated by the specific notation <code>&lt;fragment&gt;</code> with (upwards) operator precedence <code>&lt;precedence&gt;</code> . Inserts parentheses according to the current downwards precedence and operator precedence.
<hr/> <hr/> <code>\_stex_term_math_arg:nnn</code>	<code>\stex_term_arg:nnn&lt;int&gt;&lt;prec&gt;&lt;body&gt;</code> Annotates <code>&lt;body&gt;</code> as the <code>&lt;int&gt;</code> th argument of the current OMA or OMBIND, with (downwards) argument precedence <code>&lt;prec&gt;</code> .
<hr/> <hr/> <code>\_stex_term_math_assoc_arg:nnnn</code>	<code>\stex_term_arg:nnn&lt;int&gt;&lt;prec&gt;&lt;notation&gt;&lt;body&gt;</code> Annotates <code>&lt;body&gt;</code> as the <code>&lt;int&gt;</code> th (associative) <i>sequence</i> argument (as comma-separated list of terms) of the current OMA or OMBIND, with (downwards) argument precedence <code>&lt;prec&gt;</code> and associative notation <code>&lt;notation&gt;</code> .

<hr/> <hr/>	$\backslash\infprec$ $\backslashneginfprec$	Maximal and minimal notation precedences.
<hr/> <hr/>	$\backslashdobrackets$	$\backslashdobrackets \{ \langle body \rangle \}$  Puts $\langle body \rangle$ in parentheses; scaled if in display mode unscaled otherwise. Uses the current $\text{\S I E X}$ brackets (by default ( and )), which can be changed temporarily using $\backslash\withbrackets$ .
<hr/> <hr/>	$\backslash\withbrackets$	$\backslash\withbrackets \langle left \rangle \langle right \rangle \{ \langle body \rangle \}$  Temporarily (i.e. within $\langle body \rangle$ ) sets the brackets used by $\text{\S I E X}$ for automated bracketing (by default ( and )) to $\langle left \rangle$ and $\langle right \rangle$ . Note that $\langle left \rangle$ and $\langle right \rangle$ need to be allowed after $\backslash\left$ and $\backslash\right$ 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}_{\comp\langle }{ {#1}_{\comp\langle } }
 $\bar{foobar} a\{b,c,d,e,f\}g$  and  $\bar{foobar}[foo] a\{b,c\}g$  and  $\bar{foobar} abc$ 

\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}
 $\bar{plus}\{a,\mult\{b,c\}\}$  and  $\bar{mult}\{a,\plus\{\frac{ab}{c}\}\}$ 
 $\bar{plus}\{a,\mult\{b,c\}\}$  and  $\bar{mult}\{a,\plus\{\frac{ab}{c}\}\}$ 
 $\displaystyle \bar{plus}\{a,\mult\{b,c\}\}$  and  $\displaystyle \bar{mult}\{a,\plus\{\frac{ab}{c}\}\}$ 
\withbrackets[ {  $\displaystyle$  }
\end{module}

```

Module 8.1.2[MathTest2]  
modulesImporting module: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo  $\langle a|[b:c,d:e,f]^9 \rangle$   
and  $\langle a|[b:c]^9 \rangle$  and  $\langle a|[b]^c \rangle$   
 $a+(b\cdot c)$  and  $a\cdot\frac{a}{b}+\frac{a}{c}$   
 $a+(b\cdot c)$  and  $a\cdot\frac{a}{b}+\frac{a}{c}$

---

---

`\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 here.
some a and some b and also some c here.
bar
or just some c
bar
or first b, then c, and finally a
```

---

---

`\stex_highlight_term:nn`

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

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

---

`\comp`

`\comp{<args>}`

`\compemph`

`\compemph@uri`

`\defemph`

`\defemph@uri`

`\symrefemph`

`\symrefemph@uri`

---

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**b**:*M*

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

feature?op

»macro:->\stex\_invoke\_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?StructureTest1?Magma}<

Test: a+b

Test2: *(U,+)*

## Chapter 10

# TeX-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 T<sub>E</sub>X/L<sup>A</sup>T<sub>E</sub>X that allows to markup T<sub>E</sub>X/L<sup>A</sup>T<sub>E</sub>X documents semantically without leaving the document format, essentially turning T<sub>E</sub>X/L<sup>A</sup>T<sub>E</sub>X 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  $\text{\LaTeX}$  files. This structure can be used by MKM systems for added-value services, either directly from the  $\text{\LaTeX}$  sources, or after translation. Even though it is part of the  $\text{\LaTeX}$  collection, it can be used independently, like it's sister package **statements**.

$\text{\LaTeX}$  is a version of  $\text{\TeX}/\text{\LaTeX}$  that allows to markup  $\text{\TeX}/\text{\LaTeX}$  documents semantically without leaving the document format, essentially turning  $\text{\TeX}/\text{\LaTeX}$  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{spfcase}{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{spfcase}
\end{sproof}
```

Example 1: A very explicit proof, marked up semantically

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).<sup>4</sup>

<sup>4</sup>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 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 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.

<b>Proof:</b>	We prove that $\sum_{i=1}^n 2i - 1 = n^2$ by induction over $n$	
<b>P.1</b>	For the induction we have to consider the following cases:	
<b>P.1.1</b>	$n = 1$ : then we compute $1 = 1^2$	□
<b>P.1.1</b>	$n = 2$ : This case is not really necessary, but we do it for the fun of it (and to get more intuition). We compute $1 + 3 = 2^2 = 4$	□
<b>P.1.1</b>	$n > 1$ :	
<b>P.1.1.1</b>	Now, we assume that the assertion is true for a certain $k \geq 1$ , i.e. $\sum_{i=1}^k (2i - 1) = k^2$ .	
<b>P.1.1.1</b>	We have to show that we can derive the assertion for $n = k + 1$ from this assumption, i.e. $\sum_{i=1}^{k+1} (2i - 1) = (k + 1)^2$ .	
<b>P.1.1.1</b>	We obtain $\sum_{i=1}^{k+1} (2i - 1) = \sum_{i=1}^k (2i - 1) + 2(k + 1) - 1$ by splitting the sum	
<b>P.1.1.1</b>	Thus we have $\sum_{i=1}^{k+1} (2i - 1) = k^2 + 2k + 1$ by inductive hypothesis.	
<b>P.1.1.1</b>	We can simplify the right-hand side to $(k + 1)^2$ , which proves the assertion.	□
<b>P.1.1</b>	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

<b>subproof</b>	The <b>pfcases</b> environment is used to mark up a subproof. This environment takes an optional <b>KeyVal</b> argument for semantic annotations and a second argument that allows to specify an introductory comment (just like in the <b>proof</b> environment). The <b>method</b> key can be used to give the name of the proof method executed to make this subproof.
<b>spfcases</b>	The <b>pfcases</b> environment is used to mark up a proof by cases. Technically it is a variant of the <b>subproof</b> where the <b>method</b> is <b>by-cases</b> . Its contents are <b>spfcases</b> environments that mark up the cases one by one.
<b>spfcases</b>	The content of a <b>pfcases</b> environment are a sequence of case proofs marked up in the <b>pfcases</b> environment, which takes an optional <b>KeyVal</b> argument for semantic annotations. The second argument is used to specify the the description of the case under consideration. The content of a <b>pfcases</b> environment is the same as that of a <b>proof</b> , i.e. <b>steps</b> , <b>proofcomments</b> , and <b>pfcases</b> environments. <b>\spfcasesketch</b> is a variant of the <b>spfcases</b> environment that takes the same arguments, but instead of the <b>spfsteps</b> in the body uses a third argument for a proof sketch.
<b>\spfcasesketch</b>	
<b>sproofcomment</b>	The <b>proofcomment</b> environment is much like a <b>step</b> , 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 <b>\premise</b> .



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 ( $n$ th-order) logic, or a  $\Pi$  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 L<sup>A</sup>T<sub>E</sub>X

The `omdoc` package is part of the  $\S\text{TeX}$  collection, a version of  $\text{T}\text{E}\text{X}/\text{L}\text{A}\text{T}\text{E}\text{X}$  that allows to markup  $\text{T}\text{E}\text{X}/\text{L}\text{A}\text{T}\text{E}\text{X}$  documents semantically without leaving the document format, essentially turning  $\text{T}\text{E}\text{X}/\text{L}\text{A}\text{T}\text{E}\text{X}$  into a document format for mathematical knowledge management (MKM).

This package supplies an infrastructure for writing OMDOC documents in  $\text{L}\text{A}\text{T}\text{E}\text{X}$ . This includes a simple structure sharing mechanism for  $\S\text{TeX}$  that allows 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  $\S\text{TeX}$  sources, or after translation.

### 14.1 Introduction

$\S\text{TeX}$  is a version of  $\text{T}\text{E}\text{X}/\text{L}\text{A}\text{T}\text{E}\text{X}$  that allows to markup  $\text{T}\text{E}\text{X}/\text{L}\text{A}\text{T}\text{E}\text{X}$  documents semantically without leaving the document format, essentially turning  $\text{T}\text{E}\text{X}/\text{L}\text{A}\text{T}\text{E}\text{X}$  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  $\S\text{TeX}$  sources, or after translation. Currently, trans-document referencing provided by this package can only be used in the  $\S\text{TeX}$  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



source and the formatter does the copying during document formatting/presentation.<sup>6</sup>

## 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:

<code>class=&lt;name&gt;</code>	load <code>&lt;name&gt;.cls</code> instead of <code>article.cls</code>
<code>topsect=&lt;sect&gt;</code>	The top-level sectioning level; the default for <code>&lt;sect&gt;</code> is <code>section</code>
<code>showignores</code>	show the the contents of the <code>ignore</code> environment after all
<code>showmeta</code>	show the metadata; see <code>metakeys.sty</code>
<code>showmods</code>	show modules; see <code>modules.sty</code>
<code>extrefs</code>	allow external references; see <code>sref.sty</code>
<code>defindex</code>	index definienda; see <code>statements.sty</code>
<code>minimal</code>	for testing; do not load any $\text{\TeX}$ packages

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

### 14.2.2 Document Structure

`document` The top-level `document` environment can be given key/value information by the `\documentkeys` macro in the preamble<sup>2</sup>. 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 L<sup>A</sup>T<sub>E</sub>XML transformation.

`omgroup` The structure of the document is given by the `omgroup` environment just like in OM-DOC. In the L<sup>A</sup>T<sub>E</sub>X 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

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

`blindomgroup` L<sup>A</sup>T<sub>E</sub>X 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`

<sup>6</sup>EDNOTE: integrate with latexml's XMRef in the Math mode.

<sup>2</sup>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”.
- The inner one groups the frontmatter<sup>3</sup> 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

<sup>3</sup>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  $\text{\LaTeX}$  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`.

For prematurely stopping the formatting of a document,  $\text{\LaTeX}$  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

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

The `\STRlabel` macro has a variant `\STRsemantics`, where the label argument is optional, and which takes a third argument, which is ignored in  $\text{\LaTeX}$ . 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.<sup>7</sup>

#### 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  $\text{\LaTeX}$  preamble of the course notes file. `\setSGvar`{ $\langle vname \rangle$ }{ $\langle text \rangle$ } to set the global variable  $\langle vname \rangle$  to  $\langle text \rangle$  and `\useSGvar`{ $\langle vname \rangle$ } to reference it.

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

<sup>7</sup>EdNOTE: document LMID und LMXRef here if we decide to keep them.

### 14.2.6 Colors

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 `TeX` 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  $\text{\TeX}$ and 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

The `mikoslides` class takes a variety of class options:<sup>8</sup>

- |                           |  |
|---------------------------|--|
| <code>slides</code>       | <ul style="list-style-type: none"><li>• The options <code>slides</code> and <code>notes</code> switch between slides mode and notes mode (see Section 15.2.2).</li></ul>   |
| <code>notes</code>        |  |
| <code>sectocframes</code> | <ul style="list-style-type: none"><li>• If the option <code>sectocframes</code> is given, then for the <code>omgroups</code>, special frames with the <code>omgroup</code> title (and number) are generated.</li></ul> |

<code>showmeta</code>	<ul style="list-style-type: none"> <li>• <code>showmeta</code>. If this is set, then the metadata keys are shown (see [Koh20b] for details and customization options).</li> </ul>
<code>frameimages</code> <code>fiboxed</code>	<ul style="list-style-type: none"> <li>• If the option <code>frameimages</code> is set, then slide mode also shows the <code>\frameimage</code>-generated frames (see section 15.2.4). If also the <code>fiboxed</code> option is given, the slides are surrounded by a box.</li> </ul>
<code>topsect</code>	<ul style="list-style-type: none"> <li>• <code>topsect=&lt;sect&gt;</code> can be used to specify the top-level sectioning level; the default for <code>&lt;sect&gt;</code> is <code>section</code>.</li> </ul>

### 15.2.2 Notes and Slides

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

⚠ 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 L<sup>A</sup>T<sub>E</sub>X 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 `\notesttrue` or `\notesfalse` is strongly discouraged however.

<sup>8</sup>EDNOTE: leaving out `noproblems` for the moment until we decide what to do with it.

<sup>4</sup>MK: it would be very nice, if we did not need this environment, and this should be possible in principle, but not without intensive L<sup>A</sup>T<sub>E</sub>X 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).

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}`.

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

The default logo provided by the `mikoslides` package is the  $\text{\LaTeX}$  logo it can be customized using `\setslidelogo{<logo name>}`.

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 Attribution-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

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  $\text{\LaTeX}$ notes. 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.<sup>9</sup>

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

<sup>9</sup>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      The \excursion{<ref>}{<path>}{<text>} is syntactic sugar for
\activateexcursion
\begin{nomtext}[title=Excursion]
  \activateexcursion{founif}{../ex/founif}
  We will cover first-order unification in \sref{founif}.
\end{nomtext}
```

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

Sometimes, we want to reference – in an excursion – part of another. We can use

```
\excursionref \excursionref{<label>} for that.
```

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 \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<sup>5</sup>. 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

<code>solutions</code>	The <code>problem</code> package takes the options <code>solutions</code> (should solutions be output?), <code>notes</code>
<code>notes</code>	(should the problem notes be presented?), <code>hints</code> (do we give the hints?), <code>gnotes</code> (do we
<code>hints</code>	show grading notes?), <code>pts</code> (do we display the points awarded for solving the problem?),
<code>gnotes</code>	<code>min</code> (do we display the estimated minutes for problem soling). If theses are specified, then
<code>pts</code>	the corresponding auxiliary parts of the problems are output, otherwise, they remain
<code>min</code>	invisible.
<code>boxed</code>	The <code>boxed</code> option specifies that problems should be formatted in framed boxes so
<code>test</code>	that they are more visible in the text. Finally, the <code>test</code> 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.
<code>mh</code>	The <code>mh</code> option turns on MathHub support; see [ <code>Kohlhase:mss</code> ].
<code>showmeta</code>	Finally, if the <code>showmeta</code> is set, then the metadata keys are shown (see [ <code>Kohlhase:metakeys</code> ]
	for details and customization options).

---

<sup>5</sup>for the moment multiple choice problems are not supported, but may well be in a future version

## 16.2.2 Problems and Solutions

**problem** 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=elephants,pts=10,min=2,title=Fitting Elephants]
    How many Elephants 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=elephants,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** 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 Elephants 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** 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.  
**exnote**  
**gnote** The **gnote** (grading notes) environment can be used to document situations 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.

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.

### 16.2.3 Multiple Choice Blocks

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

See Figure ?? for an example

### 16.2.4 Including Problems

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.

### 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=Noooooooooooo,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` 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  
`number` — 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 L<sup>A</sup>T<sub>E</sub>X source.

`\testspace` `\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` Finally, the `\testheading` takes an optional keyword argument where the keys  
`duration` `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.  
`min`  
`reqpts`

### 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.

### 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 `STEX`GitHub repository [[sTeX](#)].

1. none reported yet.

---

Name:
MatriculationNumber:

2022-01-18

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.

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

[illegible]

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.

```
1 <*cls>
2
3 %%%%%%%%%%% basics.dtx %%%%%%%%%%%
4
5 \RequirePackage{expl3,l3keys2e}
6 \ProvidesExplClass{stex}{2021/08/01}{1.9}{bla}
7 \LoadClass[border=1px,varwidth]{standalone}
8 \setlength\textwidth{15cm}
9
10 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
11 \ProcessOptions
12
13 \RequirePackage{stex}
14 </cls>
```

### 18.2 Preliminaries

```
15 <*package>
16
17 %%%%%%%%%%% basics.dtx %%%%%%%%%%%
18
19 \RequirePackage{expl3,l3keys2e,ltxcmds}
20 \ProvidesExplPackage{stex}{2021/08/01}{1.9}{bla}
21 \RequirePackage{expl-keystr-compat}
22 \RequirePackage{morewrites}
23 \RequirePackage{amsmath}
24
25 Package options:
26 \keys_define:nn { stex } {
27   debug      .clist_set:N = \c_stex_debug_clist ,
```

```

26 showmods .bool_set:N = \c_stex_showmods_bool ,
27 lang .clist_set:N = \c_stex_languages_clist ,
28 mathhub .tl_set_x:N = \mathhub ,
29 sms .bool_set:N = \c_stex_persist_mode_bool ,
30 image .bool_set:N = \c_tikzinput_image_bool ,
31 unknown .code:n = {}
32 }
33 \ProcessKeysOptions { stex }

```

**\stex** The  $\text{\TeX}$  logo:

**\sTeX**

```

34 \protected\def\stex{%
35 \ifundefined{texorpdfstring}%
36 {\let\texorpdfstring\@firstoftwo}%
37 {}%
38 \texorpdfstring{\raisebox{-.5ex}{S}\kern-.5ex\TeX}{sTeX}\xspace%
39 }
40 \def\sTeX{\stex}

```

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

## 18.3 Messages and logging

```

41 <@@=stex_log>

Warnings and error messages
42 \msg_new:nnn{stex}{error/unknownlanguage}{
43   Unknown~language:~#1
44 }
45 \msg_new:nnn{stex}{warning/nomathhub}{
46   MATHHUB~system~variable~not~found~and~no~
47   \detokenize{\mathhub}~value~set!
48 }
49 \msg_new:nnn{stex}{error/deactivated-macro}{
50   The~\detokenize{#1}~command~is~only~allowed~in~#2!
51 }

```

**\stex\_debug:nn** A simple macro issuing package messages with subpath.

```

52 \cs_new_protected:Nn \stex_debug:nn {
53   \clist_if_in:NnTF \c_stex_debug_clist { all } {
54     \exp_args:Nnnx\msg_set:nnn{stex}{debug / #1}{
55       \\Debug~#1:~#2\\
56     }
57     \msg_none:nn{stex}{debug / #1}
58   }{
59     \clist_if_in:NnT \c_stex_debug_clist { #1 } {
60       \exp_args:Nnnx\msg_set:nnn{stex}{debug / #1}{
61         \\Debug~#1:~#2\\
62       }
63       \msg_none:nn{stex}{debug / #1}
64     }
65   }
66 }

```

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

Redirecting messages:

```

67 \clist_if_in:NnTF \c_stex_debug_clist {all} {
68   \msg_redirect_module:nnn{ stex }{ none }{ term }
69 }{
70   \clist_map_inline:Nn \c_stex_debug_clist {
71     \msg_redirect_name:nnn{ stex }{ debug / ##1 }{ term }
72   }
73 }
74
75 \stex_debug:nn{log}{debug~mode~on}

```

## 18.4 Persistence

76 `<@=stex_persist>`

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

```

77 \iow_new:N \c__stex_persist_sms_iow
78 \AddToHook{begindocument}{
79   \bool_if:NTF \c_stex_persist_mode_bool {
80     \ExplSyntaxOn \input{\jobname.sms} \ExplSyntaxOff
81   } {
82     \iow_open:Nn \c__stex_persist_sms_iow {\jobname.sms}
83   }
84 }
85 \AddToHook{enddocument}{
86   \bool_if:NF \c_stex_persist_mode_bool {
87     \iow_close:N \c__stex_persist_sms_iow
88   }
89 }

```

(End definition for `\c__stex_persist_sms_iow`.)

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

```

90 \cs_new_protected:Nn \stex_add_to_sms:n {
91   \bool_if:NF \c_stex_persist_mode_bool {
92     \iow_now:Nn \c__stex_persist_sms_iow { #1 }
93   }
94 }

```

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

## 18.5 HTML Annotations

95 `<@=stex_annotate>`  
96 `\RequirePackage{rustex}`

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

```

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

```

`\if@latexml` Conditionals for L<sup>A</sup>T<sub>E</sub>X<sub>M</sub>L:

```

\latexml_if_p:
\latexml_if:TF
98 \ifcsname if@latexml\endcsname\else

```

```

99     \expandafter\newif\csname if@latexml\endcsname\@latexmlfalse
100 \fi
101
102 \prg_new_conditional:Nnn \latexml_if: {p, T, F, TF} {
103   \if@latexml
104     \prg_return_true:
105   \else:
106     \prg_return_false:
107   \fi:
108 }

```

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

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

```

109 \tl_new:N \l__stex_annotate_arg_tl
110 \tl_const:Nx \c__stex_annotate_emptyarg_tl {
111   \rustex_if:TF {
112     \rustex_direct_HTML:n { \c_ampersand_str lrm; }
113   }{-}
114 }

```

(End definition for \l\_\_stex\_annotate\_arg\_tl and \c\_\_stex\_annotate\_emptyarg\_tl.)

`\_stex_annotate_checkempty:n`

```

115 \cs_new_protected:Nn \_stex_annotate_checkempty:n {
116   \tl_set:Nn \l__stex_annotate_arg_tl { #1 }
117   \tl_if_empty:NT \l__stex_annotate_arg_tl {
118     \tl_set_eq:NN \l__stex_annotate_arg_tl \c__stex_annotate_emptyarg_tl
119   }
120 }

```

(End definition for \\_stex\_annotate\_checkempty:n.)

`\l_stex_html_do_output_bool` Whether to (locally) produce HTML output

```

\stex_if_do_html:
121 \bool_new:N \l_stex_html_do_output_bool
122 \bool_set_true:N \l_stex_html_do_output_bool
123 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
124   \bool_if:nTF \l_stex_html_do_output_bool
125     \prg_return_true: \prg_return_false:
126 }

```

(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

```

127 \cs_new_protected:Nn \stex_suppress_html:n {
128   \exp_args:Nne \use:nn {
129     \bool_set_false:N \l_stex_html_do_output_bool
130     #1
131   }{
132     \stex_if_do_html:T {
133       \bool_set_true:N \l_stex_html_do_output_bool
134     }
135   }
136 }

```

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

`\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 (L<sup>A</sup>T<sub>E</sub>XML, R<sub>U</sub>S<sub>T</sub>E<sub>X</sub>, p<sub>D</sub>F<sub>L</sub>A<sub>T</sub>E<sub>X</sub>).

The p<sub>D</sub>F<sub>L</sub>A<sub>T</sub>E<sub>X</sub>-macros largely do nothing; the R<sub>U</sub>S<sub>T</sub>E<sub>X</sub>-implementations are pretty clear in what they do, the L<sup>A</sup>T<sub>E</sub>XML-implementations resort to perl bindings.

```

137 \rustex_if:TF{
138   \cs_new_protected:Nn \stex_annotate:nnn {
139     \__stex_annotate_checkempty:n { #3 }
140     \rustex_annotate_HTML:nn {
141       property="stex:#1" ~
142       resource="#2"
143     } {
144       \mode_if_vertical:TF{
145         \tl_use:N \l__stex_annotate_arg_tl\par
146       }{
147         \tl_use:N \l__stex_annotate_arg_tl
148       }
149     }
150   }
151   \cs_new_protected:Nn \stex_annotate_invisible:n {
152     \__stex_annotate_checkempty:n { #1 }
153     \rustex_annotate_HTML:nn {
154       stex:visible="false" ~
155       style:display="none"
156     } {
157       \mode_if_vertical:TF{
158         \tl_use:N \l__stex_annotate_arg_tl\par
159       }{
160         \tl_use:N \l__stex_annotate_arg_tl
161       }
162     }
163   }
164   \cs_new_protected:Nn \stex_annotate_invisible:nnn {
165     \__stex_annotate_checkempty:n { #3 }
166     \rustex_annotate_HTML:nn {
167       property="stex:#1" ~
168       resource="#2" ~
169       stex:visible="false" ~
170       style:display="none"
171     } {
172       \mode_if_vertical:TF{
173         \tl_use:N \l__stex_annotate_arg_tl\par
174       }{
175         \tl_use:N \l__stex_annotate_arg_tl
176       }
177     }
178   }
179   \NewDocumentEnvironment{stex_annotate_env} { m m } {
180     \par
181     \rustex_annotate_HTML_begin:n {
182       property="stex:#1" ~
183       resource="#2"
184     }

```

```

185   }{
186   \par\rustex_annotate_HTML_end:
187   }
188 }{
189 \latexml_if:TF {
190   \cs_new_protected:Nn \stex_annotate:nnn {
191     \__stex_annotate_checkempty:n { #3 }
192     \mode_if_math:TF {
193       \cs:w latexml@annotate@math\cs_end:{#1}{#2}{
194         \tl_use:N \l__stex_annotate_arg_tl
195       }
196     }{
197       \cs:w latexml@annotate@text\cs_end:{#1}{#2}{
198         \tl_use:N \l__stex_annotate_arg_tl
199       }
200     }
201   }
202   \cs_new_protected:Nn \stex_annotate_invisible:n {
203     \__stex_annotate_checkempty:n { #1 }
204     \mode_if_math:TF {
205       \cs:w latexml@invisible@math\cs_end:{
206         \tl_use:N \l__stex_annotate_arg_tl
207       }
208     } {
209       \cs:w latexml@invisible@text\cs_end:{
210         \tl_use:N \l__stex_annotate_arg_tl
211       }
212     }
213   }
214   \cs_new_protected:Nn \stex_annotate_invisible:nnn {
215     \__stex_annotate_checkempty:n { #3 }
216     \cs:w latexml@annotate@invisible\cs_end:{#1}{#2}{
217       \tl_use:N \l__stex_annotate_arg_tl
218     }
219   }
220   \NewDocumentEnvironment{stex_annotate_env} { m m } {
221     \par\begin{latexml@annotateenv}{#1}{#2}
222   }{
223     \par\end{latexml@annotateenv}
224   }
225 }{
226   \cs_new_protected:Nn \stex_annotate:nnn {#3}
227   \cs_new_protected:Nn \stex_annotate_invisible:n {}
228   \cs_new_protected:Nn \stex_annotate_invisible:nnn {}
229   \NewDocumentEnvironment{stex_annotate_env} { m m } {}{}
230 }
231 }

```

(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

```

232 <@=stex_language>

```

`\c_stex_languages_prop`  
`\c_stex_language_abbrevs_prop`

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

```

233 \prop_const_from_keyval:Nn \c_stex_languages_prop {
234   en = english ,
235   de = ngerman ,
236   ar = arabic ,
237   bg = bulgarian ,
238   ru = russian ,
239   fi = finnish ,
240   ro = romanian ,
241   tr = turkish ,
242   fr = french
243 }
244
245 \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop {
246   english   = en ,
247   ngerman   = de ,
248   arabic    = ar ,
249   bulgarian = bg ,
250   russian   = ru ,
251   finnish   = fi ,
252   romanian  = ro ,
253   turkish   = tr ,
254   french    = fr
255 }
256 % todo: chinese simplified (zhs)
257 %       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:

```

258 \clist_if_empty:NF \c_stex_languages_clist {
259   \clist_clear:N \l_tmpa_clist
260   \clist_map_inline:Nn \c_stex_languages_clist {
261     \prop_get:NnNTF \c_stex_languages_prop { #1 } \l_tmpa_str {
262       \clist_put_right:No \l_tmpa_clist \l_tmpa_str
263     } {
264       \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
265     }
266   }
267   \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
268   \RequirePackage[\clist_use:Nn \l_tmpa_clist,]{babel}
269 }

```

## 18.7 Activating/Deactivating Macros

`\stex_deactivate_macro:Nn`

```

270 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
271   \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
272   \def#1{
273     \msg_error:nnxx{stex}{error/deactivated-macro}{#1}{#2}
274   }
275 }

```



*(End definition for \stex\_deactivate\_macro:Nn. This function is documented on page 10.)*

**\stex\_reactivate\_macro:N**

```
276 \cs_new_protected:Nn \stex_reactivate_macro:N {  
277   \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname  
278 }
```

*(End definition for \stex\_reactivate\_macro:N. This function is documented on page 10.)*

```
279 \</package>
```

## Chapter 19

# STEX -MathHub Implementation

```
280 <*package>
281
282 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
283
284 <@@=stex_path>
285
286 Warnings and error messages
287 \msg_new:nnn{stex}{error/norepository}{
288   No~archive~#1~found~in~#2
289 }
290 \msg_new:nnn{stex}{error/notinarchive}{
291   Not~currently~in~an~archive,~but~\detokenize{#1}~
292   needs~one!
293 }
294 \msg_new:nnn{stex}{error/nofile}{
295   \detokenize{#1}~could~not~find~file~#2
296 }
```

### 19.1 Generic Path Handling

We treat paths as L<sup>A</sup>T<sub>E</sub>X3-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
295 \cs_new_protected:Nn \stex_path_from_string:Nn {
296   \str_set:Nx \l_tmpa_str { #2 }
297   \str_if_empty:NTF \l_tmpa_str {
298     \seq_clear:N #1
299   }{
300     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
301     \sys_if_platform_windows:T{
302       \seq_clear:N \l_tmpa_tl
303       \seq_map_inline:Nn #1 {
304         \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
305         \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
306       }
307     }
308   }
```

```

306     }
307     \seq_set_eq:NN #1 \l_tmpa_tl
308   }
309   \stex_path_canonicalize:N #1
310 }
311 }
312 \cs_generate_variant:Nn \stex_path_from_string:Nn
313 { 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
314 \cs_new_protected:Nn \stex_path_to_string:NN {
315   \exp_args:NNe \str_set:Nn #2 { \seq_use:Nn #1 / }
316 }
317
318 \cs_new:Nn \stex_path_to_string:N {
319   \seq_use:Nn #1 /
320 }

```

(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 . and .., respectively.
\c__stex_path_up_str
321 \str_const:Nn \c__stex_path_dot_str {.}
322 \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.

```

323 \cs_new_protected:Nn \stex_path_canonicalize:N {
324   \seq_if_empty:NF #1 {
325     \seq_clear:N \l_tmpa_seq
326     \seq_get_left:NN #1 \l_tmpa_tl
327     \str_if_empty:NT \l_tmpa_tl {
328       \seq_put_right:Nn \l_tmpa_seq {}
329     }
330     \seq_map_inline:Nn #1 {
331       \str_set:Nn \l_tmpa_tl { ##1 }
332       \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_dot_str {} {
333         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
334           \seq_if_empty:NTF \l_tmpa_seq {
335             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
336               \c__stex_path_up_str
337             }
338           }{
339             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
340             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
341               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
342                 \c__stex_path_up_str
343               }
344             }{
345               \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
346             }

```

```

347     }
348   }{
349     \str_if_empty:NF \l_tmpa_tl {
350       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
351     }
352   }
353 }
354 }
355 \seq_gset_eq:NN #1 \l_tmpa_seq
356 }
357 }

```

(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`

```

358 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
359   \seq_if_empty:NTF #1 {
360     \prg_return_false:
361   }{
362     \seq_get_left:NN #1 \l_tmpa_tl
363     \str_if_empty:NTF \l_tmpa_tl {
364       \prg_return_true:
365     }{
366       \prg_return_false:
367     }
368   }
369 }

```

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

## 19.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

370 \str_new:N\l_stex_kpsewhich_return_str
371 \cs_new_protected:Nn \stex_kpsewhich:n {
372   \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
373   \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
374   \tl_trim_spaces:N \l_stex_kpsewhich_return_str
375 }

```

(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`

```

376 \sys_if_platform_windows:TF{
377   \stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
378 }{
379   \stex_kpsewhich:n{-var-value~PWD}
380 }
381
382 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
383 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
384 \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

385 `<@@=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  $\text{\TeX}$ -purposes.

`\g__stex_files_stack` keeps track of file changes

386 `\seq_gclear_new:N\g__stex_files_stack`

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`

`\c_stex_mainfile_str`

387 `\str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}`

388 `\stex_path_from_string:Nn \c_stex_mainfile_seq`

389 `\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`.

390 `\seq_gclear_new:N\g_stex_currentfile_seq`

391 `\AddToHook{file/before}{`

392 `\stex_path_from_string:Nn\g_stex_currentfile_seq{\CurrentFilePath}`

393 `\stex_path_if_absolute:NTF\g_stex_currentfile_seq{`

394 `\exp_args:NNe\seq_put_right:Nn\g_stex_currentfile_seq{\CurrentFile}`

395 `}{`

396 `\stex_path_from_string:Nn\g_stex_currentfile_seq{`

397 `\c_stex_pwd_str/\CurrentFilePath/\CurrentFile`

398 `}`

399 `}`

400 `\seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq`

401 `\exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq`

402 `}`

403 `\AddToHook{file/after}{`

404 `\seq_if_empty:NF\g__stex_files_stack{`

405 `\seq_gpop:Nn\g__stex_files_stack\l_tmpa_seq`

406 `}`

407 `\seq_if_empty:NTF\g__stex_files_stack{`

408 `\seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq`

409 `}{`

410 `\seq_get:NN\g__stex_files_stack\l_tmpa_seq`

411 `\seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq`

412 `}`

413 `}`

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

## 19.4 MathHub Repositories

```

414 <@=stex_mathhub>

\mathhub
\c_stex_mathhub_seq
\c_stex_mathhub_str
415 \str_if_empty:NTF\mathhub{
416   \stex_kpsewhich:n{-var-value~MATHHUB}
417   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
418
419   \str_if_empty:NTF\c_stex_mathhub_str{
420     \msg_warning:nn{stex}{warning/nomathhub}
421   }{
422     \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
423     \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
424   }
425 }{
426   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
427   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
428     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
429       \c_stex_pwd_str/\mathhub
430     }
431   }
432   \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
433   \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
434 }

```

(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
435 \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
436   \str_set:Nx \l_tmpa_str { #1 }
437   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
438     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
439     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
440     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
441     \__stex_mathhub_find_manifest:N \l_tmpa_seq
442     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
443       \msg_error:nnxx{stex}{error/norepository}{#1}{
444         \stex_path_to_string:N \c_stex_mathhub_str
445       }
446     } {
447       \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
448     }
449   }
450 }

```

(End definition for `\__stex_mathhub_do_manifest:n`.)

```

\l__stex_mathhub_manifest_file_seq
451 \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`:

```

452 \cs_new_protected:Nn \_stex_mathhub_find_manifest:N {
453   \seq_set_eq:NN \l_tmpa_seq #1
454   \bool_set_true:N \l_tmpa_bool
455   \bool_while_do:Nn \l_tmpa_bool {
456     \seq_if_empty:NTF \l_tmpa_seq {
457       \bool_set_false:N \l_tmpa_bool
458     }{
459       \file_if_exist:nTF{
460         \stex_path_to_string:N \l_tmpa_seq/MANIFEST.MF
461       }{
462         \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
463         \bool_set_false:N \l_tmpa_bool
464       }{
465         \file_if_exist:nTF{
466           \stex_path_to_string:N \l_tmpa_seq/META-INF/MANIFEST.MF
467         }{
468           \seq_put_right:Nn \l_tmpa_seq{META-INF}
469           \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
470           \bool_set_false:N \l_tmpa_bool
471         }{
472           \file_if_exist:nTF{
473             \stex_path_to_string:N \l_tmpa_seq/meta-inf/MANIFEST.MF
474           }{
475             \seq_put_right:Nn \l_tmpa_seq{meta-inf}
476             \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
477             \bool_set_false:N \l_tmpa_bool
478           }{
479             \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
480           }
481         }
482       }
483     }
484   }
485   \seq_set_eq:NN \l__stex_mathhub_manifest_file_seq \l_tmpa_seq
486 }

```

(End definition for `\_stex_mathhub_find_manifest:N`.)

`\c_stex_mathhub_manifest_ior` File variable used for MANIFEST-files

```

487 \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:

```

488 \cs_new_protected:Nn \_stex_mathhub_parse_manifest:n {
489   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
490   \ior_open:Nn \c_stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
491   \ior_map_inline:Nn \c_stex_mathhub_manifest_ior {
492     \str_set:Nn \l_tmpa_str {##1}
493     \exp_args:NNoo \seq_set_split:Nnn
494       \l_tmpb_seq \c_colon_str \l_tmpa_str
495     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {

```

```

496 \exp_args:NNe \str_set:Nn \l_tmpb_tl {
497 \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
498 }
499 \exp_args:No \str_case:nnTF \l_tmpa_tl {
500 {id} {
501 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
502 { id } \l_tmpb_tl
503 }
504 {narration-base} {
505 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
506 { narr } \l_tmpb_tl
507 }
508 {url-base} {
509 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
510 { docurl } \l_tmpb_tl
511 }
512 {source-base} {
513 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
514 { ns } \l_tmpb_tl
515 }
516 {ns} {
517 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
518 { ns } \l_tmpb_tl
519 }
520 {dependencies} {
521 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
522 { deps } \l_tmpb_tl
523 }
524 }{}{}
525 }{}
526 }
527 \ior_close:N \c__stex_mathhub_manifest_ior
528 }

```

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

`\stex_set_current_repository:n`

```

529 \cs_new_protected:Nn \stex_set_current_repository:n {
530 \stex_require_repository:n { #1 }
531 \prop_set_eq:Nc \l_stex_current_repository_prop {
532 c_stex_mathhub_#1_manifest_prop
533 }
534 }

```

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

`\stex_require_repository:n`

```

535 \cs_new_protected:Nn \stex_require_repository:n {
536 \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
537 \stex_debug:nn{mathhub}{Opening~archive:~#1}
538 \__stex_mathhub_do_manifest:n { #1 }
539 \exp_args:Nx \stex_add_to_sms:n {
540 \prop_const_from_keyval:cn { c_stex_mathhub_#1_manifest_prop } {
541 id = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { id } ,
542 ns = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { ns } ,

```



```

543     narr = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { narr } ,
544     deps = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { deps }
545   }
546 }
547 }
548 }

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

549 \prop_new:N \l_stex_current_repository_prop
550
551 \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
552 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
553   \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
554 } {
555   \__stex_mathhub_parse_manifest:n { main }
556   \prop_get:Nn \c_stex_mathhub_main_manifest_prop {id}
557   \l_tmpa_str
558   \prop_set_eq:cN { c_stex_mathhub_ \l_tmpa_str _manifest_prop }
559   \c_stex_mathhub_main_manifest_prop
560   \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
561   \stex_debug:nn{mathhub}{Current~repository:~
562     \prop_item:Nn \l_stex_current_repository_prop {id}
563   }
564 }

```

(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.

```

565 \cs_new_protected:Nn \stex_in_repository:nn {
566   \str_set:Nx \l_tmpa_str { #1 }
567   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
568   \str_if_empty:NTF \l_tmpa_str {
569     \exp_args:Ne \l_tmpa_cs{
570       \prop_item:Nn \l_stex_current_repository_prop { id }
571     }
572   }{
573     \stex_require_repository:n \l_tmpa_str
574     \str_set:Nx \l_tmpa_str { #1 }
575     \exp_args:Nne \use:nn {
576       \stex_set_current_repository:n \l_tmpa_str
577       \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
578     }{
579       \stex_set_current_repository:n {
580         \prop_item:Nn \l_stex_current_repository_prop { id }
581       }
582     }
583   }
584 }

```

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

```

\inputref
\stex_inputref:nn
585 \newif \ifinputref \inputreffalse
586
587 \cs_new_protected:Nn \stex_inputref:nn {
588   \stex_in_repository:nn {#1} {
589     \ifinputref
590       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
591     \else
592       \inputreftrue
593       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
594     \inputreffalse
595   \fi
596 }
597 }
598 \NewDocumentCommand \inputref { 0{} m}{
599   \stex_inputref:nn{ #1 }{ #2 }
600 }
601
602 \cs_new_protected:Nn \stex_mhbibresource:nn {
603   \stex_in_repository:nn {#1} {
604     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
605   }
606 }
607 \newcommand\addmhbibresource[2][]{
608   \stex_mhbibresource:nn{ #1 }{ #2 }
609 }

```

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

### `\mhpath`

```

610 \def \mhpath #1 #2 {
611   \exp_args:Nc \str_if_eq:nnTF{#1}{#{
612     \c_stex_mathhub_str /
613     \prop_item:Nn \l_stex_current_repository_prop { id }
614     / source / #2
615   }{
616     \c_stex_mathhub_str / #1 / source / #2
617   }
618 }

```

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

### `\libinput`

```

619 \cs_new_protected:Npn \libinput #1 {
620   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
621     \msg_error:nnn{stex}{error/notinarchive}\libinput
622   }
623   \bool_set_false:N \l_tmpa_bool
624   \tl_clear:N \l_tmpa_tl
625   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
626   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
627   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str
628   \seq_pop_left:NNT \l_tmpb_seq \l_tmpb_str {
629     \seq_put_right:No \l_tmpa_seq \l_tmpb_str

```

```

630 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
631 / meta-inf / lib / #1.tex}{
632 \bool_set_true:N \l_tmpa_bool
633 \tl_put_right:Nx \l_tmpa_tl {
634 \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
635 / meta-inf / lib / #1.tex}
636 }
637 }{}
638 }
639 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
640 / \l_tmpa_str / lib / #1.tex
641 }{
642 \bool_set_true:N \l_tmpa_bool
643 \tl_put_right:Nx \l_tmpa_tl {
644 \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
645 / \l_tmpa_str / lib / #1.tex}
646 }
647 }{}
648 \bool_if:NF \l_tmpa_bool {
649 \msg_error:nnnx{stex}{error/nofile}\libinput{#1.tex}
650 }
651 \l_tmpa_tl
652 }

```

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

```

653 </package>

```

## Chapter 20

# STEX -References Implementation

```
654 <*package>
655
656 %%%%%%%%%% references.dtx %%%%%%%%%%
657
658 %\RequirePackage{hyperref}
659 %\RequirePackage{cleveref}
660 <@@=stex_refs>
661
662 Warnings and error messages
663
664 \iow_new:N \c__stex_refs_refs_iow
665 \AddToHook{begindocument}{
666   \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
667 }
668 \AddToHook{enddocument}{
669   \iow_close:N \c__stex_refs_refs_iow
670 }
671
672 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
673
674 \NewDocumentCommand \STEXreftitle { m } {
675   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
676 }
677
```

### 20.1 Document URIs and URLs

```
675 \seq_new:N \g__stex_refs_all_refs_seq
676
677 \str_new:N \l_stex_current_docns_str
678
679 \cs_new_protected:Nn \stex_get_document_uri: {
680   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
681   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
682   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
683   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
684 }
685
```

```

684 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
685
686 \str_clear:N \l_tmpa_str
687 \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
688   \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
689 }
690
691 \str_if_empty:NTF \l_tmpa_str {
692   \str_set:Nx \l_stex_current_docns_str {
693     file:/\stex_path_to_string:N \l_tmpa_seq
694   }
695 }{
696   \bool_set_true:N \l_tmpa_bool
697   \bool_while_do:Nn \l_tmpa_bool {
698     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
699     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
700       {source} { \bool_set_false:N \l_tmpa_bool }
701     }{}{
702       \seq_if_empty:NT \l_tmpa_seq {
703         \bool_set_false:N \l_tmpa_bool
704       }
705     }
706   }
707
708   \seq_if_empty:NTF \l_tmpa_seq {
709     \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
710   }{
711     \str_set:Nx \l_stex_current_docns_str {
712       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
713     }
714   }
715 }
716 }
717
718 \str_new:N \l_stex_current_docurl_str
719 \cs_new_protected:Nn \stex_get_document_url: {
720   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
721   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
722   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
723   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
724   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
725
726   \str_clear:N \l_tmpa_str
727   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
728     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
729       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
730     }
731   }
732
733   \str_if_empty:NTF \l_tmpa_str {
734     \str_set:Nx \l_stex_current_docurl_str {
735       file:/\stex_path_to_string:N \l_tmpa_seq
736     }
737   }{
738     \bool_set_true:N \l_tmpa_bool

```

```

738 \bool_while_do:Nn \l_tmpa_bool {
739   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
740   \exp_args:No \str_case:nnTF { \l_tmpb_str } {
741     {source} { \bool_set_false:N \l_tmpa_bool }
742   }{}{
743     \seq_if_empty:NT \l_tmpa_seq {
744       \bool_set_false:N \l_tmpa_bool
745     }
746   }
747 }
748
749 \seq_if_empty:NTF \l_tmpa_seq {
750   \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
751 }{
752   \str_set:Nx \l_stex_current_docurl_str {
753     \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
754   }
755 }
756 }
757 }

```

## 20.2 Setting Reference Targets

```

758 \str_const:Nn \c__stex_refs_url_str{URL}
759 \str_const:Nn \c__stex_refs_ref_str{REF}
760 % @currentlabel -> number
761 % @currentlabelname -> title
762 % @currentHref -> name.number <- id of some kind
763 % \theH# -> \arabic{section}
764 % \the# -> number
765 % \hyper@makecurrent{#}
766 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
767   \stex_get_document_uri:
768   \str_set:Nx \l_tmpa_str { #1 }
769   \str_if_empty:NT \l_tmpa_str {
770     \int_zero:N \l_tmpa_int
771     \bool_set_true:N \l_tmpa_bool
772     \bool_while_do:Nn \l_tmpa_bool {
773       \cs_if_exist:cTF {
774         sref_\l_stex_current_docns_str\c_hash_str REF_\int_use:N \l_tmpa_int _type
775       }{
776         \int_incr:N \l_tmpa_int
777       }{
778         \str_set:Nx \l_tmpa_str { REF_\int_use:N \l_tmpa_int }
779         \bool_set_false:N \l_tmpa_bool
780       }
781     }
782   }
783   \str_set:Nx \l_tmpa_str {
784     \l_stex_current_docns_str\c_hash_str\l_tmpa_str
785   }
786   \seq_gput_right:No \g__stex_refs_all_refs_seq \l_tmpa_str
787   \stex_if_smsmode:TF {
788     \stex_get_document_url:

```

```

789 \str_gset_eq:cN {sref_url_\l_tmpa_str_str}\l_stex_current_docurl_str
790 \str_gset_eq:cN {sref_\l_tmpa_str_type}\c__stex_refs_url_str
791 }{
792 \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter{\@currentlabel\iffalse}}
793 \exp_after:wN\label\exp_after:wN{sref_\l_tmpa_str}
794 \str_gset:cn {sref_\l_tmpa_str_type}\c__stex_refs_ref_str
795 }
796 }

797 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
798 \str_gset_eq:cN {sref_sym_#1_uri} \l_stex_current_docns_str
799 }

```

## 20.3 Using References

```

800 \str_new:N \l__stex_refs_indocument_str
801 \keys_define:nn { stex / sref } {
802   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
803   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
804   pre           .tl_set:N = \l__stex_refs_pre_tl ,
805   post          .tl_set:N = \l__stex_refs_post_tl ,
806   %indoc        .str_set_x:N = \l__stex_refs_repo_str ,
807 }
808
809 \bool_new:N \c__stex_refs_hyperref_bool
810 \bool_set_false:N \c__stex_refs_hyperref_bool
811 \AddToHook{begindocument}{
812   \@ifpackageloaded{hyperref}{
813     \bool_set_true:N \c__stex_refs_hyperref_bool
814   }{}
815 }
816
817
818 \cs_new_protected:Nn \__stex_refs_args:n {
819   \tl_clear:N \l__stex_refs_linktext_tl
820   \tl_clear:N \l__stex_refs_fallback_tl
821   \tl_clear:N \l__stex_refs_pre_tl
822   \tl_clear:N \l__stex_refs_post_tl
823   \str_clear:N \l__stex_refs_repo_str
824   \keys_set:nn { stex / sref } { #1 }
825 }
826
827 \NewDocumentCommand \sref { 0{} m}{
828   \__stex_refs_args:n { #1 }
829   \str_if_empty:NTF \l__stex_refs_indocument_str {
830     \str_set:Nn \l_tmpa_str { #2 }
831     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
832     \tl_set:Nn \l_tmpa_tl {
833       \l__stex_refs_fallback_tl
834     }
835     \seq_map_inline:Nn \g__stex_refs_all_refs_seq {
836       \str_set:Nn \l_tmpb_str { ##1 }
837       \str_if_eq:eeT { \l_tmpa_str } {
838         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int }{-1 }
839       } {

```

```

840     \seq_map_break:n {
841       \tl_set:Nn \l_tmpa_tl {
842         % doc uri in \l_tmpb_str
843         \str_set:Nx \l_tmpa_str {sref_url_\l_tmpb_str_type}
844         \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
845           % reference
846           \l__stex_refs_pre_tl\ref{sref_\l_tmpb_str}\l__stex_refs_post_tl
847         }{
848           % URL
849           \if_bool:N \c__stex_refs_hyperref_bool {
850             \exp_args:Nx \href{\use:c{sref_url_\l_tmpb_str_str}}{\l__stex_refs_fallback
851           }{
852             \l__stex_refs_fallback_tl
853           }
854         }
855       }
856     }
857   }
858 }
859 \l_tmpa_tl
860 }{
861   % TODO
862 }
863 }
864
865 </package>

```



## Chapter 21

# STEX -Modules Implementation

```
866 <*package>
867
868 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
869
870 <@@=stex_modules>
871
872   Warnings and error messages
873   \msg_new:nnn{stex}{error/unknownmodule}{
874     No~module~#1~found
875   }
876   \msg_new:nnn{stex}{error/syntax}{
877     Syntax~error:~#1
878   }
879   \msg_new:nnn{stex}{error/siglanguage}{
880     Module~#1~declares~signature~#2,~but~does~not~
881     declare~its~language
882   }
```

`\l_stex_current_module_prop` The current module:

```
881 \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

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

```
883 \seq_new:N \g_stex_modules_in_file_seq
884 \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
885 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
886   \prop_if_empty:NTF \l_stex_current_module_prop
887   \prg_return_false: \prg_return_true:
888 }

```

(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:nTF
889 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
890   \prop_if_exist:cTF { c_stex_module_#1_prop }
891   \prg_return_true: \prg_return_false:
892 }

```

(End definition for \stex\_if\_module\_exists:nTF. This function is documented on page 16.)

```

\stex_add_to_current_module:n
\STEXexport
893 \cs_new_protected:Nn \stex_add_to_current_module:n {
894   \prop_get:NnN \l_stex_current_module_prop { content } \l_tmpa_tl
895   \tl_put_right:Nn \l_tmpa_tl { #1 }
896   \prop_put:Nno \l_stex_current_module_prop { content } { \l_tmpa_tl }
897 }
898 \cs_new_protected:Npn \STEXexport {
899   \begingroup
900   \newlinechar=-1\relax
901   \endlinechar=-1\relax
902   %\catcode'\ = 9\relax
903   \expandafter\endgroup\STEXexport:n
904 }
905 \cs_new_protected:Nn \STEXexport:n {
906   \ignorespaces #1
907   \stex_add_to_current_module:n { \ignorespaces #1 }
908   \stex_smsmode_set_codes:
909 }
910 \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
911 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
912   \str_set:Nx \l_tmpa_str { #1 }
913   \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
914   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
915   \prop_put:Nno \l_stex_current_module_prop { constants } \l_tmpa_seq
916 }

```

(End definition for \stex\_add\_constant\_to\_current\_module:n. This function is documented on page 16.)

```

\stex_add_import_to_current_module:n
917 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
918   \str_set:Nx \l_tmpa_str { #1 }
919   \prop_get:NnN \l_stex_current_module_prop { imports } \l_tmpa_seq
920   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
921   \prop_put:Nno \l_stex_current_module_prop { imports } \l_tmpa_seq
922 }

```

(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).

```

923 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
924   \str_set:Nx \l_tmpa_str { #1 }
925   \seq_set_eq:NN \l_tmpa_seq #2
926   % split off file extension
927   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
928   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
929   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
930   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
931
932   \bool_set_true:N \l_tmpa_bool
933   \bool_while_do:Nn \l_tmpa_bool {
934     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
935     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
936       {source} { \bool_set_false:N \l_tmpa_bool }
937     }{}{
938       \seq_if_empty:NT \l_tmpa_seq {
939         \bool_set_false:N \l_tmpa_bool
940       }
941     }
942   }
943
944   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
945   \str_if_empty:NTF \l_stex_modules_subpath_str {
946     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
947   }{
948     \str_set:Nx \l_stex_modules_ns_str {
949       \l_tmpa_str/\l_stex_modules_subpath_str
950     }
951   }
952 }

```

(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`

```

953 \str_new:N \l_stex_modules_ns_str
954 \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.

```

955 \cs_new_protected:Nn \stex_modules_current_namespace: {
956   \str_clear:N \l_stex_modules_subpath_str
957   \prop_get:NnTF \l_stex_current_repository_prop { ns } \l_tmpa_str {
958     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
959   }{
960     % split off file extension
961     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
962     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str

```

```

963 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
964 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
965 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
966 \str_set:Nx \l_stex_modules_ns_str {
967   file:/\stex_path_to_string:N \l_tmpa_seq
968 }
969 }
970 }

```

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

## 21.1 The module environment

module arguments:

```

971 \keys_define:nn { stex / module } {
972   title      .str_set_x:N = \l_stex_module_title_str ,
973   ns         .str_set_x:N = \l_stex_module_ns_str ,
974   lang       .str_set_x:N = \l_stex_module_lang_str ,
975   sig        .str_set_x:N = \l_stex_module_sig_str ,
976   creators   .str_set_x:N = \l_stex_module_creators_str ,
977   contributors .str_set_x:N = \l_stex_module_contributors_str ,
978   meta       .str_set_x:N = \l_stex_module_meta_str ,
979   srccite    .str_set_x:N = \l_stex_module_srccite_str
980 }
981
982 \cs_new_protected:Nn \__stex_modules_args:n {
983   \str_clear:N \l_stex_module_title_str
984   \str_clear:N \l_stex_module_ns_str
985   \str_clear:N \l_stex_module_lang_str
986   \str_clear:N \l_stex_module_sig_str
987   \str_clear:N \l_stex_module_creators_str
988   \str_clear:N \l_stex_module_contributors_str
989   \str_clear:N \l_stex_module_meta_str
990   \str_clear:N \l_stex_module_srccite_str
991   \keys_set:nn { stex / module } { #1 }
992 }
993
994 % module parameters here? In the body?
995

```

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

```

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

```

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

```

999 \stex_if_in_module:TF {
1000   % Nested module
1001   \prop_get:NnN \l_stex_current_module_prop
1002     { ns } \l_stex_module_ns_str
1003   \str_set:Nx \l_stex_module_name_str {
1004     \prop_item:Nn \l_stex_current_module_prop

```

```

1005     { name } / \l_stex_module_name_str
1006   }
1007 }{
1008   % not nested:
1009   \str_if_empty:NT \l_stex_module_ns_str {
1010     \stex_modules_current_namespace:
1011     \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1012     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1013       / { \l_stex_module_ns_str }
1014     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1015     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1016       \str_set:Nx \l_stex_module_ns_str {
1017         \stex_path_to_string:N \l_tmpa_seq
1018       }
1019     }
1020   }
1021 }

```

Next, we determine the language of the module:

```

1022 \str_if_empty:NT \l_stex_module_lang_str {
1023   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1024   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1025   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1026   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1027   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1028     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1029       inferred~from~file~name}
1030     \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1031   }
1032 }
1033
1034 \str_if_empty:NF \l_stex_module_lang_str {
1035   \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1036     \l_tmpa_str {
1037     \ltx@ifpackageloaded{babel}{
1038       \exp_args:Nx \selectlanguage { \l_tmpa_str }
1039     }{}
1040   } {
1041     \msg_error:nmx{stex}{error/unknownlanguage}{\l_tmpa_str}
1042   }
1043 }

```

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

```

1044 \str_if_empty:NTF \l_stex_module_sig_str {
1045   \str_clear:N \l_tmpa_str
1046   \seq_clear:N \l_tmpa_seq
1047   \tl_clear:N \l_tmpa_tl
1048   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
1049     name      = \l_stex_module_name_str ,
1050     ns        = \l_stex_module_ns_str ,
1051     imports   = \exp_not:o { \l_tmpa_seq } ,
1052     constants = \exp_not:o { \l_tmpa_seq } ,
1053     content   = \exp_not:o { \l_tmpa_tl } ,

```

```

1054     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1055     lang      = \l_stex_module_lang_str ,
1056     sig       = \l_stex_module_sig_str ,
1057     meta      = \l_stex_module_meta_str
1058   }
1059 }{
1060   \str_if_empty:NT \l_stex_module_lang_str {
1061     \msg_error:nnxx{stex}{error/siglanguage}{
1062       \l_stex_module_ns_str?\l_stex_module_name_str
1063     }\l_stex_module_sig_str}
1064   }
1065
1066   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1067   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1068   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1069   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1070   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1071   \str_set:Nx \l_tmpa_str {
1072     \stex_path_to_string:N \l_tmpa_seq /
1073     \l_tmpa_str . \l_stex_module_sig_str .tex
1074   }
1075   \IfFileExists \l_tmpa_str {
1076     \exp_args:No \stex_in_smsmode:nn { \l_tmpa_str } {
1077       \seq_clear:N \l_stex_all_modules_seq
1078       \prop_clear:N \l_stex_current_module_prop
1079       \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1080       \input { \l_tmpa_str }
1081     }
1082   }{
1083     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1084   }
1085   \stex_activate_module:n {
1086     \l_stex_module_ns_str ? \l_stex_module_name_str
1087   }
1088   \prop_set_eq:Nc \l_stex_current_module_prop {
1089     c_stex_module_
1090     \l_stex_module_ns_str ?
1091     \l_stex_module_name_str
1092     _prop
1093   }
1094 }

```

We load the metatheory:

```

1095   \str_if_empty:NT \l_stex_module_meta_str {
1096     \str_set:Nx \l_stex_module_meta_str {
1097       \c_stex_metatheory_ns_str ? Metatheory
1098     }
1099   }
1100   \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1101     \exp_args:Nx \stex_add_to_current_module:n {
1102       \stex_activate_module:n {\l_stex_module_meta_str}
1103     }
1104     \stex_activate_module:n {\l_stex_module_meta_str}
1105   }

```

1106 }

(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}`

```

1107 \cs_new_protected:Nn \_stex_modules_begin_module:nn {
1108   \stex_reactivate_macro:N \STEXexport
1109   \stex_reactivate_macro:N \importmodule
1110   \stex_reactivate_macro:N \symdecl
1111   \stex_reactivate_macro:N \notation
1112   \stex_reactivate_macro:N \symdef
1113   \stex_module_setup:nn{#1}{#2}
1114
1115   \stex_debug:nn{modules}{
1116     New~module:\\
1117     Namespace:~\l_stex_module_ns_str\\
1118     Name:~\l_stex_module_name_str\\
1119     Language:~\l_stex_module_lang_str\\
1120     Signature:~\l_stex_module_sig_str\\
1121     Metatheory:~\l_stex_module_meta_str\\
1122     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1123   }
1124
1125   \seq_put_right:Nx \l_stex_all_modules_seq {
1126     \l_stex_module_ns_str ? \l_stex_module_name_str
1127   }
1128
1129   \seq_gput_right:Nx \g_stex_modules_in_file_seq
1130     { \l_stex_module_ns_str ? \l_stex_module_name_str }
1131
1132   \stex_if_smsmode:TF {
1133     \stex_smsmode_set_codes:
1134   } {
1135     \begin{stex_annotate_env} {theory} {
1136       \l_stex_module_ns_str ? \l_stex_module_name_str
1137     }
1138
1139     \stex_annotate_invisible:nnn{header}{} {
1140       \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1141       \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1142       \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1143         \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1144       }
1145     }
1146   }
1147   % TODO: Inherit metatheory for nested modules?
1148 }
1149 \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}`

```

1150 \cs_new_protected:Nn \__stex_modules_end_module: {
1151   \str_set:Nx \l_tmpa_str {
1152     c_stex_module_
1153     \prop_item:Nn \l_stex_current_module_prop { ns } ?
1154     \prop_item:Nn \l_stex_current_module_prop { name }
1155     _prop
1156   }
1157   %^^A \prop_new:c { \l_tmpa_str }
1158   \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
1159   \stex_debug:nn{modules}{Closing module~\prop_item:Nn \l_stex_current_module_prop { name }}
1160 }

```

(End definition for \\_\_stex\_modules\_end\_module:.)

**@module** The core environment, with no header

```

1161 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1162 \NewDocumentEnvironment { @module } { 0{} m } {
1163   \par
1164   \__stex_modules_begin_module:nn{#1}{#2}
1165 } {
1166   \__stex_modules_end_module:
1167   \stex_if_smsmode:TF {
1168     \exp_args:Nx \stex_add_to_sms:n {
1169       \prop_gset_from_keyval:cn {
1170         c_stex_module_
1171         \prop_item:Nn \l_stex_current_module_prop { ns } ?
1172         \prop_item:Nn \l_stex_current_module_prop { name }
1173         _prop
1174       } {
1175         name      = \prop_item:cn { \l_tmpa_str } { name } ,
1176         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
1177         imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
1178         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
1179         content   = \prop_item:cn { \l_tmpa_str } { content } ,
1180         file      = \prop_item:cn { \l_tmpa_str } { file } ,
1181         lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
1182         sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
1183         meta      = \prop_item:cn { \l_tmpa_str } { meta }
1184       }
1185     }
1186   }{
1187     \end{stex_annotate_env}
1188   }
1189 }

```

**\stex\_modules\_heading:** Code for document headers

```

1190 \cs_if_exist:NTF \thesection {
1191   \newcounter{module}[section]
1192 }{
1193   \newcounter{module}
1194 }
1195
1196 \bool_if:NT \c_stex_showmods_bool {
1197   \latexml_if:F { \RequirePackage{mdframed} }

```



```

1198 }
1199
1200 \cs_new_protected:Nn \stex_modules_heading: {
1201   \stepcounter{module}
1202   \par
1203   \bool_if:NT \c_stex_showmods_bool {
1204     \noindent{\textbf{Module} ~
1205       \cs_if_exist:NT \thesection {\thesection.}
1206       \themodule ~ [\l_stex_module_name_str]
1207     }
1208     \str_if_empty:NTF \l_stex_module_title_str {
1209       }{
1210         \quad(\l_stex_module_title_str)\hfill
1211       }\par
1212     }
1213     \edef\@currentlabel{Module~\thesection.\themodule~[\l_stex_module_name_str]}
1214     % TODO
1215     \stex_ref_new_doc_target:n \l_stex_module_name_str
1216   }

```

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

Finally:

```

1217 \NewDocumentEnvironment { module } { 0{} m } {
1218   \bool_if:NT \c_stex_showmods_bool {
1219     \begin{mdframed}
1220   }
1221   \begin{@module}[#1]{#2}
1222   \stex_modules_heading:
1223 }{
1224   \end{@module}
1225   \bool_if:NT \c_stex_showmods_bool {
1226     \end{mdframed}
1227   }
1228 }

```

## 21.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n
1229 \NewDocumentCommand \STEXModule { m } {
1230   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1231   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1232   \tl_set:Nn \l_tmpa_tl {
1233     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1234   }
1235   \seq_map_inline:Nn \l_stex_all_modules_seq {
1236     \str_set:Nn \l_tmpb_str { ##1 }
1237     \str_if_eq:eeT { \l_tmpa_str } {
1238       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1239     } {
1240       \seq_map_break:n {
1241         \tl_set:Nn \l_tmpa_tl {
1242           \stex_invoke_module:n { ##1 }
1243         }

```

```

1244     }
1245   }
1246 }
1247 \l_tmpa_tl
1248 }
1249
1250 \cs_new_protected:Nn \stex_invoke_module:n {
1251   \stex_debug:nn{modules}{Invoking~module~#1}
1252   \peek_charcode_remove:NTF ! {
1253     \__stex_modules_invoke_uri:nN { #1 }
1254   } {
1255     \peek_charcode_remove:NTF ? {
1256       \__stex_modules_invoke_symbol:nn { #1 }
1257     } {
1258       \msg_error:nnx{stex}{error/syntax}{
1259         ?~or~!~expected~after~
1260         \c_backslash_str STEXModule{#1}
1261       }
1262     }
1263   }
1264 }
1265
1266 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1267   \str_set:Nn #2 { #1 }
1268 }
1269
1270 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1271   \stex_invoke_symbol:n{#1?#2}
1272 }

```

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

`\stex_activate_module:n`

```

1273 \cs_new_protected:Nn \stex_activate_module:n {
1274   \stex_debug:nn{modules}{Activating~module~#1}
1275   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1276     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1277     \prop_item:cn { c_stex_module_#1_prop } { content }
1278   }
1279 }

```

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

1280 `\</package>`

## Chapter 22

# STEX -Module Inheritance Implementation

```
1281 <*package>
1282
1283 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1284
```

### 22.1 SMS Mode

```
1285 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1286 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1287 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1288 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1289
1290 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1291   \makeatletter
1292   \makeatother
1293   \ExplSyntaxOn
1294   \ExplSyntaxOff
1295 }
1296
1297 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1298   \symdef
1299   \importmodule
1300   \notation
1301   \symdecl
1302   \STEXexport
1303 }
1304
1305 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1306   \tl_to_str:n {
1307     module,
1308     @module
```

```

1309 }
1310 }

```

(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
1311 \bool_new:N \g__stex_smsmode_bool
1312 \bool_set_false:N \g__stex_smsmode_bool
1313 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1314   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1315 }

```

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

`\_stex_smsmode_if_catcodes_p:` Checks whether the SMS mode category code scheme is active.

```

\_stex_smsmode_if_catcodes:TF
1316 \bool_new:N \g__stex_smsmode_catcode_bool
1317 \bool_set_false:N \g__stex_smsmode_catcode_bool
1318 \prg_new_conditional:Nnn \_stex_smsmode_if_catcodes: { p, T, F, TF } {
1319   \bool_if:NTF \g__stex_smsmode_catcode_bool
1320   \prg_return_true: \prg_return_false:
1321 }

```

(End definition for `\_stex_smsmode_if_catcodes:TF`.)

`\stex_smsmode_set_codes:`

```

1322 \cs_new_protected:Nn \stex_smsmode_set_codes: {
1323   \stex_if_smsmode:T {
1324     \_stex_smsmode_if_catcodes:F {
1325       \bool_gset_true:N \g__stex_smsmode_catcode_bool
1326       \exp_after:wN \char_gset_active_eq:NN
1327       \c_backslash_str \_stex_smsmode_cs:
1328       \tex_global:D \char_set_catcode_active:N \
1329       \tex_global:D \char_set_catcode_other:N $
1330       \tex_global:D \char_set_catcode_other:N ^
1331       \tex_global:D \char_set_catcode_other:N _
1332       \tex_global:D \char_set_catcode_other:N &
1333       \tex_global:D \char_set_catcode_other:N ##
1334     }
1335   }
1336 } \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.

```

1337 \cs_new_protected:Nn \_stex_smsmode_unset_codes: {
1338   \_stex_smsmode_if_catcodes:T {
1339     \bool_gset_false:N \g__stex_smsmode_catcode_bool
1340     \exp_after:wN \tex_global:D \exp_after:wN
1341     \char_set_catcode_escape:N \c_backslash_str
1342     \tex_global:D \char_set_catcode_math_toggle:N $
1343     \tex_global:D \char_set_catcode_math_superscript:N ^
1344     \tex_global:D \char_set_catcode_math_subscript:N _
1345     \tex_global:D \char_set_catcode_alignment:N &
1346     \tex_global:D \char_set_catcode_parameter:N ##
1347   }
1348 } \iffalse $ \fi % to make syntax highlighting work again

```

(End definition for `\_stex_smsmode_unset_codes:`.)

`\stex_in_smsmode:nn`

```

1349 \cs_new_protected:Nn \stex_in_smsmode:nn {
1350   \vbox_set:Nn \l_tmpa_box {
1351     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1352     \bool_gset_true:N \g__stex_smsmode_bool
1353     \stex_smsmode_set_codes:
1354     #2
1355     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1356     \stex_if_smsmode:F {
1357       \__stex_smsmode_unset_codes:
1358     }
1359   }
1360   \box_clear:N \l_tmpa_box
1361 }

```

(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:

```

1362 \cs_new_protected:Nn \_stex_smsmode_cs: {
1363   \str_clear:N \l_tmpa_str
1364   \peek_analysis_map_inline:n {
1365     % #1: token (one expansion)
1366     % #2: charcode
1367     % #3 catcode
1368     \token_if_eq_charcode:NNTF ##3 B {
1369       % token is a letter
1370       \exp_args:NNNo \str_put_right:Nn \l_tmpa_str { ##1 }
1371     } {
1372       \str_if_empty:NTF \l_tmpa_str {
1373         % we don't allow (or need) single non-letter CSs
1374         % for now
1375         \peek_analysis_map_break:
1376       }{
1377         \str_if_eq:onTF \l_tmpa_str { begin } {
1378           \peek_analysis_map_break:n {
1379             \exp_after:wN \_stex_smsmode_checkbegin:n ##1
1380           }
1381         } {
1382           \str_if_eq:onTF \l_tmpa_str { end } {
1383             \peek_analysis_map_break:n {
1384               \exp_after:wN \_stex_smsmode_checkend:n ##1
1385             }
1386           } {
1387             \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
1388             \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1389             \g_stex_smsmode_allowedmacros_tl
1390             { \use:c{\l_tmpa_str} } {
1391               \stex_debug:nn{modules}{Executing-1:~\l_tmpa_str}
1392               \peek_analysis_map_break:n {
1393                 \exp_after:wN \l_tmpa_tl ##1
1394               }

```

```

1395     } {
1396         \exp_args:NNo \exp_args:NNo \tl_if_in:NnTF
1397         \g_stex_smsmode_allowedmacros_escape_tl
1398         { \use:c{\l_tmpa_str} } {
1399             \__stex_smsmode_unset_codes:
1400             \stex_debug:nn{modules}{Executing~2:~\l_tmpa_str}
1401             % TODO \__stex_smsmode_rescan_cs:
1402             % \int_compare:nNnTF {##2} = {92} {
1403             %     \peek_analysis_map_break:n {
1404             %         \__stex_smsmode_unset_codes:
1405             %         \__stex_smsmode_rescan_cs:
1406             %     }
1407             % } {
1408             %     \peek_analysis_map_break:n {
1409             %         \exp_after:wN \l_tmpa_tl ##1
1410             %     }
1411             % }
1412         } {
1413             \int_compare:nNnTF {##2} = {92} {
1414                 \peek_analysis_map_break:n { \__stex_smsmode_cs: }
1415             }{
1416                 \peek_analysis_map_break:n { \exp_after:wN\relax ##1 }
1417             }
1418         }
1419     }
1420 }
1421 }
1422 }
1423 }
1424 }
1425 }

```

(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:

```

1426 \cs_new_protected:Nn \__stex_smsmode_rescan_cs: {
1427     \str_clear:N \l_tmpb_str
1428     \peek_analysis_map_inline:n {
1429         \token_if_eq_charcode:NNTF ##3 B {
1430             % token is a letter
1431             \exp_args:NNo \str_put_right:Nn \l_tmpb_str { ##1 }
1432         } {
1433             \peek_analysis_map_break:n {
1434                 \exp_after:wN \use:c \exp_after:wN {
1435                     \exp_after:wN \l_tmpa_str\exp_after:wN
1436                 } \use:c { \l_tmpb_str \exp_after:wN } ##1
1437             }
1438         }
1439     }
1440 }

```

(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.

```

1441 \cs_new_protected:Nn \__stex_smsmode_checkbegin:n {
1442   \str_set:Nn \l_tmpa_str { #1 }
1443   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1444     \__stex_smsmode_unset_codes:
1445     \begin{#1}
1446   }
1447 }
```

(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.

```

1448 \cs_new_protected:Nn \__stex_smsmode_checkend:n {
1449   \str_set:Nn \l_tmpa_str { #1 }
1450   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1451     \end{#1}
1452   }
1453 }
```

(End definition for `\__stex_smsmode_checkend:n`.)

## 22.2 Inheritance

```

1454 <@@=stex_importmodule>
```

`\stex_import_module_uri:nn`

```

1455 \cs_new_protected:Nn \stex_import_module_uri:nn {
1456   \str_set:Nx \l__stex_importmodule_archive_str { #1 }
1457   \str_set:Nn \l__stex_importmodule_path_str { #2 }
1458
1459   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l__stex_importmodule_path_str }
1460   \seq_pop_right:NN \l_tmpb_seq \l__stex_importmodule_name_str
1461   \str_set:Nx \l__stex_importmodule_path_str { \seq_use:Nn \l_tmpb_seq ? }
1462
1463   \stex_modules_current_namespace:
1464   \bool_lazy_all:nTF {
1465     {\str_if_empty_p:N \l__stex_importmodule_archive_str}
1466     {\str_if_empty_p:N \l__stex_importmodule_path_str}
1467     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } }
1468   }{
1469     \str_set_eq:NN \l__stex_importmodule_path_str \l_stex_modules_subpath_str
1470     \str_set_eq:NN \l_stex_module_ns
1471   }{
1472     \str_if_empty:NT \l__stex_importmodule_archive_str {
1473       \prop_if_empty:NF \l_stex_current_repository_prop {
1474         \prop_get:NnN \l_stex_current_repository_prop { id } \l__stex_importmodule_archive_s
1475       }
1476     }
1477     \str_if_empty:NTF \l__stex_importmodule_archive_str {
1478       \str_if_empty:NF \l__stex_importmodule_path_str {
1479         \str_set:Nx \l_stex_module_ns_str {
1480           \l_stex_module_ns_str / \l__stex_importmodule_path_str
1481         }
1482       }
1483     }
```

```

1483   }{
1484     \stex_require_repository:n \l__stex_importmodule_archive_str
1485     \prop_get:cnN { c_stex_mathhub\_l__stex_importmodule_archive_str _manifest_prop } { ns
1486       \l_stex_module_ns_str
1487     \str_if_empty:NF \l__stex_importmodule_path_str {
1488       \str_set:Nx \l_stex_module_ns_str {
1489         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1490       }
1491     }
1492   }
1493 }
1494 }

```

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

<code>\l__stex_importmodule_name_str</code>	Store the return values of <code>\stex_import_module_uri:nn</code> .
<code>\l__stex_importmodule_archive_str</code>	1495 <code>\str_new:N \l__stex_importmodule_name_str</code>
<code>\l__stex_importmodule_path_str</code>	1496 <code>\str_new:N \l__stex_importmodule_archive_str</code>
<code>\l__stex_importmodule_file_str</code>	1497 <code>\str_new:N \l__stex_importmodule_path_str</code>
	1498 <code>\str_new:N \g__stex_importmodule_file_str</code>

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

```

\stex_import_require_module:nnnn    {<ns>} {<archive-ID>} {<path>} {<name>}
1499 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1500   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1501
1502     % archive
1503     \str_set:Nx \l_tmpa_str { #2 }
1504     \str_if_empty:NTF \l_tmpa_str {
1505       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1506     } {
1507       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1508       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1509       \seq_put_right:Nn \l_tmpa_seq { source }
1510     }
1511
1512     % path
1513     \str_set:Nx \l_tmpb_str { #3 }
1514     \str_if_empty:NTF \l_tmpb_str {
1515       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1516
1517       \ltx@ifpackageloaded{babel} {
1518         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1519           { \language } \l_tmpb_str {
1520           \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1521         }
1522       } {
1523         \str_clear:N \l_tmpb_str
1524       }
1525
1526       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1527       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1528         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }

```



```

1529 }{
1530   \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1531   \IfFileExists{ \l_tmpa_str.tex }{
1532     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1533   }{
1534     % try english as default
1535     \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1536     \IfFileExists{ \l_tmpa_str.en.tex }{
1537       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1538     }{
1539       \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1540     }
1541   }
1542 }
1543
1544 } {
1545   \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1546   \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1547
1548   \ltx@ifpackageloaded{babel} {
1549     \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1550       { \language } \l_tmpb_str {
1551       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1552     }
1553   } {
1554     \str_clear:N \l_tmpb_str
1555   }
1556
1557   \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1558
1559   \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1560   \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1561     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1562   }{
1563     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1564     \IfFileExists{ \l_tmpa_str/#4.tex }{
1565       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1566     }{
1567       % try english as default
1568       \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1569       \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1570         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1571       }{
1572         \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1573         \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1574           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1575         }{
1576           \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1577           \IfFileExists{ \l_tmpa_str.tex }{
1578             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1579           }{
1580             % try english as default
1581             \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1582             \IfFileExists{ \l_tmpa_str.en.tex }{

```

```

1583         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1584     }{
1585         \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1586     }
1587 }
1588 }
1589 }
1590 }
1591 }
1592 }
1593
1594 \seq_set_eq:NN \l_tmpa_seq \g_stex_modules_in_file_seq
1595 \seq_clear:N \g_stex_modules_in_file_seq
1596 % \exp_args:Nnx \use:nn {
1597     \exp_args:No \stex_in_smsmode:nn { \g__stex_importmodule_file_str } {
1598         \seq_clear:N \l_stex_all_modules_seq
1599         \prop_clear:N \l_stex_current_module_prop
1600         \str_set:Nx \l_tmpb_str { #2 }
1601         \str_if_empty:NF \l_tmpb_str {
1602             \stex_set_current_repository:n { #2 }
1603         }
1604         \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1605         \input { \g__stex_importmodule_file_str }
1606     }
1607 % }{
1608
1609 % }
1610 \prop_gput:Noo \g_stex_module_files_prop
1611 \g__stex_importmodule_file_str \g_stex_modules_in_file_seq
1612 \seq_set_eq:NN \g_stex_modules_in_file_seq \l_tmpa_seq
1613
1614 \stex_if_module_exists:nF { #1 ? #4 } {
1615     \msg_error:nnx{stex}{error/unknownmodule}{
1616         #1?#4~(in~file~\g__stex_importmodule_file_str)
1617     }
1618 }
1619 }
1620 \stex_activate_module:n { #1 ? #4 }
1621 }

```

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

## `\importmodule`

```

1622 \NewDocumentCommand \importmodule { 0{} m } {
1623     \stex_import_module_uri:nn { #1 } { #2 }
1624     \stex_debug:nn{modules}{Importing~module:~
1625         \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1626     }
1627     \stex_if_smsmode:F {
1628         \stex_import_require_module:nnnn
1629         { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1630         { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1631         \stex_annotate_invisible:nnn
1632         {import} { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } {}

```

```

1633 }
1634 \exp_args:Nx \stex_add_to_current_module:n {
1635   \stex_import_require_module:nnnn
1636   { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1637   { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1638 }
1639 \exp_args:Nx \stex_add_import_to_current_module:n {
1640   \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1641 }
1642 \stex_smsmode_set_codes:
1643 }
1644 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

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

### `\usemodule`

```

1645 \NewDocumentCommand \usemodule { 0{} m } {
1646   \stex_if_smsmode:F {
1647     \stex_import_module_uri:nn { #1 } { #2 }
1648     \stex_import_require_module:nnnn
1649     { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1650     { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1651     \stex_annotate_invisible:nnn
1652     {usemodule} { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } {}
1653   }
1654   \stex_smsmode_set_codes:
1655 }

```

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

```

1656 </package>

```

## Chapter 23

# STEX -Symbols Implementation

```
1657 <*package>
1658
1659 %%%%%%%%%% symbols.dtx %%%%%%%%%%
1660
Warnings and error messages
1661
```

### 23.1 Symbol Declarations

```
1662 <@@=stex_symdecl>

\l_stex_all_symbols_seq Stores all available symbols
1663 \seq_new:N \l_stex_all_symbols_seq

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

\STEXsymbol
1664 \NewDocumentCommand \STEXsymbol { m } {
1665   \stex_get_symbol:n { #1 }
1666   \exp_args:No
1667   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1668 }

(End definition for \STEXsymbol. This function is documented on page 27.)
symdecl arguments:
1669 \keys_define:nn { stex / symdecl } {
1670   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1671   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1672   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1673   type      .tl_set:N = \l_stex_symdecl_type_tl ,
1674   align     .str_set:N = \l_stex_symdecl_align_str , % TODO(?)
1675   gfc       .str_set:N = \l_stex_symdecl_gfc_str , % TODO(?)
1676   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
1677   def       .tl_set:N = \l_stex_symdecl_definiens_tl
1678 }
```

```

1679
1680 \bool_new:N \l_stex_symdecl_make_macro_bool
1681
1682 \cs_new_protected:Nn \__stex_symdecl_args:n {
1683   \str_clear:N \l_stex_symdecl_name_str
1684   \str_clear:N \l_stex_symdecl_args_str
1685   \bool_set_false:N \l_stex_symdecl_local_bool
1686   \tl_clear:N \l_stex_symdecl_type_tl
1687   \tl_clear:N \l_stex_symdecl_definiens_tl
1688
1689   \keys_set:nn { stex / symdecl } { #1 }
1690 }

```

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

```

1691
1692 \NewDocumentCommand \symdecl { s O{} m } {
1693   \__stex_symdecl_args:n { #2 }
1694   \IfBooleanTF #1 {
1695     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1696   } {
1697     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1698   }
1699   \stex_symdecl_do:n { #3 }
1700   \stex_smsmode_set_codes:
1701 }
1702 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

**\stex\_symdecl\_do:n**

```

1703 \cs_new_protected:Nn \stex_symdecl_do:n {
1704   \stex_if_in_module:F {
1705     % TODO throw error? some default namespace?
1706   }
1707
1708   \str_if_empty:NT \l_stex_symdecl_name_str {
1709     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1710   }
1711
1712   \prop_if_exist:cT { g_stex_symdecl_
1713     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1714     \prop_item:Nn \l_stex_current_module_prop {name} ?
1715     \l_stex_symdecl_name_str
1716     _prop
1717   }{
1718     % TODO throw error (beware of circular dependencies)
1719   }
1720
1721   \prop_clear:N \l_tmpa_prop
1722   \prop_put:Nnx \l_tmpa_prop { module } {
1723     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1724     \prop_item:Nn \l_stex_current_module_prop {name}
1725   }

```

```

1726 \seq_clear:N \l_tmpa_seq
1727 \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
1728 \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1729 \prop_put:Nno \l_tmpa_prop { local } \l_stex_symdecl_local_bool
1730 \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1731
1732 \exp_args:No \stex_add_constant_to_current_module:n {
1733   \l_stex_symdecl_name_str
1734 }
1735
1736 % arity/args
1737 \int_zero:N \l_tmpb_int
1738
1739 \bool_set_true:N \l_tmpa_bool
1740 \str_map_inline:Nn \l_stex_symdecl_args_str {
1741   \token_case_meaning:NnF ##1 {
1742     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1743     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1744     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1745     {\tl_to_str:n a} {
1746       \bool_set_false:N \l_tmpa_bool
1747       \int_incr:N \l_tmpb_int
1748     }
1749     {\tl_to_str:n B} {
1750       \bool_set_false:N \l_tmpa_bool
1751       \int_incr:N \l_tmpb_int
1752     }
1753   }{
1754     \msg_set:nnn{stex}{error/wrongargs}{
1755       args~value~in~symbol~declaration~for~
1756       \prop_item:Nn \l_stex_current_module_prop {ns} ?
1757       \prop_item:Nn \l_stex_current_module_prop {name} ?
1758       \l_stex_symdecl_name_str ~
1759       needs~to~be~
1760       i,~a,~b~or~B,~but~##1~given
1761     }
1762     \msg_error:nn{stex}{error/wrongargs}
1763   }
1764 }
1765 \bool_if:NTF \l_tmpa_bool {
1766   % possibly numeric
1767   \str_if_empty:NTF \l_stex_symdecl_args_str {
1768     \prop_put:Nnn \l_tmpa_prop { args } {}
1769     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1770   }{
1771     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1772     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1773     \str_clear:N \l_tmpa_str
1774     \int_step_inline:nn \l_tmpa_int {
1775       \str_put_right:Nn \l_tmpa_str i
1776     }
1777     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1778   }
1779 } {

```

```

1780 \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1781 \prop_put:Nnx \l_tmpa_prop { arity }
1782 { \str_count:N \l_stex_symdecl_args_str }
1783 }
1784 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
1785
1786
1787 % semantic macro
1788
1789 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1790 \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1791 \prop_item:Nn \l_tmpa_prop { module } ?
1792 \prop_item:Nn \l_tmpa_prop { name }
1793 } }
1794
1795 \bool_if:NF \l_stex_symdecl_local_bool {
1796 \exp_args:Nx \stex_add_to_current_module:n {
1797 \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1798 \prop_item:Nn \l_tmpa_prop { module } ?
1799 \prop_item:Nn \l_tmpa_prop { name }
1800 } }
1801 }
1802 }
1803 }
1804
1805 % add to all symbols
1806
1807 \bool_if:NF \l_stex_symdecl_local_bool {
1808 \exp_args:Nx \stex_add_to_current_module:n {
1809 \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1810 \prop_item:Nn \l_tmpa_prop { module } ?
1811 \prop_item:Nn \l_tmpa_prop { name }
1812 }
1813 }
1814 }
1815
1816 \stex_debug:nn{symbols}{New~symbol:~
1817 \prop_item:Nn \l_tmpa_prop { module } ?
1818 \prop_item:Nn \l_tmpa_prop { name } ^^J
1819 Type:~\exp_not:o { \l_stex_symdecl_type_tl } ^^J
1820 Args:~\prop_item:Nn \l_tmpa_prop { args }
1821 }
1822
1823 % circular dependencies require this:
1824
1825 \prop_if_exist:cF {
1826 g_stex_symdecl_
1827 \prop_item:Nn \l_tmpa_prop { module } ?
1828 \prop_item:Nn \l_tmpa_prop { name }
1829 _prop
1830 } {
1831 \prop_gset_eq:cN {
1832 g_stex_symdecl_
1833 \prop_item:Nn \l_tmpa_prop { module } ?

```

```

1834     \prop_item:Nn \l_tmpa_prop { name }
1835     _prop
1836   } \l_tmpa_prop
1837 }
1838
1839 \stex_if_smsmode:TF {
1840   \bool_if:NF \l_stex_symdecl_local_bool {
1841     \exp_args:Nx \stex_add_to_sms:n {
1842       \prop_gset_from_keyval:cn {
1843         g_stex_symdecl_
1844         \prop_item:Nn \l_tmpa_prop { module } ?
1845         \prop_item:Nn \l_tmpa_prop { name }
1846         _prop
1847       } {
1848         name      = \prop_item:Nn \l_tmpa_prop { name }      ,
1849         module    = \prop_item:Nn \l_tmpa_prop { module }    ,
1850         notations = \prop_item:Nn \l_tmpa_prop { notations } ,
1851         local     = \prop_item:Nn \l_tmpa_prop { local }     ,
1852         type      = \prop_item:Nn \l_tmpa_prop { type }      ,
1853         args      = \prop_item:Nn \l_tmpa_prop { args }      ,
1854         arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
1855         assocs    = \prop_item:Nn \l_tmpa_prop { assocs }
1856       }
1857       \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1858         \prop_item:Nn \l_tmpa_prop { module } ?
1859         \prop_item:Nn \l_tmpa_prop { name }
1860       }
1861     }
1862   }
1863 }{
1864   \exp_args:NNx \seq_put_right:Nn \l_stex_all_symbols_seq {
1865     \prop_item:Nn \l_tmpa_prop { module } ?
1866     \prop_item:Nn \l_tmpa_prop { name }
1867   }
1868   \stex_if_do_html:T {
1869     \stex_annotate_invisible:nnn {symdecl} {
1870       \prop_item:Nn \l_tmpa_prop { module } ?
1871       \prop_item:Nn \l_tmpa_prop { name }
1872     } {
1873       \stex_annotate_invisible:nnn{type}{}{\l_stex_symdecl_type_tl$}
1874       \stex_annotate_invisible:nnn{args}{}{
1875         \prop_item:Nn \l_tmpa_prop { args }
1876       }
1877       \stex_annotate_invisible:nnn{macroname}{}{#1}
1878       \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
1879         \stex_annotate_invisible:nnn{definiens}{}
1880         {\l_stex_symdecl_definiens_tl$}
1881       }
1882     }
1883   }
1884 }
1885 }

```

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



`\stex_get_symbol:n`

```
1886 \str_new:N \l_stex_get_symbol_uri_str
1887
1888 \cs_new_protected:Nn \stex_get_symbol:n {
1889   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
1890     \__stex_symdecl_get_symbol_from_cs:n { #1 }
1891   }{
1892     % argument is a string
1893     % is it a command name?
1894     \cs_if_exist:cTF { #1 }{
1895       \cs_set_eq:Nc \l_tmpa_tl { #1 }
1896       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
1897       \str_if_empty:NTF \l_tmpa_str {
1898         \exp_args:Nx \cs_if_eq:NNTF {
1899           \tl_head:N \l_tmpa_tl
1900         } \stex_invoke_symbol:n {
1901           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
1902         }{
1903           \__stex_symdecl_get_symbol_from_string:n { #1 }
1904         }
1905       } {
1906         \__stex_symdecl_get_symbol_from_string:n { #1 }
1907       }
1908     }{
1909       % argument is not a command name
1910       \__stex_symdecl_get_symbol_from_string:n { #1 }
1911       % \l_stex_all_symbols_seq
1912     }
1913   }
1914 }
1915
1916 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
1917   \str_set:Nn \l_tmpa_str { #1 }
1918   \bool_set_false:N \l_tmpa_bool
1919   \stex_if_in_module:T {
1920     \prop_get:NnN \l_stex_current_module_prop
1921     { constants } \l_tmpa_seq
1922     \exp_args:NNo \seq_if_in:NnT \l_tmpa_seq { \l_tmpa_str } {
1923       \bool_set_true:N \l_tmpa_bool
1924       \str_set:Nx \l_stex_get_symbol_uri_str {
1925         \prop_item:Nn \l_stex_current_module_prop { ns } ?
1926         \prop_item:Nn \l_stex_current_module_prop { name } ? #1
1927       }
1928     }
1929   }
1930   \bool_if:NF \l_tmpa_bool {
1931     \tl_set:Nn \l_tmpa_tl {
1932       \msg_set:nnn{stex}{error/unknownsymbol}{
1933         No~symbol~#1~found!
1934       }
1935     }
1936     \msg_error:nn{stex}{error/unknownsymbol}
1937   }
1938   \str_set:Nn \l_tmpa_str { #1 }
1939   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
```

```

1939 \seq_map_inline:Nn \l_stex_all_symbols_seq {
1940   \str_set:Nn \l_tmpb_str { ##1 }
1941   \str_if_eq:eeT { \l_tmpa_str } {
1942     \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1943   } {
1944     \seq_map_break:n {
1945       \tl_set:Nn \l_tmpa_tl {
1946         \str_set:Nn \l_stex_get_symbol_uri_str {
1947           ##1
1948         }
1949       }
1950     }
1951   }
1952 }
1953 \l_tmpa_tl
1954 }
1955 }
1956
1957 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
1958   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
1959   { \tl_tail:N \l_tmpa_tl }
1960   \tl_if_single:NTF \l_tmpa_tl {
1961     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
1962       \exp_after:wN \str_set:Nn \exp_after:wN
1963       \l_stex_get_symbol_uri_str \l_tmpa_tl
1964     }{
1965       % TODO
1966       % tail is not a single group
1967     }
1968   }{
1969     % TODO
1970     % tail is not a single group
1971   }
1972 }

```

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

## 23.2 Notations

```

1973 <@@=stex_notation>
      notation arguments:
1974 \keys_define:nn { stex / notation } {
1975   lang      .tl_set_x:N = \l__stex_notation_lang_str ,
1976   variant   .tl_set_x:N = \l__stex_notation_variant_str ,
1977   prec      .str_set_x:N = \l__stex_notation_prec_str ,
1978   op        .tl_set:N   = \l__stex_notation_op_tl ,
1979   unknown   .code:n     = \str_set:Nx
1980             \l__stex_notation_variant_str \l_keys_key_str
1981 }
1982
1983 \cs_new_protected:Nn \__stex_notation_args:n {
1984   \str_clear:N \l__stex_notation_lang_str
1985   \str_clear:N \l__stex_notation_variant_str

```

```

1986 \str_clear:N \l__stex_notation_prec_str
1987 \tl_clear:N \l__stex_notation_op_tl
1988
1989 \keys_set:nn { stex / notation } { #1 }
1990 }

```

## **\notation**

```

1991 \NewDocumentCommand \notation { 0{ } m } {
1992   \__stex_notation_args:n { #1 }
1993   \tl_clear:N \l_stex_symdecl_definiens_tl
1994   \stex_get_symbol:n { #2 }
1995   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
1996 }
1997 \stex_deactivate_macro:Nn \notation {module~environments}

```

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

## **\stex\_notation\_do:nn**

```

1998 \cs_new_protected:Nn \stex_notation_do:nn {
1999   \prop_set_eq:Nc \l_tmpa_prop {
2000     g_stex_symdecl_ #1 _prop
2001   }
2002
2003   \prop_clear:N \l_tmpb_prop
2004   \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
2005   \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
2006   \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
2007
2008   % precedences
2009   \seq_clear:N \l_tmpb_seq
2010   \exp_args:NNno
2011   \str_if_empty:NTF \l__stex_notation_prec_str {
2012     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2013     \int_compare:nNnTF \l_tmpa_str = 0 {
2014       \exp_args:NNnx
2015       \prop_put:Nno \l_tmpb_prop { opprec }
2016       { \neginfprec }
2017     }{
2018       \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
2019     }
2020   } {
2021     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2022       \exp_args:NNnx
2023       \prop_put:Nno \l_tmpb_prop { opprec }
2024       { \neginfprec }
2025       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2026       \int_step_inline:nn { \l_tmpa_str } {
2027         \exp_args:NNx
2028         \seq_put_right:Nn \l_tmpb_seq { \infprec }
2029       }
2030     }{
2031       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2032       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2033         \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
2034         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {

```

```

2035         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2036         \l_tmpa_seq {\tl_to_str:n{x} } { \l_tmpa_str }
2037         \seq_map_inline:Nn \l_tmpa_seq {
2038             \seq_put_right:Nn \l_tmpb_seq { ##1 }
2039         }
2040     }
2041     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2042 }{
2043     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2044     \int_compare:nNnTF \l_tmpa_str = 0 {
2045         \exp_args:NNnx
2046         \prop_put:Nno \l_tmpb_prop { opprec }
2047         { \infprec }
2048     }{
2049         \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
2050     }
2051 }
2052 }
2053 }
2054
2055 \seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
2056 \int_step_inline:nn { \l_tmpa_str } {
2057     \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
2058         \exp_args:NNx
2059         \seq_put_right:Nn \l_tmpb_seq {
2060             \prop_item:Nn \l_tmpb_prop { opprec }
2061         }
2062     }
2063 }
2064
2065 \prop_put:Nno \l_tmpb_prop { argprec } \l_tmpb_seq
2066 \tl_clear:N \l_tmpa_tl
2067
2068 \int_compare:nNnTF \l_tmpa_str = 0 {
2069     \exp_args:NNe
2070     \cs_set:Npn \l__stex_notation_macrocode_cs {
2071         \_stex_term_math_oms:nnnn { #1 }
2072         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2073         { \prop_item:Nn \l_tmpb_prop { opprec } }
2074         { \exp_not:n { #2 } }
2075     }
2076     \__stex_notation_final:
2077 }{
2078     \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
2079     \str_if_in:NnTF \l_tmpb_str b {
2080         \exp_args:Nne \use:nn
2081         {
2082             \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2083             \cs_set:Npn \l_tmpa_str { {
2084                 \_stex_term_math_omb:nnnn { #1 }
2085                 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2086                 { \prop_item:Nn \l_tmpb_prop { opprec } }
2087                 { \exp_not:n { #2 } }
2088             }}

```

```

2089   }{
2090     \str_if_in:NnTF \l_tmpb_str B {
2091       \exp_args:Nne \use:nn
2092       {
2093         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2094         \cs_set:Npn \l_tmpa_str } { {
2095           \stex_term_math_omb:nnnn { #1 }
2096           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2097           { \prop_item:Nn \l_tmpb_prop { opprec } }
2098           { \exp_not:n { #2 } }
2099         } }
2100       }{
2101         \exp_args:Nne \use:nn
2102         {
2103           \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2104           \cs_set:Npn \l_tmpa_str } { {
2105             \stex_term_math_oma:nnnn { #1 }
2106             { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2107             { \prop_item:Nn \l_tmpb_prop { opprec } }
2108             { \exp_not:n { #2 } }
2109           } }
2110       }
2111     }
2112
2113     \int_zero:N \l_tmpa_int
2114     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2115     \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2116     \__stex_notation_arguments:
2117   }
2118 }

```

(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

```

2119 \cs_new_protected:Nn \__stex_notation_arguments: {
2120   \int_incr:N \l_tmpa_int
2121   \str_if_empty:NnTF \l_tmpa_str {
2122     \__stex_notation_final:
2123   }{
2124     \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2125     \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2126     \str_if_eq:NnTF \l_tmpb_str a {
2127       \__stex_notation_argument_assoc:n
2128     }{
2129       \str_if_eq:NnTF \l_tmpb_str B {
2130         \__stex_notation_argument_assoc:n
2131       }{
2132         \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2133         \tl_put_right:Nx \l_tmpa_tl {
2134           { \stex_term_math_arg:nnn
2135             { \int_use:N \l_tmpa_int }
2136             { \l_tmpb_str }
2137             { ####\int_use:N \l_tmpa_int }
2138           }

```

```

2139     }
2140     \__stex_notation_arguments:
2141   }
2142 }
2143 }
2144 }

```

(End definition for \\_\_stex\_notation\_arguments:.)

\\_\_stex\_notation\_argument\_assoc:n

```

2145 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
2146   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2147   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
2148   \tl_put_right:Nx \l_tmpa_tl {
2149     { \stex_term_math_assoc_arg:nnnn
2150       { \int_use:N \l_tmpa_int }
2151       { \l_tmpb_str }
2152       \exp_args:No \exp_not:n
2153       {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
2154       { ####\int_use:N \l_tmpa_int }
2155     }
2156   }
2157   \__stex_notation_arguments:
2158 }

```

(End definition for \\_\_stex\_notation\_argument\_assoc:n.)

\\_\_stex\_notation\_final: Called after processing all notation arguments

```

2159 \cs_new_protected:Nn \__stex_notation_final: {
2160   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
2161   \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
2162   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2163   \exp_args:Nne \use:nn
2164   {
2165     \cs_generate_from_arg_count:cNnn {
2166       stex_notation_ \l_tmpa_str \c_hash_str
2167       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2168       _cs
2169     }
2170     \cs_gset:Npn \l_tmpb_str { { {
2171       \exp_after:wN \exp_after:wN \exp_after:wN
2172       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2173       { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2174     } } }
2175
2176     \tl_if_empty:NF \l__stex_notation_op_tl {
2177       \cs_gset:cpx {
2178         stex_op_notation_ \l_tmpa_str \c_hash_str
2179         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2180         _cs
2181       } {
2182         \stex_term_oms:nnn {
2183           \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2184           \l__stex_notation_lang_str

```

```

2185     }{
2186         \l_tmpa_str
2187     }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2188 }
2189 }
2190
2191
2192
2193 \stex_debug:nn{symbols}{
2194     Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2195     ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
2196     Operator~precedence:~
2197     \prop_item:Nn \l_tmpb_prop { opprec }^^J
2198     Argument~precedences:~
2199     \seq_use:Nn \l_tmpa_seq {,~}^^J
2200     Notation: \cs_meaning:c {
2201         stex_notation_ \l_tmpa_str \c_hash_str
2202         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2203         _cs
2204     }
2205 }
2206
2207 \prop_gset_eq:cN {
2208     g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2209     \c_hash_str \l__stex_notation_lang_str _prop
2210 } \l_tmpb_prop
2211
2212 \exp_args:Nx
2213 \stex_add_to_current_module:n {
2214     \prop_get:cnN {
2215         g_stex_symdecl_
2216         \prop_item:Nn \l_tmpb_prop { symbol }
2217         _prop
2218     } { notations } \exp_not:N \l_tmpa_seq
2219     \seq_put_right:Nn \exp_not:N \l_tmpa_seq {
2220         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2221     }
2222     \prop_put:cno {
2223         g_stex_symdecl_
2224         \prop_item:Nn \l_tmpb_prop { symbol }
2225         _prop
2226     } { notations } \exp_not:N \l_tmpa_seq
2227 }
2228
2229 \stex_if_smsmode:TF {
2230     \stex_smsmode_set_codes:
2231     \exp_args:Nx \stex_add_to_sms:n {
2232         \prop_gset_from_keyval:cn {
2233             g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2234             \c_hash_str \l__stex_notation_lang_str _prop
2235         } {
2236             symbol = \prop_item:Nn \l_tmpb_prop { symbol } ,
2237             language = \prop_item:Nn \l_tmpb_prop { language } ,
2238             variant = \prop_item:Nn \l_tmpb_prop { variant } ,

```

```

2239         opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2240         argprec     = \prop_item:Nn \l_tmpb_prop { argprec }     ,
2241     }
2242 }
2243 }{
2244   \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
2245   \seq_put_right:Nx \l_tmpa_seq {
2246     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2247   }
2248   \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
2249   \prop_set_eq:cN {
2250     g_stex_symdecl_ \l_tmpa_str _prop
2251   } \l_tmpa_prop
2252
2253   % HTML annotations
2254   \stex_if_do_html:T {
2255     \stex_annotate_invisible:nnn { notation }
2256     { \prop_item:Nn \l_tmpb_prop { symbol } } {
2257       \stex_annotate_invisible:nnn { notationfragment }
2258       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
2259       \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2260       \stex_annotate_invisible:nnn { precedence }
2261       { \prop_item:Nn \l_tmpb_prop { opprec } ;
2262         \seq_use:Nn \l_tmpa_seq { x }
2263       }{}
2264
2265       \int_zero:N \l_tmpa_int
2266       \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2267       \tl_clear:N \l_tmpa_tl
2268       \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
2269         \int_incr:N \l_tmpa_int
2270         \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2271         \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2272         \str_if_eq:VnTF \l_tmpb_str a {
2273           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2274             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2275             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2276           } }
2277         }{
2278           \str_if_eq:VnTF \l_tmpb_str B {
2279             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2280               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2281               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2282             } }
2283           }{
2284             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2285               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2286             } }
2287           }
2288         }
2289       }
2290       \stex_annotate_invisible:nnn { notationcomp }{}{
2291         $ \exp_args:Nno \use:nn { \use:c {
2292           stex_notation_ \prop_item:Nn \l_tmpb_prop { symbol }

```



```

2293         \c_hash_str \l__stex_notation_variant_str
2294         \c_hash_str \l__stex_notation_lang_str_cs
2295     } } { \l_tmpa_tl } $
2296   }
2297 }
2298 }
2299 }
2300 }

```

(End definition for `\__stex_notation_final:`)

**\symdef**

```

2301 \keys_define:nn { stex / symdef } {
2302   name      .str_set:x:N = \l_stex_symdecl_name_str ,
2303   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2304   args      .str_set:x:N = \l_stex_symdecl_args_str ,
2305   type      .tl_set:N   = \l_stex_symdecl_type_tl ,
2306   def       .tl_set:N   = \l_stex_symdecl_definiens_tl ,
2307   op        .tl_set:N   = \l__stex_notation_op_tl ,
2308   lang      .str_set:x:N = \l__stex_notation_lang_str ,
2309   variant   .str_set:x:N = \l__stex_notation_variant_str ,
2310   prec      .str_set:x:N = \l__stex_notation_prec_str ,
2311   unknown   .code:n     = \str_set:Nx
2312             \l__stex_notation_variant_str \l_keys_key_str
2313 }
2314
2315 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2316   \str_clear:N \l_stex_symdecl_name_str
2317   \str_clear:N \l_stex_symdecl_args_str
2318   \bool_set_false:N \l_stex_symdecl_local_bool
2319   \tl_clear:N \l_stex_symdecl_type_tl
2320   \tl_clear:N \l_stex_symdecl_definiens_tl
2321   \str_clear:N \l__stex_notation_lang_str
2322   \str_clear:N \l__stex_notation_variant_str
2323   \str_clear:N \l__stex_notation_prec_str
2324   \tl_clear:N \l__stex_notation_op_tl
2325
2326   \keys_set:nn { stex / symdef } { #1 }
2327 }
2328
2329 \NewDocumentCommand \symdef { 0{} m } {
2330   \__stex_notation_symdef_args:n { #1 }
2331   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2332   \stex_symdecl_do:n { #2 }
2333   \exp_args:Nx \stex_notation_do:nn {
2334     \prop_item:Nn \l_tmpa_prop { module } ?
2335     \prop_item:Nn \l_tmpa_prop { name }
2336   }
2337 }
2338 \stex_deactivate_macro:Nn \symdef {module~environments}

```

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

```

2339 \endpackage

```

## Chapter 24

# STEX -Terms Implementation

```
2340 <*package>
2341
2342 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2343
2344 <@@=stex_terms>
2345
2346   Warnings and error messages
2347 \msg_new:nnn{stex}{error/nonotation}{
2348   Symbol~#1~invoked,~but~has~no~notation~#2!
2349 }
2350 \msg_new:nnn{stex}{error/notationarg}{
2351   Error~in~parsing~notation~#1
2352 }
2353 \msg_new:nnn{stex}{error/noop}{
2354   Symbol~#1~has~no~operator~notation~for~notation~#2
2355 }
```

### 24.1 Symbol Invocations

Arguments:

```
2355 \keys_define:nn { stex / terms } {
2356   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2357   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2358   unknown .code:n = \str_set:Nx
2359     \l__stex_terms_variant_str \l_keys_key_str
2360 }
2361
2362 \cs_new_protected:Nn \__stex_terms_args:n {
2363   \str_clear:N \l__stex_terms_lang_str
2364   \str_clear:N \l__stex_terms_variant_str
2365   \str_clear:N \l__stex_terms_prec_str
2366   \tl_clear:N \l__stex_terms_op_tl
2367 }
2368 \keys_set:nn { stex / terms } { #1 }
```

2369 }

**\stex\_invoke\_symbol:n** Invokes a semantic macro

```
2370 \cs_new_protected:Nn \stex_invoke_symbol:n {
2371   \if_mode_math:
2372     \exp_after:wN \__stex_terms_invoke_math:n
2373   \else:
2374     \exp_after:wN \__stex_terms_invoke_text:n
2375   \fi: { #1 }
2376 }
```

(End definition for \stex\_invoke\_symbol:n. This function is documented on page 27.)

**\\_\_stex\_terms\_invoke\_math:n**

```
2377 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2378   \peek_charcode_remove:NTF ! {
2379     \peek_charcode:NTF [ {
2380       \__stex_terms_invoke_op:nw { #1 }
2381     }{
2382       \peek_charcode_remove:NTF ! {
2383         \peek_charcode:NTF [ {
2384           \__stex_terms_invoke_op_custom:nw
2385         }{
2386           % TODO throw error
2387         }
2388       }{
2389         \__stex_terms_invoke_op:nw { #1 } []
2390       }
2391     }
2392   }{
2393     \peek_charcode_remove:NTF * {
2394       \__stex_terms_invoke_text:n { #1 }
2395     }{
2396       \peek_charcode:NTF [ {
2397         \__stex_terms_invoke_math:nw { #1 }
2398       }{
2399         \__stex_terms_invoke_math:nw { #1 } []
2400       }
2401     }
2402   }
2403 }
```

(End definition for \\_\_stex\_terms\_invoke\_math:n.)

**\\_\_stex\_terms\_invoke\_op\_custom:nw**

```
2404 \cs_new_protected:Npn \__stex_terms_invoke_op_custom:nw #1 [#2] {
2405   \stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
2406     \stex_highlight_term:nn{#1}{#2}
2407   }
2408 }
```

(End definition for \\_\_stex\_terms\_invoke\_op\_custom:nw.)

\\_stex\_terms\_invoke\_op:nw

```

2409 \cs_new_protected:Npn \_stex_terms_invoke_op:nw #1 [#2] {
2410   \_stex_terms_args:n { #2 }
2411   \cs_if_exist:cTF {
2412     stex_op_notation_ #1 \c_hash_str
2413     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2414   }{
2415     \csname stex_op_notation_ #1 \c_hash_str
2416       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2417     \endcsname
2418   }{
2419     \msg_error:nnxx{stex}{error/noop}{#1}{\l__stex_terms_variant_str \c_hash_str \l__stex_te
2420   }
2421 }

```

(End definition for \\_stex\_terms\_invoke\_op:nw.)

\\_stex\_terms\_invoke\_math:nw

```

2422 \cs_new_protected:Npn \_stex_terms_invoke_math:nw #1 [#2] {
2423   \_stex_terms_args:n { #2 }
2424   \prop_set_eq:Nc \l_tmpa_prop {
2425     g_stex_symdecl_ #1 _prop
2426   }
2427   \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
2428   \seq_if_empty:NTF \l_tmpa_seq {
2429     \msg_error:nnxn{stex}{error/nonotation}{#1}{s}
2430   } {
2431     \seq_if_in:NxTF \l_tmpa_seq
2432     { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
2433       \use:c{
2434         stex_notation_ #1 \c_hash_str
2435         \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2436         _cs
2437       }
2438     }{
2439       \str_if_empty:NTF \l__stex_terms_variant_str {
2440         \str_if_empty:NTF \l__stex_terms_lang_str {
2441           \seq_get_left:NN \l_tmpa_seq \l_tmpa_str
2442           \use:c{
2443             stex_notation_ #1 \c_hash_str \l_tmpa_str
2444             _cs
2445           }
2446         }{
2447           \msg_error:nnxx{stex}{error/nonotation}{#1}{
2448             ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2449           }
2450         }
2451       }{
2452         \msg_error:nnxx{stex}{error/nonotation}{#1}{
2453           ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2454         }
2455       }
2456     }
2457 }

```

```
2458 }
(End definition for \_stex_terms_invoke_math:nw.)
```

```
\_stex_terms_invoke_text:n
2459 \cs_new_protected:Nn \_stex_terms_invoke_text:n {
2460   \peek_charcode_remove:NTF ! {
2461     \stex_term_custom:nn { #1 } { }
2462   }{
2463     \prop_set_eq:Nc \l_tmpa_prop {
2464       g_stex_symdecl_ #1 _prop
2465     }
2466     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2467     \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2468   }
2469 }
(End definition for \_stex_terms_invoke_text:n.)
```

## 24.2 Terms

Precedences:

```
\infprec
\neginfprec
\l__stex_terms_downprec
2470 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2471 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2472 \int_new:N \l__stex_terms_downprec
2473 \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
2474 \tl_set:Nn \l__stex_terms_left_bracket_str (
2475 \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

```
2476 \cs_new_protected:Nn \_stex_terms_maybe_brackets:nn {
2477   \bool_if:NTF \l__stex_terms_brackets_done_bool {
2478     \bool_set_false:N \l__stex_terms_brackets_done_bool
2479     #2
2480   } {
2481     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
2482       \bool_if:NTF \l_stex_inarray_bool { #2 }{
2483         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
2484         \dobrackets { #2 }
2485       }
2486     }{ #2 }
2487   }
2488 }
```

(End definition for `\_stex_terms_maybe_brackets:nn`.)

### **\dobrackets**

```
2489 \bool_new:N \l__stex_terms_brackets_done_bool
2490 %\RequirePackage{scalerel}
2491 \cs_new_protected:Npn \dobrackets #1 {
2492   %\ThisStyle{\if D\m@switch
2493   %    \exp_args:Nnx \use:nn
2494   %    { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
2495   %    { \exp_not:N\right\l__stex_terms_right_bracket_str }
2496   % \else
2497   \exp_args:Nnx \use:nn
2498   {
2499     \bool_set_true:N \l__stex_terms_brackets_done_bool
2500     \int_set:Nn \l__stex_terms_downprec \infpref
2501     \l__stex_terms_left_bracket_str
2502     #1
2503   }
2504   {
2505     \bool_set_false:N \l__stex_terms_brackets_done_bool
2506     \l__stex_terms_right_bracket_str
2507     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2508   }
2509   %\fi}
2510 }
```

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

### **\withbrackets**

```
2511 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2512   \exp_args:Nnx \use:nn
2513   {
2514     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
2515     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
2516     #3
2517   }
2518   {
2519     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
2520     {\l__stex_terms_left_bracket_str}
2521     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
2522     {\l__stex_terms_right_bracket_str}
2523   }
2524 }
```

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

### **\STEXinvisible**

```
2525 \cs_new_protected:Npn \STEXinvisible #1 {
2526   \stex_annotate_invisible:n { #1 }
2527 }
```

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

OMDoc terms:

`\_stex_term_math_oms:nnnn`

```
2528 \cs_new_protected:Nn \_stex_term_oms:nnn {
2529   \stex_annotate:nnn{ OMID }{ #2 }{
2530     \stex_highlight_term:nn { #1 } { #3 }
2531   }
2532 }
2533
2534 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2535   \__stex_terms_maybe_brackets:nn { #3 }{
2536     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2537   }
2538 }
```

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

`\_stex_term_math_oma:nnnn`

```
2539 \cs_new_protected:Nn \_stex_term_oma:nnn {
2540   \stex_annotate:nnn{ OMA }{ #2 }{
2541     \stex_highlight_term:nn { #1 } { #3 }
2542   }
2543 }
2544
2545 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2546   \__stex_terms_maybe_brackets:nn { #3 }{
2547     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2548   }
2549 }
```

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

`\_stex_term_math_omb:nnnn`

```
2550 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2551   \stex_annotate:nnn{ OMBIND }{ #2 }{
2552     \stex_highlight_term:nn { #1 } { #3 }
2553   }
2554 }
2555
2556 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2557   \__stex_terms_maybe_brackets:nn { #3 }{
2558     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2559   }
2560 }
```

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

`\_stex_term_math_arg:nnn`

```
2561 \cs_new_protected:Nn \_stex_term_arg:nn {
2562   \stex_unhighlight_term:n {
2563     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2564   }
2565 }
2566 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
2567   \exp_args:Nnx \use:nn
2568     { \int_set:Nn \l__stex_terms_downprec { #2 }

```

```

2569     \stex_term_arg:nn { #1 }{ #3 }
2570   }
2571   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2572 }

```

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

`\stex_term_math_assoc_arg:nnnn`

```

2573 \cs_new_protected:Nn \stex_term_math_assoc_arg:nnnn {
2574   \clist_set:Nn \l_tmpa_clist{ #4 }
2575   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
2576     \tl_set:Nn \l_tmpa_tl { #4 }
2577   }{
2578     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
2579     \clist_reverse:N \l_tmpa_clist
2580     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
2581
2582     \clist_map_inline:Nn \l_tmpa_clist {
2583       \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
2584         \exp_args:Nno
2585         \l_tmpa_cs { ##1 } \l_tmpa_tl
2586       }
2587     }
2588
2589   }
2590   \exp_args:Nnno
2591   \stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2592 }

```

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

`\stex_term_custom:nn`

```

2593 \cs_new_protected:Nn \stex_term_custom:nn {
2594   \str_set:Nn \l__stex_terms_custom_uri { #1 }
2595   \str_set:Nn \l_tmpa_str { #2 }
2596   \tl_clear:N \l_tmpa_tl
2597   \int_zero:N \l_tmpa_int
2598   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2599   \__stex_terms_custom_loop:
2600 }

```

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

`\__stex_terms_custom_loop:`

```

2601 \cs_new_protected:Nn \__stex_terms_custom_loop: {
2602   \bool_set_false:N \l_tmpa_bool
2603   \bool_while_do:nn {
2604     \str_if_eq_p:ee X {
2605       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2606     }
2607   }{
2608     \int_incr:N \l_tmpa_int
2609   }
2610
2611   \peek_charcode:NNTF [ {

```



```

2612 % notation/text component
2613 \__stex_terms_custom_component:w
2614 } {
2615 \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2616 % all arguments read => finish
2617 \__stex_terms_custom_final:
2618 } {
2619 % arguments missing
2620 \peek_charcode_remove:NTF * {
2621 % invisible, specific argument position or both
2622 \peek_charcode:NTF [ {
2623 % visible specific argument position
2624 \__stex_terms_custom_arg:wn
2625 } {
2626 % invisible
2627 \peek_charcode_remove:NTF * {
2628 % invisible specific argument position
2629 \__stex_terms_custom_arg_inv:wn
2630 } {
2631 % invisible next argument
2632 \__stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2633 }
2634 }
2635 } {
2636 % next normal argument
2637 \__stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
2638 }
2639 }
2640 }
2641 }

```

(End definition for \\_\_stex\_terms\_custom\_loop:.)

\\_\_stex\_terms\_custom\_arg\_inv:wn

```

2642 \cs_new_protected:Npn \__stex_terms_custom_arg_inv:wn [ #1 ] #2 {
2643 \bool_set_true:N \l_tmpa_bool
2644 \__stex_terms_custom_arg:wn [ #1 ] { #2 }
2645 }

```

(End definition for \\_\_stex\_terms\_custom\_arg\_inv:wn.)

\\_\_stex\_terms\_custom\_arg:wn

```

2646 \cs_new_protected:Npn \__stex_terms_custom_arg:wn [ #1 ] #2 {
2647 \str_set:Nx \l_tmpb_str {
2648 \str_item:Nn \l_tmpa_str { #1 }
2649 }
2650 \str_case:VnTF \l_tmpb_str {
2651 { X } {
2652 \msg_error:nnx{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2653 }
2654 { i } { \__stex_terms_custom_set_X:n { #1 } }
2655 { b } { \__stex_terms_custom_set_X:n { #1 } }
2656 { a } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
2657 { B } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
2658 }{}{

```

```

2659 \msg_error:nxx{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2660 }
2661
2662 \bool_if:nTF \l_tmpa_bool {
2663   \tl_put_right:Nx \l_tmpa_tl {
2664     \stex_annotate_invisible:n {
2665       \stex_term_arg:nn { \int_eval:n { #1 } }
2666       \exp_not:n { { #2 } }
2667     }
2668   }
2669 } {
2670   \tl_put_right:Nx \l_tmpa_tl {
2671     \stex_term_arg:nn { \int_eval:n { #1 } }
2672     \exp_not:n { { #2 } }
2673   }
2674 }
2675
2676 \__stex_terms_custom_loop:
2677 }

```

(End definition for \\_\_stex\_terms\_custom\_arg:wn.)

\\_\_stex\_terms\_custom\_set\_X:n

```

2678 \cs_new_protected:Nn \__stex_terms_custom_set_X:n {
2679   \str_set:Nx \l_tmpa_str {
2680     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
2681     X
2682     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
2683   }
2684 }

```

(End definition for \\_\_stex\_terms\_custom\_set\_X:n.)

\\_\_stex\_terms\_custom\_component:

```

2685 \cs_new_protected:Npn \__stex_terms_custom_component:w [ #1 ] {
2686   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
2687   \__stex_terms_custom_loop:
2688 }

```

(End definition for \\_\_stex\_terms\_custom\_component:.)

\\_\_stex\_terms\_custom\_final:

```

2689 \cs_new_protected:Nn \__stex_terms_custom_final: {
2690   \int_compare:nNnTF \l_tmpb_int = 0 {
2691     \exp_args:Nnno \stex_term_oms:nnn
2692   } {
2693     \str_if_in:NnTF \l_tmpa_str {b} {
2694       \exp_args:Nnno \stex_term_ombind:nnn
2695     } {
2696       \exp_args:Nnno \stex_term_oma:nnn
2697     }
2698   }
2699   { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
2700 }

```

(End definition for `\_stex_terms_custom_final:`.)

`\symref`

`\symname`

```

2701 \NewDocumentCommand \symref { m m }{
2702   \let\compemph_uri_prev:\compemph@uri
2703   \let\compemph@uri\symrefemph@uri
2704   \STEXsymbol{#1}! [#2]
2705   \let\compemph@uri\compemph_uri_prev:
2706 }
2707
2708 \keys_define:nn { stex / symname } {
2709   post      .str_set_x:N   = \l_stex_symname_post_str
2710 }
2711
2712 \cs_new_protected:Nn \stex_symname_args:n {
2713   \str_clear:N \l_stex_symname_post_str
2714   \keys_set:nn { stex / symname } { #1 }
2715 }
2716
2717 \NewDocumentCommand \symname { 0{} m }{
2718   \stex_symname_args:n { #1 }
2719   \stex_get_symbol:n { #2 }
2720   \str_set:Nx \l_tmpa_str {
2721     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
2722   }
2723   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
2724
2725   \let\compemph_uri_prev:\compemph@uri
2726   \let\compemph@uri\symrefemph@uri
2727   \exp_args:NNx \use:nn
2728   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }! [
2729     \l_tmpa_str \l_stex_symname_post_str
2730   ] }
2731   \let\compemph@uri\compemph_uri_prev:
2732 }

```

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

## 24.3 Notation Components

2733 `<@@=stex_notationcomps>`

`\stex_highlight_term:nn`

```

2734
2735 \str_new:N \l__stex_notationcomps_highlight_uri_str
2736 \cs_new_protected:Nn \stex_highlight_term:nn {
2737   \exp_args:Nnx
2738   \use:nn {
2739     \str_set:Nx \l__stex_notationcomps_highlight_uri_str { #1 }
2740     #2
2741   } {
2742     \str_set:Nx \exp_not:N \l__stex_notationcomps_highlight_uri_str
2743     { \l__stex_notationcomps_highlight_uri_str }
2744   }

```

```

2745 }
2746
2747 \cs_new_protected:Nn \stex_unhighlight_term:n {
2748 % \latexml_if:TF {
2749 %   #1
2750 % } {
2751 %   \rustex_if:TF {
2752 %     #1
2753 %   } {
2754 %     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
2755 %   }
2756 % }
2757 }

```

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

```

\comp
\compemph@uri 2758 \cs_new_protected:Npn \comp #1 {
\compemph 2759 \str_if_empty:NF \l__stex_notationcomps_highlight_uri_str {
\defemph 2760 \rustex_if:TF {
\defemph@uri 2761 \stex_annotate:nnn { comp }{ \l__stex_notationcomps_highlight_uri_str }{ #1 }
\symrefemph 2762 }{
\symrefemph@uri 2763 \exp_args:Nnx \compemph@uri { #1 } { \l__stex_notationcomps_highlight_uri_str }
2764 }
2765 }
2766 }
2767
2768 \cs_new_protected:Npn \compemph@uri #1 #2 {
2769 \compemph{ #1 }
2770 }
2771
2772
2773 \cs_new_protected:Npn \compemph #1 {
2774 \textcolor{blue}{#1}
2775 }
2776
2777 \cs_new_protected:Npn \defemph@uri #1 #2 {
2778 \defemph{#1}
2779 }
2780
2781 \cs_new_protected:Npn \defemph #1 {
2782 \textbf{#1}
2783 }
2784
2785 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
2786 \symrefemph{#1}
2787 }
2788
2789 \cs_new_protected:Npn \symrefemph #1 {
2790 \textbf{#1}
2791 }

```

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

**\ellipses**

```
2792 \NewDocumentCommand \ellipses {} { \ldots }
```

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

```
\parray
\prmatrix 2793 \bool_new:N \l_stex_inarray_bool
\parrayline 2794 \bool_set_false:N \l_stex_inarray_bool
\parraylineh 2795 \NewDocumentCommand \parray { m m } {
\parraycell 2796 \begingroup
2797 \bool_set_true:N \l_stex_inarray_bool
2798 \begin{array}{#1}
2799 #2
2800 \end{array}
2801 \endgroup
2802 }
2803
2804 \NewDocumentCommand \prmatrix { m } {
2805 \begingroup
2806 \bool_set_true:N \l_stex_inarray_bool
2807 \begin{matrix}
2808 #1
2809 \end{matrix}
2810 \endgroup
2811 }
2812
2813 \def \maybepline {
2814 \bool_if:NT \l_stex_inarray_bool {\hline}
2815 }
2816
2817 \def \parrayline #1 #2 {
2818 #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
2819 }
2820
2821 \def \pmrow #1 { \parrayline{}{ #1 } }
2822
2823 \def \parraylineh #1 #2 {
2824 #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
2825 }
2826
2827 \def \parraycell #1 {
2828 #1 \bool_if:NT \l_stex_inarray_bool {&}
2829 }
```

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

```
2830 \endpackage
```

## Chapter 25

# STEX -Structural Features Implementation

```
2831 <*package>
2832
2833 %%%%%%%%%%% features.dtx %%%%%%%%%%%
2834
2835 <@@=stex_features>
      Warnings and error messages
2836
```

### 25.1 The feature environment

structural@feature

```
2837
2838 \NewDocumentEnvironment{structural@feature}{ m m m }{
2839   \stex_if_in_module:F {
2840     \msg_set:nnn{stex}{error/nomodule}{
2841       Structural~Feature~has~to~occur~in~a~module:\\
2842       Feature~#2~of~type~#1\\
2843       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
2844     }
2845     \msg_error:nn{stex}{error/nomodule}
2846   }
2847
2848   \str_set:Nx \l_stex_module_name_str {
2849     \prop_item:Nn \l_stex_current_module_prop
2850       { name } / #2 - feature
2851   }
2852
2853   \str_set:Nx \l_stex_module_ns_str {
2854     \prop_item:Nn \l_stex_current_module_prop
2855       { ns }
2856   }
2857
```

```

2858
2859 \str_clear:N \l_tmpa_str
2860 \seq_clear:N \l_tmpa_seq
2861 \tl_clear:N \l_tmpa_tl
2862 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
2863   origname = #2,
2864   name     = \l_stex_module_name_str ,
2865   ns       = \l_stex_module_ns_str ,
2866   imports  = \exp_not:o { \l_tmpa_seq } ,
2867   constants = \exp_not:o { \l_tmpa_seq } ,
2868   content  = \exp_not:o { \l_tmpa_tl } ,
2869   file     = \exp_not:o { \g_stex_currentfile_seq } ,
2870   lang     = \l_stex_module_lang_str ,
2871   sig      = \l_tmpa_str ,
2872   meta     = \l_tmpa_str ,
2873   feature  = #1 ,
2874 }
2875
2876 \stex_if_smsmode:TF {
2877   \stex_smsmode_set_codes:
2878 } {
2879   \begin{stex_annotate_env}{ feature:#1 }{}
2880   \stex_annotate_invisible:nnn{header}{}{ #3 }
2881 }
2882 }{
2883   \str_set:Nx \l_tmpa_str {
2884     c_stex_feature_
2885     \prop_item:Nn \l_stex_current_module_prop { ns } ?
2886     \prop_item:Nn \l_stex_current_module_prop { name }
2887     _prop
2888   }
2889   \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
2890   \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
2891   \stex_if_smsmode:TF {
2892     \exp_args:Nx \stex_add_to_sms:n {
2893       \prop_gset_from_keyval:cn {
2894         c_stex_feature_
2895         \prop_item:Nn \l_stex_current_module_prop { ns } ?
2896         \prop_item:Nn \l_stex_current_module_prop { name }
2897         _prop
2898       } {
2899         origname = #2,
2900         name     = \prop_item:cn { \l_tmpa_str } { name } ,
2901         ns       = \prop_item:cn { \l_tmpa_str } { ns } ,
2902         imports  = \prop_item:cn { \l_tmpa_str } { imports } ,
2903         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
2904         content  = \prop_item:cn { \l_tmpa_str } { content } ,
2905         file     = \prop_item:cn { \l_tmpa_str } { file } ,
2906         lang     = \prop_item:cn { \l_tmpa_str } { lang } ,
2907         sig      = \prop_item:cn { \l_tmpa_str } { sig } ,
2908         meta     = \prop_item:cn { \l_tmpa_str } { meta } ,
2909         feature  = \prop_item:cn { \l_tmpa_str } { feature }
2910       }
2911     }

```

```

2912 } {
2913     \end{stex_annotate_env}
2914 }
2915 }
2916

```

## 25.2 Features

structure

```

2917
2918 \prop_new:N \l_stex_all_structures_prop
2919
2920 \keys_define:nn { stex / features / structure } {
2921     name .str_set_x:N = \l__stex_features_structure_name_str ,
2922 }
2923
2924 \cs_new_protected:Nn \__stex_features_structure_args:n {
2925     \str_clear:N \l__stex_features_structure_name_str
2926     \keys_set:nn { stex / features / structure } { #1 }
2927 }
2928
2929 %\stex_new_feature:nnnn { structure } { 0{ } m } {
2930 % \__stex_features_structure_args:n { ##1 }
2931 % \str_if_empty:NT \l__stex_features_structure_name_str {
2932 %     \str_set:Nx \l__stex_features_structure_name_str { ##2 }
2933 % }
2934 %} {
2935 %
2936 %}
2937
2938 \NewDocumentEnvironment{mathstructure}{ 0{ } m }{
2939     \__stex_features_structure_args:n { #1 }
2940     \str_if_empty:NT \l__stex_features_structure_name_str {
2941         \str_set:Nx \l__stex_features_structure_name_str { #2 }
2942     }
2943     \exp_args:Nnnx
2944     \begin{structural@feature}{ structure }
2945         { \l__stex_features_structure_name_str }{}
2946         \seq_clear:N \l_tmpa_seq
2947         \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
2948     }{
2949         \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
2950         \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
2951         \str_set:Nx \l_tmpa_str {
2952             \prop_item:Nn \l_stex_current_module_prop { ns } ?
2953             \prop_item:Nn \l_stex_current_module_prop { name }
2954         }
2955     }
2956     \seq_map_inline:Nn \l_tmpa_seq {
2957         \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
2958     }
2959     \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
2960     \exp_args:Nnx

```



```

2961 \AddToHookNext { env / mathstructure / after }{
2962 \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}{
2963 \_stex_term_math_oms:nnnn { \l_tmpa_str }{}{0}{}}
2964 }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
2965 \STEXexport {
2966 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2967 {\prop_item:Nn \l_stex_current_module_prop { origname }}
2968 {\l_tmpa_str}
2969 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2970 {#2}{\l_tmpa_str}
2971 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
2972 % \prop_item:Nn \l_stex_current_module_prop { origname },
2973 % \l_tmpa_str
2974 % }
2975 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
2976 % #2,\l_tmpa_str
2977 % }
2978 % \tl_set:cx { #2 } {
2979 % \stex_invoke_structure:n { \l_tmpa_str }
2980 }
2981 }
2982
2983 \end{structural@feature}
2984 % \g_stex_last_feature_prop
2985 }

```

\instantiate

```

2986 \seq_new:N \l__stex_features_structure_field_seq
2987 \str_new:N \l__stex_features_structure_field_str
2988 \str_new:N \l__stex_features_structure_def_tl
2989 \prop_new:N \l__stex_features_structure_prop
2990 \NewDocumentCommand \instantiate { m O{} m }{
2991 \stex_smsmode_set_codes:
2992 \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
2993 \prop_set_eq:Nc \l__stex_features_structure_prop {
2994 c_stex_feature_\l_tmpa_str _prop
2995 }
2996 \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
2997 \seq_map_inline:Nn \l__stex_features_structure_field_seq {
2998 \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
2999 \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3000 \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
3001 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
3002 {!} \l_tmpa_tl
3003 \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3004 \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
3005 \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3006 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3007 }{
3008 \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
3009 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3010 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
3011 \l_tmpa_tl
3012 \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {

```

```

3013         \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
3014         \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3015     }{
3016         \tl_clear:N \l_tmpb_tl
3017     }
3018 }
3019 }{
3020     \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
3021     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3022         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
3023         \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
3024         \tl_clear:N \l_tmpa_tl
3025     }{
3026         % TODO throw error
3027     }
3028 }
3029 % \l_tmpa_str: name
3030 % \l_tmpa_tl: definiens
3031 % \l_tmpb_tl: notation
3032 \tl_if_empty:NT \l__stex_features_structure_field_str {
3033     % TODO throw error
3034 }
3035 \str_clear:N \l_tmpb_str
3036
3037 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3038 \seq_map_inline:Nn \l_tmpa_seq {
3039     \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
3040     \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
3041     \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
3042         \seq_map_break:n {
3043             \str_set:Nn \l_tmpb_str { ####1 }
3044         }
3045     }
3046 }
3047 \prop_get:cnN { g_stex_symdecl_ \l_tmpb_str _prop } {args}
3048     \l_tmpb_str
3049
3050 \tl_if_empty:NTF \l_tmpb_tl {
3051     \tl_if_empty:NF \l_tmpa_tl {
3052         \exp_args:Nx \use:n {
3053             \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
3054         }
3055     }
3056 }{
3057     \tl_if_empty:NTF \l_tmpa_tl {
3058         \exp_args:Nx \use:n {
3059             \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\
3060         }
3061     }
3062 }{
3063     \exp_args:Nx \use:n {
3064         \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
3065         \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
3066     }

```

```

3067     }
3068   }
3069   % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
3070   % \prop_item:Nn \l_stex_current_module_prop {name} ?
3071   % #3/\l_stex_features_structure_field_str
3072   % \par
3073   % \expandafter\present\csname
3074   %   g_stex_symdecl_
3075   %   \prop_item:Nn \l_stex_current_module_prop {ns} ?
3076   %   \prop_item:Nn \l_stex_current_module_prop {name} ?
3077   %   #3/\l_stex_features_structure_field_str
3078   %   _prop
3079   % \endcsname
3080 }
3081
3082 \tl_clear:N \l__stex_features_structure_def_tl
3083
3084 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3085 \seq_map_inline:Nn \l_tmpa_seq {
3086   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3087   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3088   \exp_args:Nx \use:n {
3089     \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
3090
3091     }
3092   }
3093
3094   \prop_if_exist:cF {
3095     g_stex_symdecl_
3096     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3097     \prop_item:Nn \l_stex_current_module_prop {name} ?
3098     #3/\l_tmpa_str
3099     _prop
3100   }{
3101     \prop_get:cnN { g_stex_symdecl_ ##1 _prop } {args}
3102     \l_tmpb_str
3103     \exp_args:Nx \use:n {
3104       \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
3105     }
3106   }
3107 }
3108
3109 \symdecl*[type={\STEXsymbol{module-type}}{
3110   \_stex_term_math_oms:nnnn {
3111     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3112     \prop_item:Nn \l__stex_features_structure_prop {name}
3113     }{}{0}{}
3114   }{}{#3}
3115
3116   % TODO: -> sms file
3117
3118   \tl_set:cx{ #3 }{
3119     \stex_invoke_structure:nnn {
3120       \prop_item:Nn \l_stex_current_module_prop {ns} ?

```

```

3121     \prop_item:Nn \l_stex_current_module_prop {name} ? #3
3122   } {
3123     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3124     \prop_item:Nn \l__stex_features_structure_prop {name}
3125   }
3126 }
3127
3128 }

```

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

\stex\_invoke\_structure:nnn

```

3129 % #1: URI of the instance
3130 % #2: URI of the instantiated module
3131 \cs_new_protected:Nn \stex_invoke_structure:nnn {
3132   \tl_if_empty:nTF{ #3 }{
3133     \prop_set_eq:Nc \l__stex_features_structure_prop {
3134       c_stex_feature_ #2 _prop
3135     }
3136     \tl_clear:N \l_tmpa_tl
3137     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3138     \seq_map_inline:Nn \l_tmpa_seq {
3139       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3140       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3141       \cs_if_exist:cT {
3142         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
3143       }{
3144         \tl_if_empty:NF \l_tmpa_tl {
3145           \tl_put_right:Nn \l_tmpa_tl {,}
3146         }
3147         \tl_put_right:Nx \l_tmpa_tl {
3148           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
3149         }
3150       }
3151     }
3152     \exp_args:No \mathstrut \l_tmpa_tl
3153   }{
3154     \stex_invoke_symbol:n{#1/#3}
3155   }
3156 }

```

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

```

3157 </package>

```

## Chapter 26

# STEX -Statements Implementation

```
3158 <*package>
3159
3160 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3161
3162 \protected\def\ignorespacesandpars{
3163   \begingroup\catcode13=10\relax
3164   \@ifnextchar\par{
3165     \endgroup\expandafter\ignorespacesandpars\@gobble
3166   }{
3167     \endgroup
3168   }
3169 }
3170
3171 <@@=stex_statements>
3172
3173   Warnings and error messages
3174
3175 \def\titleemph#1{\textbf{#1}}
3176
symboldoc
3177 \NewDocumentEnvironment{symboldoc}{m}{
3178   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3179   \seq_clear:N \l_tmpb_seq
3180   \seq_map_inline:Nn \l_tmpa_seq {
3181     \str_if_eq:nnF{ ##1 }{}{
3182       \stex_get_symbol:n { ##1 }
3183       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3184         \l_stex_get_symbol_uri_str
3185       }
3186     }
3187   }
3188   \par
3189   \exp_args:Nnnx
3190   \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
3191 }{
```

```

3189 \end{stex_annotate_env}
3190 }

3191 \seq_new:N \g_stex_statements_patched_seq
3192
3193 \cs_new_protected:Nn \stex_statements_set_patched:n {
3194   \seq_put_right:Nn \g_stex_statements_patched_seq {#1}
3195 }
3196
3197 \cs_new_protected:Nn \stex_statements_patch:nn {
3198   \seq_if_in:NnF \g_stex_statements_patched_seq {#1} {
3199     \AddToHook{begindocument}{
3200       \cs_if_exist:cTF{end#1}{
3201         \AddToHook{env/#1/before}[stex]{\use:c{__stex_statements_#2_begin:n}{}}
3202         \AddToHook{env/#1/after}[stex]{\use:c{__stex_statements_#2_end:}}
3203       }{
3204         \NewDocumentEnvironment{#1}{0}{
3205           \titleemph{\stex_capitalize:n #1}~
3206           \use:c{__stex_statements_#2_begin:n}{
3207             {
3208               \use:c{__stex_statements_#2_end:}
3209             }
3210           }
3211         }
3212       }
3213     }

```

## 26.1 Definitions

definition

```

3214 \keys_define:nn {stex / definiendum }{
3215   post      .tl_set:N      = \l__stex_statements_definiendum_post_tl,
3216   root      .str_set_x:N   = \l__stex_statements_definiendum_root_str,
3217   gfa       .str_set_x:N   = \l__stex_statements_definiendum_gfa_str
3218 }
3219 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
3220   \str_clear:N \l__stex_statements_definiendum_root_str
3221   \tl_clear:N \l__stex_statements_definiendum_post_tl
3222   \str_clear:N \l__stex_statements_definiendum_gfa_str
3223   \keys_set:nn { stex / definiendum }{ #1 }
3224 }
3225 \NewDocumentCommand \definiendum { O{} m m } {
3226   \__stex_statements_definiendum_args:n { #1 }
3227   \stex_get_symbol:n { #2 }
3228   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3229   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
3230     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
3231       \tl_set:Nn \l_tmpa_tl { #3 }
3232     } {
3233       \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
3234       \tl_set:Nn \l_tmpa_tl {
3235         \l__stex_statements_definiendum_root_str\l__stex_statements_definiendum_post_tl
3236       }

```

```

3237     }
3238   } {
3239     \tl_set:Nn \l_tmpa_tl { #3 }
3240   }
3241
3242   % TODO root
3243   \rustex_if:TF {
3244     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
3245   } {
3246     \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
3247   }
3248 }
3249 \stex_deactivate_macro:Nn \definiendum {definition-environments}
3250
3251 \NewDocumentCommand \definame { 0{ } m } {
3252   \__stex_statements_definiendum_args:n { #1 }
3253   % TODO: root
3254   \stex_get_symbol:n { #2 }
3255   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3256   \str_set:Nx \l_tmpa_str {
3257     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3258   }
3259   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {-}
3260   \rustex_if:TF {
3261     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3262       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3263     }
3264   } {
3265     \defemph@uri {
3266       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3267     } { \l_stex_get_symbol_uri_str }
3268   }
3269 }
3270 \stex_deactivate_macro:Nn \definame {definition-environments}
3271
3272 \cs_new_protected:Nn \__stex_statements_defi_begin:n {
3273   \stex_reactivate_macro:N \definiendum
3274   \stex_reactivate_macro:N \definame
3275   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3276   \seq_clear:N \l_tmpb_seq
3277   \seq_map_inline:Nn \l_tmpa_seq {
3278     \str_if_eq:nnF{ ##1 }{}{
3279       \stex_get_symbol:n { ##1 }
3280       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3281         \l_stex_get_symbol_uri_str
3282       }
3283     }
3284   }
3285   \stex_smsmode_set_codes:
3286   \exp_args:Nnnx
3287   \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
3288 }
3289
3290 \cs_new_protected:Nn \__stex_statements_defi_end: {

```

```

3291 \end{stex_annotate_env}
3292 }

Hook:

3293 \stex_statements_patch:nn{definition}{defi}

inline:

3294 \NewDocumentCommand \inlinedef { m } {
3295   \begingroup
3296   \stex_reactivate_macro:N \definiendum
3297   \stex_reactivate_macro:N \definame
3298   \stex_ref_new_doc_target:n{
3299     #1
3300   }
3301 }

```

## 26.2 Assertions

assertion

```

3302 \cs_new_protected:Nn \__stex_statements_assertion_begin:n {
3303   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3304   \seq_clear:N \l_tmpb_seq
3305   \seq_map_inline:Nn \l_tmpa_seq {
3306     \str_if_eq:nnF{ ##1 }{}{
3307       \stex_get_symbol:n { ##1 }
3308       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3309         \l_stex_get_symbol_uri_str
3310       }
3311     }
3312   }
3313   \stex_smsmode_set_codes:
3314   \exp_args:Nnnx
3315   \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
3316 }
3317
3318 \cs_new_protected:Nn \__stex_statements_assertion_end: {
3319   \end{stex_annotate_env}
3320 }

Hook:

3321 \stex_statements_patch:nn{assertion}{assertion}

inline:

3322 \NewDocumentCommand \inlineass { m } {
3323   \begingroup
3324   \stex_ref_new_doc_target:n{
3325     #1
3326   }
3327 }

```



theorem

```

3328 \cs_new_protected:Nn \__stex_statements_theorem_begin:n {
3329   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3330   \seq_clear:N \l_tmpb_seq
3331   \seq_map_inline:Nn \l_tmpa_seq {
3332     \str_if_eq:nnF{ ##1 }{}{
3333       \stex_get_symbol:n { ##1 }
3334       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3335         \l_stex_get_symbol_uri_str
3336       }
3337     }
3338   }
3339   \stex_smsmode_set_codes:
3340   \exp_args:Nnnx
3341   \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
3342 }
3343
3344 \cs_new_protected:Nn \__stex_statements_theorem_end: {
3345   \end{stex_annotate_env}
3346 }

```

Hook:

```

3347 \stex_statements_patch:nn{theorem}{theorem}

```

lemma

```

3348 \cs_new_protected:Nn \__stex_statements_lemma_begin:n {
3349   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3350   \seq_clear:N \l_tmpb_seq
3351   \seq_map_inline:Nn \l_tmpa_seq {
3352     \str_if_eq:nnF{ ##1 }{}{
3353       \stex_get_symbol:n { ##1 }
3354       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3355         \l_stex_get_symbol_uri_str
3356       }
3357     }
3358   }
3359   \stex_smsmode_set_codes:
3360   \exp_args:Nnnx
3361   \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
3362 }
3363
3364 \cs_new_protected:Nn \__stex_statements_lemma_end: {
3365   \end{stex_annotate_env}
3366 }

```

Hook:

```

3367 \stex_statements_patch:nn{lemma}{lemma}

```

axiom

```

3368 \cs_new_protected:Nn \__stex_statements_axiom_begin:n {
3369   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3370   \seq_clear:N \l_tmpb_seq
3371   \seq_map_inline:Nn \l_tmpa_seq {

```

```

3372 \str_if_eq:nnF{ ##1 }{}{
3373 \stex_get_symbol:n { ##1 }
3374 \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3375 \l_stex_get_symbol_uri_str
3376 }
3377 }
3378 }
3379 \stex_smsmode_set_codes:
3380 \exp_args:Nnnx
3381 \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
3382 }
3383
3384 \cs_new_protected:Nn \__stex_statements_axiom_end: {
3385 \end{stex_annotate_env}
3386 }

Hook:

3387 \stex_statements_patch:nn{axiom}{axiom}

```

## 26.3 Examples

example

```

3388 \cs_new_protected:Nn \__stex_statements_example_begin:n {
3389 \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3390 \seq_clear:N \l_tmpb_seq
3391 \seq_map_inline:Nn \l_tmpa_seq {
3392 \str_if_eq:nnF{ ##1 }{}{
3393 \stex_get_symbol:n { ##1 }
3394 \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3395 \l_stex_get_symbol_uri_str
3396 }
3397 }
3398 }
3399 \stex_smsmode_set_codes:
3400 \exp_args:Nnnx
3401 \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpb_seq {,}}
3402 }
3403
3404 \cs_new_protected:Nn \__stex_statements_example_end: {
3405 \end{stex_annotate_env}
3406 }

Hook:

3407 \stex_statements_patch:nn{example}{example}

inline:

3408 \NewDocumentCommand \inlineex { m } {
3409 \begingroup
3410 \stex_ref_new_doc_target:n{
3411 #1
3412 \endgroup
3413 }

```

## 26.4 OMText

```

3414 \keys_define:nn { stex / omtext } {
3415   id      .str_set_x:N    = \l_stex_omtext_id_str ,
3416   title   .tl_set:N       = \l_stex_omtext_title_tl ,
3417   type    .tl_set_x:N     = \l_stex_omtext_type_tl ,
3418   for     .tl_set_x:N     = \l_stex_omtext_for_tl ,
3419   from    .tl_set_x:N     = \l_stex_omtext_from_tl ,
3420   start   .tl_set:N       = \l_stex_omtext_start_tl ,
3421 }
3422 \cs_new_protected:Nn \stex_omtext_args:n {
3423   \tl_clear:N \l_stex_omtext_title_tl
3424   \tl_clear:N \l_stex_omtext_start_tl
3425   \keys_set:nn { stex / omtext } { #1 }
3426 }
3427 \newif\if@in@omtext\@in@omtextfalse
3428 \NewDocumentEnvironment {omtext} { 0{ } } {
3429   \stex_omtext_args:n { #1 }
3430   \tl_if_empty:NTF \l_stex_omtext_start_tl {
3431     \tl_if_empty:NF \l_stex_omtext_title_tl {
3432       \titleemph{\l_stex_omtext_title_tl}:~
3433     }
3434   }{
3435     \titleemph{\l_stex_omtext_start_tl}~
3436   }
3437   \@in@omtexttrue
3438
3439   \stex_ref_new_doc_target:n \l_stex_omtext_id_str
3440   \stex_smsmode_set_codes:
3441   \ignorespacesandpars
3442 }{}
3443 \</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).<sup>10</sup>

```
3444 \package
3445 \@@=stex_sproof
3446
3447 %%%%%%%%%%%%%%%%% sproof.dtx %%%%%%%%%%%%%%%%%
3448
```

### 27.2 Proofs

We first define some keys for the proof environment.

```
3449 \keys_define:nn { stex / spf } {
3450   id          .str_set:N = \l__stex_sproof_spf_id_str,
3451   display     .tl_set:N  = \l__stex_sproof_spf_display_tl,
3452   for         .tl_set:N  = \l__stex_sproof_spf_for_tl ,
3453   from        .tl_set:N  = \l__stex_sproof_spf_from_tl ,
3454   proofend    .tl_set:N  = \l__stex_sproof_spf_proofend_tl,
3455   type        .tl_set:N  = \l__stex_sproof_spf_type_tl,
3456   title       .tl_set:N  = \l__stex_sproof_spf_title_tl,
3457   continues   .tl_set:N  = \l__stex_sproof_spf_continues_tl,
3458   functions   .tl_set:N  = \l__stex_sproof_spf_functions_tl,
3459   method      .tl_set:N  = \l__stex_sproof_spf_method_tl
3460 }
3461 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
3462   \str_clear:N \l__stex_sproof_spf_id_str
3463   \tl_clear:N \l__stex_sproof_spf_display_tl
3464   \tl_clear:N \l__stex_sproof_spf_for_tl
3465   \tl_clear:N \l__stex_sproof_spf_from_tl
3466   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
3467   \tl_clear:N \l__stex_sproof_spf_type_tl
3468   \tl_clear:N \l__stex_sproof_spf_title_tl
```

---

<sup>10</sup>EDNOTE: need an implementation for L<sup>A</sup>T<sub>E</sub>X<sub>M</sub>L

```

3469 \tl_clear:N \l__stex_sproof_spf_continues_tl
3470 \tl_clear:N \l__stex_sproof_spf_functions_tl
3471 \tl_clear:N \l__stex_sproof_spf_method_tl
3472 \keys_set:nn { stex / spf }{ #1 }
3473 }

```

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

```

3474 \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, L<sup>A</sup>T<sub>E</sub>X 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<sup>6</sup> 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 T<sub>E</sub>X 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.

```

3475 \newcount\count_ten
3476 \newenvironment{pst@with@label}[1]{
3477   \edef\pst@label{#1}
3478   \advance\count_ten by 1\relax
3479   \count_ten=1
3480 }{
3481   \advance\count_ten by -1\relax
3482 }

```

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

```

3483 \def\the@pst@label{
3484   \pst@make@label\pst@label{\number\count_ten}\l__stex_sproof_pstlabel_postfix_tl
3485 }

```

(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.

```

3486 \keys_define:nn { stex / pstlabel }{
3487   prefix      .tl_set:N   = \l__stex_sproof_pstlabel_prefix_tl,
3488   delimiter   .tl_set:N   = \l__stex_sproof_pstlabel_delimiter_tl,
3489   postfix     .tl_set:N   = \l__stex_sproof_pstlabel_postfix_tl
3490 }
3491 \cs_new_protected:Nn \__stex_sproof_pstlabel_args:n {

```

---

<sup>6</sup>This gets the labeling right but only works 8 levels deep

```

3492 \tl_set:Nn \l__stex_sproof_pstlabel_prefix_tl {P}
3493 \tl_set:Nn \l__stex_sproof_pstlabel_delimiter_tl {.}
3494 \tl_clear:N \l__stex_sproof_pstlabel_postfix_tl
3495 }
3496 \__stex_sproof_pstlabel_args:n {}
3497 \newcommand\setpstlabelstyle[1]{
3498   \__stex_sproof_pstlabel_args:n {#1}
3499 }
3500 \newcommand\setpstlabelstyledefault{%
3501   \__stex_sproof_pstlabel_args:n{prefix=P,delimiter=.,postfix={}}
3502 }

```

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

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

```

3503 \ExplSyntaxOff
3504 \def\pst@make@label@long#1#2{\@for\@I:=#1\do{\expandafter\expandafter\expandafter\@I\csname
3505 \def\pst@make@label@angles#1#2{\ensuremath{\@for\@I:=#1\do{\rangle}}#2}
3506 \def\pst@make@label@short#1#2{#2}
3507 \def\pst@make@label@empty#1#2{}
3508 \ExplSyntaxOn
3509 \def\pstlabelstyle#1{%
3510   \def\pst@make@label{\use:c{pst@make@label@#1}}%
3511 }%
3512 \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.

```

3513 \def\next@pst@label{%
3514   \global\advance\count\count10 by 1%
3515 }%

```

(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

```

3516 \def\sproof@box{
3517   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
3518 }
3519 \def\spf@proofend{\sproof@box}
3520 \def\sproofend{
3521   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
3522     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
3523   }
3524 }
3525 \def\sProofEndSymbol#1{\def\sproof@box{#1}}

```

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

**spf@\*@kw**

```

3526 \def\spf@proofsketch@kw{Proof Sketch}
3527 \def\spf@proof@kw{Proof}
3528 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

3529 \cs_if_exist:NT \bbl@loaded {
3530   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
3531   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
3532     \input{proof-ngerman.lda}
3533   }
3534   \clist_if_in:NnT \l_tmpa_clist {finnish}{
3535     \input{proof-finnish.lda}
3536   }
3537   \clist_if_in:NnT \l_tmpa_clist {french}{
3538     \input{proof-french.lda}
3539   }
3540   \clist_if_in:NnT \l_tmpa_clist {russian}{
3541     \input{proof-russian.lda}
3542   }
3543 }
3544
```

**spfsketch**

```

3545 \newcommand\spfsketch[2] [] {
3546   \__stex_sproof_spf_args:n{#1}
3547   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
3548     \titleemph{
3549       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
3550         \spf@proofsketch@kw
3551       }{
3552         \l__stex_sproof_spf_type_tl
3553       }
3554     }:
3555   }
3556   {-#2}
3557   %\sref@label@id{this \ifx\spf@type\@empty\spf@proofsketch@kw\else\spf@type\fi}
3558   \sproofend
3559 }

```

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

**spfeq** This is very similar to `\spfsketch`, but uses a computation array<sup>1112</sup>

```

3560 \newenvironment{spfeq}[2] [] {
3561   \__stex_sproof_spf_args:n{#1}
3562   %\sref@target
3563   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
3564     \titleemph{
3565       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
3566         \spf@proof@kw
3567       }{
3568         \l__stex_sproof_spf_type_tl
3569       }
3570     }:

```

<sup>11</sup>EDNOTE: This should really be more like a tabular with an ensuremath in it. or invoke text on the last column

<sup>12</sup>EDNOTE: document above

```

3571 }
3572 {~#2}
3573 \begin{displaymath}\begin{array}{rcll}
3574 }{
3575 \end{array}\end{displaymath}
3576 }

```

(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.

```

3577 \newenvironment{spf@proof}[2][]{
3578   \__stex_sproof_spf_args:n{#1}
3579   %\sref@target
3580   \count_ten=10
3581   \par\noindent
3582   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3583     \titleemph{
3584       \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {
3585         \spf@proof@kw
3586       }{
3587         \l__stex_sproof_spf_type_tl
3588       }
3589     }:
3590   }
3591   {~#2}
3592   %\sref@label@id{this \ifx\spf@type\empty\spf@proof@kw\else\spf@type\fi}
3593   \def\pst@label{}
3594   \newcount\pst@count% initialize the labeling mechanism
3595   \begin{description}\begin{pst@with@label}{\l__stex_sproof_pstlabel_prefix_tl}
3596   }{
3597     \end{pst@with@label}\end{description}
3598   }
3599   \newenvironment{sproof}[2][{\begin{spf@proof}[#1]{#2}}{\sproofend\end{spf@proof}}
3600   \newenvironment{sProof}[2][{\begin{spf@proof}[#1]{#2}}{\end{spf@proof}}

```

**\spfidea**

```

3601 \newcommand\spfidea[2][]{
3602   \__stex_sproof_spf_args:n{#1}
3603   \titleemph{
3604     \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {Proof~Idea}{
3605       \l__stex_sproof_spf_type_tl
3606     }:
3607   }~#2
3608   \sproofend
3609 }

```

(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.



spfstep 13

```

3610 \newenvironment{spfstep}[1][]{
3611   \_stex_sproof_spf_args:n{#1}
3612   \@in@omtexttrue
3613   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3614     \item[\the@pst@label]
3615   }
3616   \tl_if_empty:NF \l__stex_sproof_spf_title_tl {
3617     {(\titleemph{\l__stex_sproof_spf_title_tl})\enspace}
3618   }
3619   %\sref@label@id{\pst@label}
3620   \ignorespacesandpars
3621 }{
3622   \next@pst@label\ignorespacesandpars
3623 }

```

sproofcomment

```

3624 \newenvironment{sproofcomment}[1][]{
3625   \_stex_sproof_spf_args:n{#1}
3626   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3627     \item[\the@pst@label]
3628   }
3629 }{
3630   \next@pst@label
3631 }

```

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.

```

3632 \newenvironment{subproof}[2][]{
3633   \_stex_sproof_spf_args:n{#1}
3634   \def\@test{#2}
3635   \ifx\@test\empty\else
3636     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3637       \item[\the@pst@label]
3638     }{#2}
3639   \fi
3640   \begin{pst@with@label}{\pst@label,\number\count_ten}
3641 }{
3642   \end{pst@with@label}\next@pst@label
3643 }

```

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

```

3644 \newenvironment{spfcases}[2][]{
3645   \def\@test{#1}
3646   \ifx\@test\empty
3647     \begin{subproof}[method=by-cases]{#2}
3648   \else
3649     \begin{subproof}[#1,method=by-cases]{#2}
3650   \fi
3651 }{

```

---

<sup>13</sup>EdNOTE: MK: labeling of steps does not work yet.

```

3652 \end{subproof}
3653 }

```

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

```

3654 \newenvironment{spfcase}[2] []{
3655   \__stex_sproof_spf_args:n{#1}
3656   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3657     \item[\the@pst@label]
3658   }
3659   \def\@test{#2}
3660   \ifx\@test\@empty
3661   \else
3662     {\titleemph{#2}:~}
3663   \fi
3664   \begin{pst@with@label}{\pst@label,\number\count_ten}
3665   }{
3666     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3667       \sproofend
3668     }
3669     \end{pst@with@label}
3670     \next@pst@label
3671   }

```

**spfcase** similar to **spfcase**, takes a third argument.

```

3672 \newcommand\spfcasesketch[3] []{
3673   \__stex_sproof_spf_args:n{#1}
3674   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3675     \item[\the@pst@label]
3676   }
3677   \def\@test{#2}
3678   \ifx\@test\@empty
3679   \else
3680     {\titleemph{#2}:~}
3681   \fi#3
3682   \next@pst@label
3683 }%

```

## 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.

```

3684 \keys_define:nn { stex / just }{
3685   id          .str_set:x:N = \l__stex_sproof_just_id_str,
3686   method      .tl_set:N    = \l__stex_sproof_just_method_tl,
3687   premises    .tl_set:N    = \l__stex_sproof_just_premises_tl,
3688   args        .tl_set:N    = \l__stex_sproof_just_args_tl
3689 }

```

The next three environments and macros are purely semantic, so we ignore the keyval arguments for now and only display the content.<sup>14</sup>

<sup>14</sup>EDNOTE: need to do something about the premise in draft mode.

**justification**

```
3690 \newenvironment{justification}[1] [] {}{}
```

**\premise**

```
3691 \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.

```
3692 \newcommand\justarg[2] [] {#2}
```

```
3693 \end{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

```
3694 <*package>
3695
3696 %%%%%%%%%% others.dtx %%%%%%%%%%
3697
3698 <@@=stex_others>
3699     Warnings and error messages
3700     % None
3701
3702 \MSC Math subject classifier
3703 \NewDocumentCommand \MSC {m} {
3704     % TODO
3705 }
3706
3707 (End definition for \MSC. This function is documented on page 10.)
3708 Patching tikzinput, if loaded
3709 \ifpackageloaded{tikzinput}{
3710     \RequirePackage{stex-tikzinput}
3711 }{}
3712 </package>
```

## Chapter 29

# STEX -Metatheory Implementation

```
3707 <*package>
3708 <@@=stex_modules>
3709
3710 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
3711
3712 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
3713 \begingroup
3714 \stex_module_setup:nn{
3715   ns=\c_stex_metatheory_ns_str,
3716   meta=NONE
3717 }{Metatheory}
3718 \stex_reactivate_macro:N \symdecl
3719 \stex_reactivate_macro:N \notation
3720 \stex_reactivate_macro:N \symdef
3721 \ExplSyntaxOff
3722 \csname stex_suppress_html:n\endcsname{
3723   % is-a (a:A, a \in A, a is an A, etc.)
3724   \symdecl[args=ai]{isa}
3725   \notation[typed]{isa}{#1 \comp{:} #2}{#1 \comp, #2}
3726   \notation[in]{isa}{#1 \comp\in #2}{#1 \comp, #2}
3727   \notation[pred]{isa}{#2\comp(#1 \comp)}{#1 \comp, #2}
3728
3729   % bind (\forall, \Pi, \lambda etc.)
3730   \symdecl[args=Bi]{bind}
3731   \notation[forall]{bind}{\comp\forall #1.\;#2}{#1 \comp, #2}
3732   \notation[\Pi]{bind}{\comp\prod_{#1}#2}{#1 \comp, #2}
3733   \notation[deffun]{bind}{\comp( #1 \comp{}\;\to\;} #2}{#1 \comp, #2}
3734
3735   % dummy variable
3736   \symdecl{dummyvar}
3737   \notation[underscore]{dummyvar}{\comp\_}
3738   \notation[dot]{dummyvar}{\comp\cdot}
3739   \notation[dash]{dummyvar}{\comp{\rm --}}
3740
3741   %fromto (function space, Hom-set, implication etc.)
```

```

3742 \symdecl[args=ai]{fromto}
3743 \notation[xarrow]{fromto}{#1 \comp\to #2}{#1 \comp\times #2}
3744 \notation[arrow]{fromto}{#1 \comp\to #2}{#1 \comp\to #2}
3745
3746 % mapto (lambda etc.)
3747 %\symdecl[args=Bi]{mapto}
3748 %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
3749 %\notation[lambda]{mapto}{\comp\lambda #1 \comp. \; #2}{#1 \comp, #2}
3750 %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp. \; #2}{#1 \comp, #2}
3751
3752 % function/operator application
3753 \symdecl[args=ia]{apply}
3754 \notation[prec=0;0x\infpres,parens]{apply}{#1 \comp( #2 \comp)}{#1 \comp, #2}
3755 \notation[prec=0;0x\infpres,lambda]{apply}{#1 \; #2 }{#1 \; #2}
3756
3757 % ‘‘type’’ of all collections (sets, classes, types, kinds)
3758 \symdecl{collection}
3759 \notation[U]{collection}{\comp{\mathcal{U}}{}}
3760 \notation[set]{collection}{\comp{\textsf{Set}}{}}
3761
3762 % sequences
3763 \symdecl[args=1]{seqtype}
3764 \notation[kleene]{seqtype}{#1^{\comp\ast}}
3765
3766 \symdef[args=2,li,prec=nobrackets]{sequence-index}{#1_{#2}}
3767 \notation[ui,prec=nobrackets]{sequence-index}{#1^{#2}}
3768
3769 %\symdef[args=3,li]{sequence-from-to}{#1_{#2}\comp{\,\ellipses\,}#1_{#3}}
3770 %\notation[ui]{sequence-from-to}{#1^{#2}\comp{\,\ellipses\,}#1^{#3}}
3771 % ^ superceded by \aseqfromto and \livar/\uivar
3772
3773 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{\,\ellipses\,}}{#1\comp,#2}
3774 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{\,\ellipses\,}#2}{#1\comp,#2}
3775 \symdef[args=aui,prec=nobrackets]{aseqfromtovia}{#1\comp{\,\ellipses\,}#2\comp{\,\ellipses\,}#3}{#1\comp,#2}
3776
3777 % letin (‘‘let’’, local definitions, variable substitution)
3778 \symdecl[args=bii]{letin}
3779 \notation[let]{letin}{\comp{\rm let}}{\;#1\comp{=}\;#2\; \comp{\rm in}}{\;#3}
3780 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
3781 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
3782
3783 % structures
3784 \symdecl*[args=1]{module-type}
3785 \notation{module-type}{\mathtt{MOD} #1}
3786 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
3787 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{#1 \comp, #2}
3788
3789 }
3790 \ExplSyntaxOn
3791 \stex_add_to_current_module:n{
3792   \let\nappa\apply
3793   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
3794   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
3795   \def\livar{\csname sequence-index\endcsname[li]}

```

```

3796 \def\uivar{\csname sequence-index\endcsname[ui]}
3797 \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
3798 \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
3799 \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
3800 }
3801 \__stex_modules_end_module:
3802 \endgroup
3803 \endpackage

```

## Chapter 30

# Tikzinput Implementation

```
3804 <*package>
3805
3806 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
3807
3808 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
3809 \RequirePackage{l3keys2e}
3810
3811 \keys_define:nn { tikzinput } {
3812   image .bool_set:N = \c_tikzinput_image_bool,
3813   image .default:n = false ,
3814   unknown .code:n = {}
3815 }
3816
3817 \ProcessKeysOptions { tikzinput }
3818
3819 \bool_if:NTF \c_tikzinput_image_bool {
3820   \RequirePackage{graphicx}
3821
3822   \providecommand\usetikzlibrary[]{}
3823   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
3824 }{
3825   \RequirePackage{tikz}
3826   \RequirePackage{standalone}
3827
3828   \newcommand \tikzinput [2] [] {
3829     \setkeys{Gin}{#1}
3830     \ifx \Gin@ewidth \Gin@exclamation
3831       \ifx \Gin@eheight \Gin@exclamation
3832         \input { #2 }
3833       \else
3834         \resizebox{!}{ \Gin@eheight }{
3835           \input { #2 }
3836         }
3837       \fi
3838     \else
3839       \ifx \Gin@eheight \Gin@exclamation
3840         \resizebox{ \Gin@ewidth }{!}{
3841           \input { #2 }
```



```

3842     }
3843     \else
3844         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
3845             \input { #2 }
3846         }
3847     \fi
3848 \fi
3849 }
3850 }
3851
3852 \newcommand \ctikzinput [2] [] {
3853     \begin{center}
3854         \tikzinput [1] {#2}
3855     \end{center}
3856 }
3857
3858 \@ifpackageloaded{stex}{
3859     \RequirePackage{stex-tikzinput}
3860 }{}
3861
3862 </package>
3863 <*stex>
3864 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
3865 \RequirePackage{stex}
3866 \RequirePackage{tikzinput}
3867
3868 \newcommand\mhtikzinput [2] [] {%
3869     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
3870     \stex_in_repository:nn\Gin@mhrepos{
3871         \tikzinput[#1]{\mhpath{##1}{#2}}
3872     }
3873 }
3874 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [1] {#2}\end{center}}
3875 </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.

```
3876 \*cls)
3877 \@@=document_structure)
3878 \ProvidesExplClass{omdoc}{2020/10/19}{1.4}{OMDoc Documents}
3879 \RequirePackage{l3keys2e,expl-keystr-compat}
```

### 31.2 Class Options

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.

`\omdoc@cls@class`

```
3880 \keys_define:nn{ document-structure / pkg }{
3881   class      .str_set_x:N = \c_document_structure_class_str,
3882   minimal    .bool_set:N = \c_document_structure_minimal_bool,
3883   report     .code:n      = {
3884     \ClassWarning{omdoc}{the option 'report' is deprecated, use 'class=report', instead}
3885     \str_set:Nn \c_document_structure_class_str {report}
3886   },
3887   book       .code:n      = {
3888     \ClassWarning{omdoc}{the option 'book' is deprecated, use 'class=book', instead}
3889     \str_set:Nn \c_document_structure_class_str {book}
3890   },
3891   bookpart   .code:n      = {
3892     \ClassWarning{omdoc}{the option 'bookpart' is deprecated, use 'class=book,topsect=chapter}
3893     \str_set:Nn \c_document_structure_class_str {book}
3894     \str_set:Nn \c_document_structure_topsect_str {chapter}
3895   },
```

```

3896 docopt      .str_set_x:N = \c_document_structure_docopt_str,
3897 unknown     .code:n      = {
3898   \PassOptionsToPackage{ \CurrentOption }{ omdoc }
3899 }
3900 }
3901 \ProcessKeysOptions{ document-structure / pkg }
3902 \str_if_empty:NT \c_document_structure_class_str {
3903   \str_set:Nn \c_document_structure_class_str {article}
3904 }
3905 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
3906   {\c_document_structure_class_str}
3907

```

### 31.3 Beefing up the document environment

Now, – unless the option `minimal` is defined – we include the `stex` package

```

3908 \RequirePackage{omdoc}
3909 \bool_if:NF \c_document_structure_minimal_bool {
3910   \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 L<sup>A</sup>T<sub>E</sub>X level, but the document identifier is picked up by L<sup>A</sup>T<sub>E</sub>XML.<sup>15</sup>

```

3911 \keys_define:nn { document-structure / document }{
3912   id .str_set_x:N = \c_document_structure_document_id_str
3913 }
3914 \let\__document_structure_orig_document=\document
3915 \renewcommand{\document}[1][]{
3916   \keys_set:nn{ document-structure / document }{ #1 }
3917   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
3918   \__document_structure_orig_document
3919 }

```

Finally, we end the test for the `minimal` option.

```

3920 }
3921 \</cls>

```

### 31.4 Implementation: OMDoc Package

```

3922 \*package>
3923 \ProvidesExplPackage{omdoc}{2020/10/19}{1.4}{OMDoc document Structure}
3924 \RequirePackage{expl-keystr-compat,13keys2e}

```

### 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).

---

<sup>15</sup>EdNOTE: faking documentkeys for now. @HANG, please implement

```

3925
3926 \keys_define:nn{ document-structure / pkg }{
3927   class      .str_set_x:N = \c_document_structure_class_str,
3928   topsect    .str_set_x:N = \c_document_structure_topsect_str,
3929   % showignores .bool_set:N = \c_document_structure_showignores_bool,
3930 }
3931 \ProcessKeysOptions{ document-structure / pkg }
3932 \str_if_empty:NT \c_document_structure_class_str {
3933   \str_set:Nn \c_document_structure_class_str {article}
3934 }
3935 \str_if_empty:NT \c_document_structure_topsect_str {
3936   \str_set:Nn \c_document_structure_topsect_str {section}
3937 }

```

Then we need to set up the packages by requiring the `sref` package to be loaded.

```

3938 \RequirePackage{xspace}
3939 \RequirePackage{comment}
3940 \@ifpackageloaded{babel}{\RequirePackage[base]{babel}}

```

We set up triggers for the other languages, currently only German.

```

3941 \@ifpackageloaded{babel}{
3942   \clist_set:Nx \l_tmpa_clist {\bb1@loaded}
3943   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
3944     \input{omdoc-ngerman.ldf}
3945   }
3946 }{}
3947 %\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.

```

3948 \int_new:N \l_document_structure_section_level_int
3949 \str_case:VnF \c_document_structure_topsect_str {
3950   {part}{
3951     \int_set:Nn \l_document_structure_section_level_int {0}
3952   }
3953   {chapter}{
3954     \int_set:Nn \l_document_structure_section_level_int {1}
3955   }
3956 }{
3957   \str_case:VnF \c_document_structure_class_str {
3958     {book}{
3959       \int_set:Nn \l_document_structure_section_level_int {0}
3960     }
3961     {report}{
3962       \int_set:Nn \l_document_structure_section_level_int {0}
3963     }
3964   }{
3965     \int_set:Nn \l_document_structure_section_level_int {2}
3966   }
3967 }

```

## 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  $\text{\LaTeX}$  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`, which will be instantiated by CSS later.<sup>16</sup>

EdN:16

```
3968 \def\current@section@level{document}%
3969 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
3970 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
3971 \cs_new_protected:Npn \skipomgroup {
3972   \ifcase\l_document_structure_section_level_int
3973   \or\stepcounter{part}
3974   \or\stepcounter{chapter}
3975   \or\stepcounter{section}
3976   \or\stepcounter{subsection}
3977   \or\stepcounter{subsubsection}
3978   \or\stepcounter{paragraph}
3979   \or\stepcounter{subparagraph}
3980   \fi
3981 }
```

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

`blindomgroup`

```
3982 \newcommand\at@begin@blindomgroup[1]{%
3983 \newenvironment{blindomgroup}
3984 {
3985   \int_incr:N\l_document_structure_section_level_int
3986   \at@begin@blindomgroup\l_document_structure_section_level_int
3987 }{}}
```

`\omgroup@nonum` convenience macro: `\omgroup@nonum{<level>}{<title>}` makes an unnumbered sectioning with title `<title>` at level `<level>`.

```
3988 \newcommand\omgroup@nonum[2]{
3989   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
3990   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
3991 }
```

*(End definition for \omgroup@nonum. This function is documented on page ??.)*

`\omgroup@num` convenience macro: `\omgroup@num{<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 `rdfmata` package has been loaded. In the end we call `\sref@label@id` to enable crossreferencing.

```
3992 \newcommand\omgroup@num[2]{
```

<sup>16</sup>EDNOTE: MK: we may have to experiment with the more powerful uppercasing macro from `mfirstuc.sty` once we internationalize.

```

3993 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
3994   \@nameuse{#1}{#2}
3995 }{
3996   \cs_if_exist:NTF\rdfmata@sectioning{
3997     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
3998   }{
3999     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
4000   }
4001 }
4002 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
4003 }

```

(End definition for \omgroup@num. This function is documented on page ??.)

omgroup

```

4004 \keys_define:nn { document-structure / omgroup }{
4005   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
4006   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
4007   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
4008   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
4009   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
4010   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
4011   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
4012   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
4013   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
4014   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
4015 }
4016 \cs_new_protected:Nn \__document_structure_omgroup_args:n {
4017   \str_clear:N \l__document_structure_omgroup_id_str
4018   \str_clear:N \l__document_structure_omgroup_date_str
4019   \clist_clear:N \l__document_structure_omgroup_creators_clist
4020   \clist_clear:N \l__document_structure_omgroup_contributors_clist
4021   \tl_clear:N \l__document_structure_omgroup_srccite_tl
4022   \tl_clear:N \l__document_structure_omgroup_type_tl
4023   \tl_clear:N \l__document_structure_omgroup_short_tl
4024   \tl_clear:N \l__document_structure_omgroup_display_tl
4025   \tl_clear:N \l__document_structure_omgroup_intro_tl
4026   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
4027   \keys_set:nn { document-structure / omgroup } { #1 }
4028 }

```

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.

```

4029 \newif\if@mainmatter\@mainmattertrue
4030 \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.

```

4031 \keys_define:nn { document-structure / sectioning }{
4032   name .str_set_x:N = \l__document_structure_sect_name_str ,
4033   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
4034   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
4035   num .bool_set:N = \l__document_structure_sect_num_bool ,
4036 }

```

```

4037 \cs_new_protected:Nn \__document_structure_sect_args:n {
4038   \str_clear:N \l__document_structure_sect_name_str
4039   \str_clear:N \l__document_structure_sect_ref_str
4040   \bool_set_false:N \l__document_structure_sect_clear_bool
4041   \bool_set_false:N \l__document_structure_sect_num_bool
4042   \keys_set:nn { document-structure / sectioning } { #1 }
4043 }
4044 \newcommand\omdoc@sectioning[3][]{
4045   \__document_structure_sect_args:n {#1}
4046   \let\omdoc@sect@name\l__document_structure_sect_name_str
4047   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
4048   \if@mainmatter% numbering not overridden by frontmatter, etc.
4049     \bool_if:NTF \l__document_structure_sect_num_bool {
4050       \omgroup@num{#2}{#3}
4051     }{
4052       \omgroup@nonum{#2}{#3}
4053     }
4054     \def\current@section@level{\omdoc@sect@name}
4055   \else
4056     \omgroup@nonum{#2}{#3}
4057   \fi
4058 }% if@mainmatter

```

and another one, if redefines the `\addtocontentsline` macro of L<sup>A</sup>T<sub>E</sub>X to import the respective macros. It takes as an argument a list of module names.

```

4059 \newcommand\omgroup@redefine@addtocontents[1]{%
4060   %\edef\__document_structureimport{#1}%
4061   %\@for\@I:=\__document_structureimport\do{%
4062     %\edef\@path{\csname module@\@I @path\endcsname}%
4063     %\@ifundefined{tf@toc}\relax%
4064     % {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
4065   %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
4066   %\def\addcontentsline##1##2##3{%
4067     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
4068   %\else% hyperref.sty not loaded
4069   %\def\addcontentsline##1##2##3{%
4070     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
4071   %\fi
4072 }% hyperref.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 registers the current level of `omgroups` in the `\omgroup@level` counter.

```

4073 \int_new:N \l_document_structure_omgroup_level_int
4074 \newenvironment{omgroup}[2][]{% keys, title
4075 {
4076   \__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.

```

4077 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
4078   \omgroup@redefine@addtocontents{
4079     %\@ifundefined{module@id}\used@modules%
4080     %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}

```

```

4081     }
4082 }

now we only need to construct the right sectioning depending on the value of \section@level.

4083 \int_incr:N \l_document_structure_omgroup_level_int
4084 \int_incr:N \l_document_structure_section_level_int
4085 \ifcase\l_document_structure_section_level_int
4086   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
4087   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
4088   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
4089   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
4090   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
4091   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
4092   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
4093 \fi
4094 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
4095 \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str
4096 }% for customization
4097 {}

```

and finally, we localize the sections

```

4098 \newcommand\omdoc@part@kw{Part}
4099 \newcommand\omdoc@chapter@kw{Chapter}
4100 \newcommand\omdoc@section@kw{Section}
4101 \newcommand\omdoc@subsection@kw{Subsection}
4102 \newcommand\omdoc@subsubsection@kw{Subsubsection}
4103 \newcommand\omdoc@paragraph@kw{paragraph}
4104 \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`

```

4105 \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 environ-
ments, we save their behavior (possibly defining it) in orig*matter macros and make
them undefined (so that we can define the environments).

4106 \cs_if_exist:NTF\frontmatter{
4107   \let\__document_structure_orig_frontmatter\frontmatter
4108   \let\frontmatter\relax
4109 }{
4110   \tl_set:Nn\__document_structure_orig_frontmatter{
4111     \clearpage
4112     \@mainmatterfalse
4113     \pagenumbering{roman}
4114   }
4115 }
4116 \cs_if_exist:NTF\backmatter{

```



```

4117 \let\__document_structure_orig_backmatter\backmatter
4118 \let\backmatter\relax
4119 }{
4120 \tl_set:Nn\__document_structure_orig_backmatter{
4121 \clearpage
4122 \@mainmatterfalse
4123 \pagenumbering{roman}
4124 }
4125 }

```

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.

```

4126 \newenvironment{frontmatter}{
4127 \__document_structure_orig_frontmatter
4128 }{
4129 \cs_if_exist:NTF\mainmatter{
4130 \mainmatter
4131 }{
4132 \clearpage
4133 \@mainmattertrue
4134 \pagenumbering{arabic}
4135 }
4136 }

```

**backmatter** As `backmatter` is at the end of the document, we do nothing for `\endbackmatter`.

```

4137 \newenvironment{backmatter}{
4138 \__document_structure_orig_backmatter
4139 }{
4140 \cs_if_exist:NTF\mainmatter{
4141 \mainmatter
4142 }{
4143 \clearpage
4144 \@mainmattertrue
4145 \pagenumbering{arabic}
4146 }
4147 }

```

finally, we make sure that page numbering is arabic and we have main matter as the default

```

4148 \@mainmattertrue\pagenumbering{arabic}

```

**\prematurestop** We initialize `\afterprematurestop`, and provide `\prematurestop@endomgroup` which looks up `\omgroup@level` and recursively ends enough `{omgroup}`s.

```

4149 \newcommand\afterprematurestop{}
4150 \def\prematurestop@endomgroup{
4151 \int_compare:nNf \l_document_structure_omgroup_level_int = 0 {
4152 \end{omgroup}
4153 \int_decr:N \l_document_structure_omgroup_level_int
4154 \prematurestop@endomgroup
4155 }
4156 }
4157 \providecommand\prematurestop{

```

```

4158 \message{Stopping sTeX processing prematurely}
4159 \prematurestop@endomgroup
4160 \afterprematurestop
4161 \end{document}
4162 }

```

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

## 31.8 Global Variables

**\setSGvar** set a global variable

```

4163 \RequirePackage{etoolbox}
4164 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

**\useSGvar** use a global variable

```

4165 \newrobustcmd\useSGvar[1]{%
4166 \@ifundefined{sTeX@Gvar@#1}
4167 {\PackageError{omdoc}
4168 {The sTeX Global variable #1 is undefined}
4169 {set it with \protect\setSGvar}}
4170 \@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.

```

4171 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
4172 \@ifundefined{sTeX@Gvar@#1}
4173 {\PackageError{omdoc}
4174 {The sTeX Global variable #1 is undefined}
4175 {set it with \protect\setSGvar}}
4176 {\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.

```
4177 \*cls)
4178 \@@=mikoslides)
4179 \ProvidesExplClass{mikoslides}{2020/12/06}{1.3}{MiKo slides Class}
4180 \RequirePackage{13keys2e,expl-keystr-compat}
4181
4182 \keys_define:nn{mikoslides / cls}{
4183   class .code:n = {
4184     \PassOptionsToClass{\CurrentOption}{omdoc}
4185     \str_if_eq:nnT{#1}{book}{
4186       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
4187     }
4188     \str_if_eq:nnT{#1}{report}{
4189       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
4190     }
4191   },
4192   notes .bool_set:N = \c__mikoslides_notes_bool ,
4193   slides .code:n = { \bool_set_false:N \c__mikoslides_notes_bool },
4194   unknown .code:n = {
4195     \PassOptionsToClass{\CurrentOption}{omdoc}
4196     \PassOptionsToClass{\CurrentOption}{beamer}
4197     \PassOptionsToPackage{\CurrentOption}{mikoslides}
4198   }
4199 }
4200 \ProcessKeysOptions{ mikoslides / cls }
4201 \bool_if:NTF \c__mikoslides_notes_bool {
4202   \PassOptionsToPackage{notes=true}{mikoslides}
4203 }{
4204   \PassOptionsToPackage{notes=false}{mikoslides}
4205 }
4206 \</cls)
```

now we do the same for the mikoslides package.

```

4207 <*package>
4208 \ProvidesExplPackage{mikoslides}{2020/12/06}{1.3}{MiKo slides Package}
4209 \RequirePackage{l3keys2e,expl-keystr-compat}
4210
4211 \keys_define:nn{mikoslides / pkg}{
4212   topsect      .str_set_x:N = \c__mikoslides_topsect_str,
4213   defaulttopsect .str_set_x:N = \c__mikoslides_defaulttopsec_str,
4214   notes        .bool_set:N = \c__mikoslides_notes_bool ,
4215   slides       .code:n      = { \bool_set_false:N \c__mikoslides_notes_bool },
4216   sectocframes .bool_set:N = \c__mikoslides_sectocframes_bool ,
4217   frameimages  .bool_set:N = \c__mikoslides_frameimages_bool ,
4218   fiboxed      .bool_set:N = \c__mikoslides_fiboxed_bool ,
4219   nopproblems  .bool_set:N = \c__mikoslides_nopproblems_bool,
4220   unknown      .code:n      = {
4221     \PassOptionsToClass{\CurrentOption}{stex}
4222     \PassOptionsToClass{\CurrentOption}{tikzinput}
4223   }
4224 }
4225 \ProcessKeysOptions{ mikoslides / pkg }
4226 \newif\ifnotes
4227 \bool_if:NTF \c__mikoslides_notes_bool {
4228   \notesttrue
4229 }{
4230   \notesfalse
4231 }
4232

```

we give ourselves a macro `\@@topsect` that needs only be evaluated once, so that the `\ifdefstring` conditionals work below.

```

4233 \str_if_empty:NTF \c__mikoslides_topsect_str {
4234   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_defaulttopsec_str
4235 }{
4236   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_topsect_str
4237 }
4238 </package>

```

Depending on the options, we either load the article-based omdoc or the beamer class (and set some counters).

```

4239 <*cls>
4240 \bool_if:NTF \c__mikoslides_notes_bool {
4241   \LoadClass{omdoc}
4242 }{
4243   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
4244   \newcounter{Item}
4245   \newcounter{paragraph}
4246   \newcounter{subparagraph}
4247   \newcounter{Hfootnote}
4248   \RequirePackage{omdoc}
4249 }

```

now it only remains to load the mikoslides package that does all the rest.

```

4250 \RequirePackage{mikoslides}
4251 </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.

```

4252 \*package>
4253 \bool_if:NT \c__mikoslides_notes_bool {
4254   \RequirePackage{a4wide}
4255   \RequirePackage{marginnote}
4256   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
4257   \RequirePackage{mdframed}
4258   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
4259   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
4260 }
4261 \RequirePackage{stex-compatibility}
4262 \RequirePackage{stex-tikzinput}
4263 \RequirePackage{etoolbox}
4264 \RequirePackage{amssymb}
4265 \RequirePackage{amsmath}
4266 \RequirePackage{comment}
4267 \RequirePackage{textcomp}
4268 \RequirePackage{url}
4269 \RequirePackage{graphicx}
4270 \RequirePackage{pgf}

```

## 32.2 Notes and Slides

For the lecture notes cases, we also provide the `\usetheme` macro that would otherwise come from the `beamer` class. While the latter loads `beamertheme<theme>.sty`, the notes version loads `beamernotestheme<theme>.sty`.<sup>17</sup>

```

4271 \bool_if:NT \c__mikoslides_notes_bool {
4272   \renewcommand\usetheme[2][\usepackage[#1]{beamernotestheme#2}]
4273 }

```

We define the sizes of slides in the notes. Somehow, we cannot get by with the same here.

```

4274 \newcounter{slide}
4275 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
4276 \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.

```

4277 \bool_if:NTF \c__mikoslides_notes_bool {
4278   \renewenvironment{note}{\ignorespaces}{\ignorespaces}
4279 }{
4280   \excludecomment{note}
4281 }

```

---

<sup>17</sup>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.

```
4282 \bool_if:NT \c__mikoslides_notes_bool {
4283   \newlength{\slideframewidth}
4284   \setlength{\slideframewidth}{1.5pt}
```

**frame** We first define the keys.

```
4285 \cs_new_protected:Nn \__mikoslides_do_yes_param:Nn {
4286   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
4287     \bool_set_true:N #1
4288   }{
4289     \bool_set_false:N #1
4290   }
4291 }
4292 \keys_define:nn{mikoslides / frame}{
4293   label .str_set_x:N = \l__mikoslides_frame_label_str,
4294   allowframebreaks .code:n = {
4295     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowframebreaks_bool { #1 }
4296   },
4297   allowdisplaybreaks .code:n = {
4298     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowdisplaybreaks_bool { #1 }
4299   },
4300   fragile .code:n = {
4301     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_fragile_bool { #1 }
4302   },
4303   shrink .code:n = {
4304     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_shrink_bool { #1 }
4305   },
4306   squeeze .code:n = {
4307     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_squeeze_bool { #1 }
4308   },
4309   t .code:n = {
4310     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_t_bool { #1 }
4311   },
4312 }
4313 \cs_new_protected:Nn \__mikoslides_frame_args:n {
4314   \str_clear:N \l__mikoslides_frame_label_str
4315   \bool_set_true:N \l__mikoslides_frame_allowframebreaks_bool
4316   \bool_set_true:N \l__mikoslides_frame_allowdisplaybreaks_bool
4317   \bool_set_true:N \l__mikoslides_frame_fragile_bool
4318   \bool_set_true:N \l__mikoslides_frame_shrink_bool
4319   \bool_set_true:N \l__mikoslides_frame_squeeze_bool
4320   \bool_set_true:N \l__mikoslides_frame_t_bool
4321   \keys_set:nn { mikoslides / frame }{ #1 }
4322 }
```

We define the environment, read them, and construct the slide number and label.

```
4323 \renewenvironment{frame}[1][]{
4324   \__mikoslides_frame_args:n{#1}
4325   \sffamily
4326   \stepcounter{slide}
4327   \def\@currentlabel{\theslide}
4328   \str_if_empty:NF \l__mikoslides_frame_label_str {
4329     \label{\l__mikoslides_frame_label_str}
```

```
4330 }
```

We redefine the `itemize` environment so that it looks more like the one in `beamer`.

```
4331 \def\itemize@level{outer}
4332 \def\itemize@outer{outer}
4333 \def\itemize@inner{inner}
4334 \renewcommand\newpage{\addtocounter{framenum}{1}}
4335 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
4336 \renewenvironment{itemize}{
4337   \ifx\itemize@level\itemize@outer
4338     \def\itemize@label{$\rhd$}
4339   \fi
4340   \ifx\itemize@level\itemize@inner
4341     \def\itemize@label{$\scriptstyle\rhd$}
4342   \fi
4343   \begin{list}
4344     {\itemize@label}
4345     {\setlength{\labelsep}{.3em}
4346      \setlength{\labelwidth}{.5em}
4347      \setlength{\leftmargin}{1.5em}
4348     }
4349   \edef\itemize@level{\itemize@inner}
4350 }{
4351   \end{list}
4352 }
```

We create the box with the `mdframed` environment from the `equinymous` package.

```
4353 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth]
4354 }{
4355   \medskip\miko@slidelabel\end{mdframed}
4356 }
```

Now, we need to redefine the `frametitle` (we are still in course notes mode).

`\frametitle`

```
4357 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
4358 }
```

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

EdN:18

`\pause` 18

```
4359 \bool_if:NT \c__mikoslides_notes_bool {
4360   \newcommand\pause{}
4361 }
```

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

`nomtext`

```
4362 \bool_if:NTF \c__mikoslides_notes_bool {
4363   \newenvironment{nomtext}[1][\begin{omtext}[#1]}\end{omtext}}{
4364 }{
4365   \excludecomment{nomtext}
4366 }
```

---

<sup>18</sup>EdNOTE: MK: fake it in notes mode for now

nomgroup

```
4367 \bool_if:NTF \c__mikoslides_notes_bool {
4368   \newenvironment{nomgroup}[2] [] {\begin{omgroup}[#1]{#2}}{\end{omgroup}}
4369 }{
4370   \excludecomment{nomgroup}
4371 }
```

ndefinition

```
4372 \bool_if:NTF \c__mikoslides_notes_bool {
4373   \newenvironment{ndefinition}[1] [] {\begin{definition}[#1]}{\end{definition}}
4374 }{
4375   \excludecomment{ndefinition}
4376 }
```

nassertion

```
4377 \bool_if:NTF \c__mikoslides_notes_bool {
4378   \newenvironment{nassertion}[1] [] {\begin{assertion}[#1]}{\end{assertion}}
4379 }{
4380   \excludecomment{nassertion}
4381 }
```

nsproof

```
4382 \bool_if:NTF \c__mikoslides_notes_bool {
4383   \newenvironment{nsproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
4384 }{
4385   \excludecomment{nsproof}
4386 }
```

nexample

```
4387 \bool_if:NTF \c__mikoslides_notes_bool {
4388   \newenvironment{nexample}[1] [] {\begin{example}[#1]}{\end{example}}
4389 }{
4390   \excludecomment{nexample}
4391 }
```

\inputref@\*skip We customize the hooks for in \inputref.

```
4392 \def\inputref@preskip{\smallskip}
4393 \def\inputref@postskip{\medskip}
```

(End definition for \inputref@\*skip. This function is documented on page ??.)

\inputref\*

```
4394 \let\orig@inputref\inputref
4395 \def\inputref{\@ifstar\ninputref\orig@inputref}
4396 \newcommand\ninputref[2] [] {
4397   \bool_if:NT \c__mikoslides_notes_bool {
4398     \orig@inputref[#1]{#2}
4399   }
4400 }
```

(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  $\TeX$  logo. Customization can be done by `\setslidelogo{<logo name>}`.

```

4401 \newlength{\slidelogoheight}
4402
4403 \bool_if:NTF \c__mikoslides_notes_bool {
4404   \setlength{\slidelogoheight}{.4cm}
4405 }{
4406   \setlength{\slidelogoheight}{1cm}
4407 }
4408 \newsavebox{\slidelogo}
4409 \sbox{\slidelogo}{\TeX}
4410 \newrobustcmd{\setslidelogo}[1]{
4411   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
4412 }
```

(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.

```

4413 \def\source{Michael Kohlhase}% customize locally
4414 \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 Attribution-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.

```

4415 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
4416 \newsavebox{\cclogo}
4417 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
4418 \newif\ifcchref\cchreffalse
4419 \AtBeginDocument{
4420   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
4421 }
4422 \def\licensing{
4423   \ifcchref
4424     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
4425   \else
4426     {\usebox{\cclogo}}
4427   \fi
4428 }
4429 \newrobustcmd{\setlicensing}[2][]{
4430   \def@url{#1}
4431   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
4432   \ifx@url@empty
4433     \def\licensing{{\usebox{\cclogo}}}
4434   \else
4435     \def\licensing{
```

```

4436     \ifcchref
4437     \href{#1}{\usebox{\cclogo}}
4438   \else
4439     {\usebox{\cclogo}}
4440   \fi
4441 }
4442 \fi
4443 }

```

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

EdN:19

`\slidelabel` Now, we set up the slide label for the article mode.<sup>19</sup>

```

4444 \newrobustcmd\miko@slidelabel{
4445   \vbox to \slidelogoheight{
4446     \vss\hbox to \slidewidth
4447     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
4448   }
4449 }

```

(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.

```

4450 \def\Gin@mhrepos{}
4451 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
4452 \define@key{Gin}{label}{\def\currentlabel{\arabic{slide}}\label{#1}}
4453 \newrobustcmd\frameimage[2][{}]{
4454   \stepcounter{slide}
4455   \bool_if:NT \c__mikoslides_frameimages_bool {
4456     \def\Gin@ewidth{}\setkeys{Gin}{#1}
4457     \bool_if:NF \c__mikoslides_notes_bool { \vfill }
4458     \begin{center}
4459       \bool_if:NTF \c__mikoslides_fiboxed_bool {
4460         \fbox{
4461           \ifx\Gin@ewidth\@empty
4462             \ifx\Gin@mhrepos\@empty
4463               \mhgraphics[width=\slidewidth,#1]{#2}
4464             \else
4465               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
4466             \fi
4467           \else% \Gin@ewidth empty
4468             \ifx\Gin@mhrepos\@empty
4469               \mhgraphics[#1]{#2}
4470             \else
4471               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
4472             \fi
4473           \fi% \Gin@ewidth empty
4474         }
4475       }{
4476         \ifx\Gin@ewidth\@empty

```

---

<sup>19</sup>EdNOTE: see that we can use the themes for the slides some day. This is all fake.

```

4477         \ifx\Gin@mhrepos\empty
4478             \mhgraphics[width=\slidewidth,#1]{#2}
4479         \else
4480             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
4481         \fi
4482         \ifx\Gin@mhrepos\empty
4483             \mhgraphics[#1]{#2}
4484         \else
4485             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
4486         \fi
4487     \fi% Gin@ewidth empty
4488 }
4489 \end{center}
4490 \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
4491 \bool_if:NF \c__mikoslides_notes_bool { \vfill }
4492 }
4493 } % 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.

```

4494 \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 do is to adapt the green so that it is dark enough for most beamers

```

4495 \AddToHook{begindocument}{
4496     \definecolor{green}{rgb}{0,.5,0}
4497     \definecolor{purple}{cmyk}{.3,1,0,.17}
4498 }

```

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`.

```

4499 % \def\STpresent#1{\textcolor{blue}{#1}}
4500 \def\defemph#1{\textcolor{magenta}{#1}}
4501 \def\symrefemph#1{\textcolor{cyan}{#1}}
4502 \def\compemph#1{\textcolor{blue}{#1}}
4503 \def\titleemph#1{\textcolor{blue}{#1}}
4504 \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.

```

4505 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
4506 \def\smalltextwarning{
4507     \pgfuseimage{miko@small@dbend}
4508     \xspace
4509 }
4510 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}

```

```

4511 \newrobustcmd\textwarning{
4512   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
4513   \xspace
4514 }
4515 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}
4516 \newrobustcmd\bigtextwarning{
4517   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
4518   \xspace
4519 }

```

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

```

4520 \newrobustcmd\putgraphicsat[3]{
4521   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
4522 }
4523 \newrobustcmd\putat[2]{
4524   \begin{picture}(0,0)\put(#1){#2}\end{picture}
4525 }

```

## 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`.

```

4526 \bool_if:NT \c__mikoslides_sectocframes_bool {
4527   \str_if_eq:VnTF \__mikoslidestopsect{part}{
4528     \newcounter{chapter}\counterwithin*{section}{chapter}
4529   }{
4530     \str_if_eq:VnT\__mikoslidestopsect{chapter}{
4531       \newcounter{chapter}\counterwithin*{section}{chapter}
4532     }
4533   }
4534 }

```

`\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
4535 \def\part@prefix{}
4536 \@ifpackageloaded{omdoc}{}{
4537   \str_case:VnF \__mikoslidestopsect {
4538     {part}{
4539       \int_set:Nn \l_document_structure_section_level_int {0}
4540       \def\thesection{\arabic{chapter}.\arabic{section}}
4541       \def\part@prefix{\arabic{chapter}.}
4542     }
4543     {chapter}{
4544       \int_set:Nn \l_document_structure_section_level_int {1}
4545       \def\thesection{\arabic{chapter}.\arabic{section}}
4546       \def\part@prefix{\arabic{chapter}.}
4547     }
4548   }{
4549     \int_set:Nn \l_document_structure_section_level_int {2}
4550     \def\part@prefix{}

```

```

4551 }
4552 }
4553
4554 \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 L<sup>A</sup>T<sub>E</sub>X sectioning macros according to \section@level.

omgroup

```

4555 \renewenvironment{omgroup}[2][]{
4556   \__document_structure_omgroup_args:n { #1 }
4557   \int_incr:N \l_document_structure_omgroup_level_int
4558   \int_incr:N \l_document_structure_section_level_int
4559   \bool_if:NT \c__mikoslides_sectocframes_bool {
4560     \stepcounter{slide}
4561     \begin{frame}[noframenumbering]
4562     \vfill\Large\centering
4563     \red{
4564       \ifcase\l_document_structure_section_level_int\or
4565         \stepcounter{part}
4566         \def\__mikoslideslabel{\omdoc@part@kw~\Roman{part}}
4567         \def\currentsectionlevel{\omdoc@part@kw}
4568       \or
4569         \stepcounter{chapter}
4570         \def\__mikoslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
4571         \def\currentsectionlevel{\omdoc@chapter@kw}
4572       \or
4573         \stepcounter{section}
4574         \def\__mikoslideslabel{\part@prefix\arabic{section}}
4575         \def\currentsectionlevel{\omdoc@section@kw}
4576       \or
4577         \stepcounter{subsection}
4578         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
4579         \def\currentsectionlevel{\omdoc@subsection@kw}
4580       \or
4581         \stepcounter{subsubsection}
4582         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
4583         \def\currentsectionlevel{\omdoc@subsubsection@kw}
4584       \or
4585         \stepcounter{mparagraph}
4586         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{msubsection}.\arabic{mparagraph}}
4587         \def\currentsectionlevel{\omdoc@paragraph@kw}
4588       \fi% end ifcase
4589       \__mikoslideslabel%\sref@label@id\__mikoslideslabel
4590       \quad #2%
4591     }%
4592     \vfill%
4593     \end{frame}%
4594   }
4595   \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str%
4596 }{}
4597 }

```

We set up a `beamer` template for theorems like `ams` style, but without a block environment.

```

4598 \def\inserttheorembodyfont{\normalfont}
4599 \bool_if:NF \c__mikoslides_notes_bool {
4600   \defbeamertemplate{theorem begin}{miko}
4601   {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
4602     \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
4603     \inserttheorempunctuation\inserttheorembodyfont\space}
4604   \defbeamertemplate{theorem end}{miko}{}
```

and we set it as the default one.

```

4605   \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.

```

4606   \expandafter\def\csname Parent2\endcsname{}
4607 }
4608 \bool_if:NT \c__mikoslides_notes_bool {
4609   \renewenvironment{columns}[1][{}]{%
4610     \par\noindent%
4611     \begin{minipage}%
4612       \slidewidth\centering\leavevmode%
4613   }{%
4614     \end{minipage}\par\noindent%
4615   }%
4616   \newsavebox\columnbox%
4617   \renewenvironment<>{column}[2][{}]{%
4618     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
4619   }{%
4620     \end{minipage}\end{lrbox}\usebox\columnbox%
4621   }%
4622 }
4623 \bool_if:NTF \c__mikoslides_noproblems_bool {
4624   \newenvironment{problems}{}{}
4625 }{
4626   \excludecomment{problems}
4627 }
```

## 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.

```

4628 \gdef\printexcursions{}
4629 \newcommand\excursionref[2]{% label, text
4630   \bool_if:NT \c__mikoslides_notes_bool {
4631     \begin{omtext}[title=Excursion]
4632       #2 \sref[fallback=the appendix]{#1}.
4633     \end{omtext}
4634   }
4635 }
4636 \newcommand\activate@excursion[2][{}]{
4637   \gappto\printexcursions{\inputref[#1]{#2}}
```

```

4638 }
4639 \newcommand\excursion[4][{}]{% repos, label, path, text
4640   \bool_if:NT \c__mikoslides_notes_bool {
4641     \activate@excursion[#1]{#3}\excursionref{#2}{#4}
4642   }
4643 }

```

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

## \excursiongroup

```

4644 \keys_define:nn{mikoslides / excursiongroup }{
4645   id          .str_set_x:N = \l__mikoslides_excursion_id_str,
4646   intro       .tl_set:N   = \l__mikoslides_excursion_intro_tl,
4647   mhrepos     .str_set_x:N = \l__mikoslides_excursion_mhrepos_str
4648 }
4649 \cs_new_protected:Nn \__mikoslides_excursion_args:n {
4650   \tl_clear:N \l__mikoslides_excursion_intro_tl
4651   \str_clear:N \l__mikoslides_excursion_id_str
4652   \str_clear:N \l__mikoslides_excursion_mhrepos_str
4653   \keys_set:nn {mikoslides / excursiongroup }{ #1 }
4654 }
4655 \newcommand\excursiongroup[1][{}]{
4656   \__mikoslides_excursion_args:n{ #1 }
4657   \ifdefempty\printexcursions{}% only if there are excursions
4658   {\begin{note}
4659     \begin{omgroup}[#1]{Excursions}%
4660     \ifdefempty\l__mikoslides_excursion_intro_tl{{
4661       \inputref[\l__mikoslides_excursion_mhrepos_str]{
4662         \l__mikoslides_excursion_intro_tl
4663       }
4664     }
4665     \printexcursions%
4666     \end{omgroup}
4667   \end{note}}
4668 }
4669 \end{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.

```
4670 <*package>
4671 <@@=problems>
4672 \ProvidesExplPackage{problem}{2019/03/20}{1.3}{Semantic Markup for Problems}
4673 \RequirePackage{l3keys2e,expl-keystr-compat}
4674
4675 \keys_define:nn { problem / pkg }{
4676   notes      .default:n    = { true },
4677   notes      .bool_set:N   = \c__problems_notes_bool,
4678   gnotes     .default:n    = { true },
4679   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
4680   hints      .default:n    = { true },
4681   hints      .bool_set:N   = \c__problems_hints_bool,
4682   solutions  .default:n    = { true },
4683   solutions  .bool_set:N   = \c__problems_solutions_bool,
4684   pts        .default:n    = { true },
4685   pts        .bool_set:N   = \c__problems_pts_bool,
4686   min        .default:n    = { true },
4687   min        .bool_set:N   = \c__problems_min_bool,
4688   boxed      .default:n    = { true },
4689   boxed      .bool_set:N   = \c__problems_boxed_bool,
4690   unknown    .code:n       = {}
4691 }
4692 \def\solutionstrue{
4693   \bool_set_true:N \c__problems_solutions_bool
4694 }
4695 \def\solutionsfalse{
4696   \bool_set_false:N \c__problems_solutions_bool
4697 }
4698
4699 \ProcessKeysOptions{ problem / pkg }
```

Then we make sure that the necessary packages are loaded (in the right versions).



```

4700 \RequirePackage{stex-compatibility}
4701 \RequirePackage{comment}

```

The next package relies on the L<sup>A</sup>T<sub>E</sub>X3 kernel, which L<sup>A</sup>T<sub>E</sub>XML only partially supports. As it is purely presentational, we only load it when the `boxed` option is given and we run L<sup>A</sup>T<sub>E</sub>XML.

```

4702 \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.

```

4703 \def\prob@problem@kw{Problem}
4704 \def\prob@solution@kw{Solution}
4705 \def\prob@hint@kw{Hint}
4706 \def\prob@note@kw{Note}
4707 \def\prob@gnote@kw{Grading}
4708 \def\prob@pt@kw{pt}
4709 \def\prob@min@kw{min}

```

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

For the other languages, we set up triggers

```

4710 \@ifpackageloaded{babel}{
4711   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4712   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4713     \input{problem-ngerman.ldf}
4714   }
4715   \clist_if_in:NnT \l_tmpa_clist {finnish}{
4716     \input{problem-finnish.ldf}
4717   }
4718   \clist_if_in:NnT \l_tmpa_clist {french}{
4719     \input{problem-french.ldf}
4720   }
4721   \clist_if_in:NnT \l_tmpa_clist {russian}{
4722     \input{problem-russian.ldf}
4723   }
4724 }{}

```

## 33.2 Problems and Solutions

We now prepare the KeyVal support for problems. The key macros just set appropriate internal macros.

```

4725 \keys_define:nn{ problem / problem }{
4726   id      .str_set:x:N = \l__problems_prob_id_str,
4727   pts     .tl_set:N    = \l__problems_prob_pts_tl,
4728   min     .tl_set:N    = \l__problems_prob_min_tl,
4729   title   .tl_set:N    = \l__problems_prob_title_tl,
4730   refnum  .int_set:N   = \l__problems_prob_refnum_int
4731 }
4732 \cs_new_protected:Nn \__problems_prob_args:n {
4733   \str_clear:N \l__problems_prob_id_str
4734   \tl_clear:N \l__problems_prob_pts_tl
4735   \tl_clear:N \l__problems_prob_min_tl
4736   \tl_clear:N \l__problems_prob_title_tl

```

```

4737 \int_zero_new:N \l__problems_prob_refnum_int
4738 \keys_set:nn { problem / problem }{ #1 }
4739 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
4740   \let\l__problems_inclprob_refnum_int\undefined
4741 }
4742 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

4743 \newcounter{problem}
4744 \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.

```

4745 \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

```

4746 \newcommand\prob@number{
4747   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
4748     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
4749   }{
4750     \int_if_exist:NTF \l__problems_prob_refnum_int {
4751       \prob@label{\int_use:N \l__problems_prob_refnum_int }
4752     }{
4753       \prob@label\theproblem
4754     }
4755   }
4756 }

```

(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.

```

4757 \newcommand\prob@title[3]{%
4758   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
4759     #2 \l__problems_inclprob_title_tl #3
4760   }{
4761     \tl_if_exist:NTF \l__problems_prob_title_tl {
4762       #2 \l__problems_prob_title_tl #3
4763     }{
4764       #1
4765     }
4766   }
4767 }

```

(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.

```

4768 \def\prob@heading{
4769   \prob@problem@kw~\prob@number\prob@title{~}{~}{~}\strut}
4770   \%sref@label@id{\prob@problem@kw~\prob@number}{~}
4771 }

```

(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`

```

4772 \newenvironment{problem}[1][1]{
4773   \_problems_prob_args:n{#1}%\sref@target%
4774   \@in@omtexttrue% we are in a statement (for inline definitions)
4775   \stepcounter{problem}\record@problem
4776   \def\current@section@level{\prob@problem@kw}
4777   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
4778 }%
4779   {\smallskip}
4780   \bool_if:NT \c__problems_boxed_bool {
4781     \surroundwithmdframed{problem}
4782   }

```

`\record@problem` This macro records information about the problems in the `*.aux` file.

```

4783 \def\record@problem{
4784   \protected@write\@auxout{}
4785   {
4786     \string\@problem{\prob@number}
4787     {
4788       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
4789         \l__problems_inclprob_pts_tl
4790       }{
4791         \l__problems_prob_pts_tl
4792       }
4793     }%
4794     {
4795       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
4796         \l__problems_inclprob_min_tl
4797       }{
4798         \l__problems_prob_min_tl
4799       }
4800     }
4801   }
4802 }

```

(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).

```

4803 \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.

```

4804 \keys_define:nn { problem / solution }{
4805   id          .str_set_x:N = \l__problems_solution_id_str ,
4806   for         .tl_set:N    = \l__problems_solution_for_tl ,
4807   height      .dim_set:N   = \l__problems_solution_height_dim ,
4808   creators    .clist_set:N = \l__problems_solution_creators_clist ,
4809   contributors .clist_set:N = \l__problems_solution_contributors_clist ,
4810   srccite     .tl_set:N    = \l__problems_solution_srccite_tl
4811 }
4812 \cs_new_protected:Nn \__problems_solution_args:n {
4813   \str_clear:N \l__problems_solution_id_str
4814   \tl_clear:N \l__problems_solution_for_tl
4815   \tl_clear:N \l__problems_solution_srccite_tl
4816   \clist_clear:N \l__problems_solution_creators_clist
4817   \clist_clear:N \l__problems_solution_contributors_clist
4818   \dim_zero:N \l__problems_solution_height_dim
4819   \keys_set:nn { problem / solution }{ #1 }
4820 }

```

the next step is to define a helper macro that does what is needed to start a solution.

```

4821 \newcommand\@startsolution[1][ ]{
4822   \__problems_solution_args:n { #1 }
4823   \@in@omtexttrue% we are in a statement.
4824   \bool_if:NF \c__problems_boxed_bool { \hrule }
4825   \smallskip\noindent
4826   {\textbf\prob@solution@kw : \enspace}
4827   \begin{small}
4828   \def\current@section@level{\prob@solution@kw}
4829   \ignorespacesandpars
4830 }

```

**\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.

```

4831 \newcommand\startsolutions{
4832   \specialcomment{solution}{\@startsolution}{
4833     \bool_if:NF \c__problems_boxed_bool {
4834       \hrule\medskip
4835     }
4836     \end{small}%
4837   }
4838   \bool_if:NT \c__problems_boxed_bool {
4839     \surroundwithmdframed{solution}
4840   }
4841 }

```

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

**\stopsolutions**

```

4842 \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.

```

4843 \bool_if:NTF \c_problems_solutions_bool {
4844   \startsolutions
4845 }{
4846   \stopsolutions
4847 }

```

**exnote**

```

4858 \bool_if:NTF \c_problems_notes_bool {
4859   \newenvironment{exnote}[1][]{
4860     \par\smallskip\hrule\smallskip
4861     \noindent\textbf{\prob@note@kw : }\small
4862   }{
4863     \smallskip\hrule
4864   }
4865 }{
4866   \excludecomment{exnote}
4867 }

```

**hint**

```

4858 \bool_if:NTF \c_problems_notes_bool {
4859   \newenvironment{hint}[1][]{
4860     \par\smallskip\hrule\smallskip
4861     \noindent\textbf{\prob@hint@kw :~ }\small
4862   }{
4863     \smallskip\hrule
4864   }
4865   \newenvironment{exhint}[1][]{
4866     \par\smallskip\hrule\smallskip
4867     \noindent\textbf{\prob@hint@kw :~ }\small
4868   }{
4869     \smallskip\hrule
4870   }
4871 }{
4872   \excludecomment{hint}
4873   \excludecomment{exhint}
4874 }

```

**gnote**

```

4875 \bool_if:NTF \c_problems_notes_bool {
4876   \newenvironment{gnote}[1][]{
4877     \par\smallskip\hrule\smallskip
4878     \noindent\textbf{\prob@gnote@kw : }\small
4879   }{
4880     \smallskip\hrule
4881   }
4882 }{
4883   \excludecomment{gnote}
4884 }

```

### 33.3 Multiple Choice Blocks

```

4885 \newenvironment{mcb}{
4886   \begin{enumerate}
4887 }{
4888   \end{enumerate}
4889 }

```

we define the keys for the mcc macro

```

4890 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
4891   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
4892     \bool_set_true:N #1
4893   }{
4894     \bool_set_false:N #1
4895   }
4896 }
4897 \keys_define:nn { problem / mcc }{
4898   id          .str_set:x:N = \l__problems_mcc_id_str ,
4899   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
4900   T           .default:n   = { true } ,
4901   T           .bool_set:N   = \l__problems_mcc_t_bool ,
4902   F           .default:n   = { true } ,
4903   F           .bool_set:N   = \l__problems_mcc_f_bool ,
4904   Ttext       .code:n       = {
4905     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
4906   } ,
4907   Ftext       .code:n       = {
4908     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
4909   }
4910 }
4911 \cs_new_protected:Nn \l__problems_mcc_args:n {
4912   \str_clear:N \l__problems_mcc_id_str
4913   \tl_clear:N \l__problems_mcc_feedback_tl
4914   \bool_set_true:N \l__problems_mcc_t_bool
4915   \bool_set_true:N \l__problems_mcc_f_bool
4916   \bool_set_true:N \l__problems_mcc_Ttext_bool
4917   \bool_set_false:N \l__problems_mcc_Ftext_bool
4918   \keys_set:nn { problem / mcc }{ #1 }
4919 }

```

\mcc

```

4920 \newcommand\mcc[2][]{
4921   \l__problems_mcc_args:n{ #1 }
4922   \item #2
4923   \bool_if:NT \c__problems_solutions_bool {
4924     \
4925     \bool_if:NT \l__problems_mcc_t_bool {
4926       % TODO!
4927       % \ifcsstring{mcc@T}{T}{\mcc@Ttext}%
4928     }
4929     \bool_if:NT \l__problems_mcc_f_bool {

```

---

<sup>20</sup>EdNOTE: MK: maybe import something better here from a dedicated MC package

```

4930      % TODO!
4931      % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
4932    }
4933    \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
4934      !
4935    }{
4936      \l__problems_mcc_feedback_tl
4937    }
4938  }
4939 } %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.

```

4940
4941 \keys_define:nn{ problem / inclproblem }{
4942   % id      .str_set_x:N = \l__problems_inclprob_id_str,
4943   pts      .tl_set:N    = \l__problems_inclprob_pts_tl,
4944   min      .tl_set:N    = \l__problems_inclprob_min_tl,
4945   title    .tl_set:N    = \l__problems_inclprob_title_tl,
4946   refnum   .int_set:N    = \l__problems_inclprob_refnum_int,
4947   mhrepos  .str_set_x:N = \l__problems_inclprob_mhrepos_str
4948 }
4949 \cs_new_protected:Nn \l__problems_inclprob_args:n {
4950   % \str_clear:N \l__problems_prob_id_str
4951   \tl_clear:N \l__problems_inclprob_pts_tl
4952   \tl_clear:N \l__problems_inclprob_min_tl
4953   \tl_clear:N \l__problems_inclprob_title_tl
4954   \int_zero_new:N \l__problems_inclprob_refnum_int
4955   \str_clear:N \l__problems_inclprob_mhrepos_str
4956   \keys_set:nn { problem / inclproblem }{ #1 }
4957   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
4958     \let\l__problems_inclprob_pts_tl\undefined
4959   }
4960   \tl_if_empty:NT \l__problems_inclprob_min_tl {
4961     \let\l__problems_inclprob_min_tl\undefined
4962   }
4963   \tl_if_empty:NT \l__problems_inclprob_title_tl {
4964     \let\l__problems_inclprob_title_tl\undefined
4965   }
4966   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
4967     \let\l__problems_inclprob_refnum_int\undefined
4968   }
4969 }
4970
4971 \cs_new_protected:Nn \l__problems_inclprob_clear: {
4972   % \str_clear:N \l__problems_prob_id_str
4973   \let\l__problems_inclprob_pts_tl\undefined
4974   \let\l__problems_inclprob_min_tl\undefined

```

```

4975 \let\l__problems_inclprob_title_tl\undefined
4976 \let\l__problems_inclprob_refnum_int\undefined
4977 \let\l__problems_inclprob_mhrepos_str\undefined
4978 }
4979
4980 \newcommand\includeproblem[2][ ]{
4981   \__problems_inclprob_args:n{ #1 }
4982   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
4983     \input{#2}
4984   }{
4985     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
4986       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
4987     }
4988   }
4989   \__problems_inclprob_clear:
4990 }

```

(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.

```

4991 \AddToHook{enddocument}{
4992   \bool_if:NT \c__problems_pts_bool {
4993     \message{Total:~\arabic{pts}~points}
4994   }
4995   \bool_if:NT \c__problems_min_bool {
4996     \message{Total:~\arabic{min}~minutes}
4997   }
4998 }

```

The margin pars are reader-visible, so we need to translate

```

4999 \def\pts#1{
5000   \bool_if:NT \c__problems_pts_bool {
5001     \marginpar{#1~\prob@pt@kw}
5002   }
5003 }
5004 \def\min#1{
5005   \bool_if:NT \c__problems_min_bool {
5006     \marginpar{#1~\prob@min@kw}
5007   }
5008 }

```

**\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.

```

5009 \newcounter{pts}
5010 \def\show@pts{
5011   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
5012     \bool_if:NT \c__problems_pts_bool {
5013       \marginpar{\l__problems_inclprob_pts_tl;\prob@pt@kw\smallskip}
5014       \addtocounter{pts}{\l__problems_inclprob_pts_tl}

```



```

5015     }
5016   }{
5017     \tl_if_exist:NT \l__problems_prob_pts_tl {
5018       \bool_if:NT \c__problems_pts_bool {
5019         \marginpar{\l__problems_prob_pts_tl;\prob@pt@kw\smallskip}
5020         \addtocounter{pts}{\l__problems_prob_pts_tl}
5021       }
5022     }
5023   }
5024 }

```

(End definition for \show@pts. This function is documented on page ??.)  
and now the same for the minutes

\show@min

```

5025 \newcounter{min}
5026 \def\show@min{
5027   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
5028     \bool_if:NT \c__problems_min_bool {
5029       \marginpar{\l__problems_inclprob_pts_tl;min}
5030       \addtocounter{min}{\l__problems_inclprob_min_tl}
5031     }
5032   }{
5033     \tl_if_exist:NT \l__problems_prob_min_tl {
5034       \bool_if:NT \c__problems_min_bool {
5035         \marginpar{\l__problems_prob_min_tl;min}
5036         \addtocounter{min}{\l__problems_prob_min_tl}
5037       }
5038     }
5039   }
5040 }
5041 \</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.

```
5042 <@@=hwexam>
5043 <*cls>
5044 \ProvidesExplClass{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
5045 \RequirePackage{l3keys2e,expl-keystr-compatible}
5046 \DeclareOption*{
5047   \PassOptionsToClass{\CurrentOption}{omdoc}
5048   \PassOptionsToPackage{\CurrentOption}{stex}
5049   \PassOptionsToPackage{\CurrentOption}{hwexam}
5050   \PassOptionsToPackage{\CurrentOption}{tikzinput}
5051 }
5052 \ProcessOptions
```

We load `omdoc.cls`, and the desired packages. For the L<sup>A</sup>T<sub>E</sub>XML bindings, we make sure the right packages are loaded.

```
5053 \LoadClass{omdoc}
5054 \RequirePackage{stex}
5055 \RequirePackage{hwexam}
5056 \RequirePackage{tikzinput}
5057 \RequirePackage{graphicx}
5058 \RequirePackage{a4wide}
5059 \RequirePackage{amssymb}
5060 \RequirePackage{amstext}
5061 \RequirePackage{amsmath}
```

Finally, we register another keyword for the `document` environment. We give a default assignment type to prevent errors

```

5062 \newcommand\assig@default@type{\hwexam@assignment@kw}
5063 \def\document@hwexamtype{\assig@default@type}
5064 <@@=document_structure>
5065 \keys_define:nn { document-structure / document }{
5066 id .str_set_x:N = \c_document_structure_document_id_str,
5067 hwexamtype .tl_set:N = \document@hwexamtype
5068 }
5069 <@@=hwexam>
5070 </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.

```
5071 \langle *package\rangle
5072 \ProvidesExplPackage{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
5073 \RequirePackage{l3keys2e,expl-keystr-compat}
5074
5075 \newif\iftest\testfalse
5076 \DeclareOption{test}{\testtrue}
5077 \newif\ifmultiple\multiplefalse
5078 \DeclareOption{multiple}{\multipletrue}
5079 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
5080 \ProcessOptions
```

Then we make sure that the necessary packages are loaded (in the right versions).

```
5081 \RequirePackage{keyval}[1997/11/10]
5082 \RequirePackage{problem}
```

`\hwexam@*@kw` For multilinguality, we define internal macros for keywords that can be specialized in `*.ldf` files.

```
5083 \newcommand\hwexam@assignment@kw{Assignment}
5084 \newcommand\hwexam@given@kw{Given}
5085 \newcommand\hwexam@due@kw{Due}
5086 \newcommand\hwexam@testemptypage@kw{This page was intentionally left blank for extra
5087   space}%
5088 \newcommand\correction@probs@kw{prob.}%
5089 \newcommand\correction@pts@kw{total}%
5090 \newcommand\correction@reached@kw{reached}%
5091 \newcommand\correction@sum@kw{Sum}%
5092 \newcommand\correction@grade@kw{grade}%
5093 \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

```

5094 \ifpackageloaded{babel}{\RequirePackage[base]{babel}}
5095
5096 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5097 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5098   \input{hwexam-ngerman.ldf}
5099 }
5100 \clist_if_in:NnT \l_tmpa_clist {finnish}{
5101   \input{hwexam-finnish.ldf}
5102 }
5103 \clist_if_in:NnT \l_tmpa_clist {french}{
5104   \input{hwexam-french.ldf}
5105 }
5106 \clist_if_in:NnT \l_tmpa_clist {russian}{
5107   \input{hwexam-russian.ldf}
5108 }

```

## 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.

```

5109 \newcounter{assignment}
5110 \numberproblemsin{assignment}
5111 \renewcommand\prob@label[1]{\arabic{assignment}.#1}

```

We will prepare the keyval support for the `assignment` environment.

```

5112 \keys_define:nn { hwexam / assignment } {
5113   id .str_set:N = \l__hwexam_assign_id_str,
5114   number .int_set:N = \l__hwexam_assign_number_int,
5115   title .tl_set:N = \l__hwexam_assign_title_tl,
5116   type .tl_set:N = \l__hwexam_assign_type_tl,
5117   given .tl_set:N = \l__hwexam_assign_given_tl,
5118   due .tl_set:N = \l__hwexam_assign_due_tl,
5119   loadmodules .code:n = {
5120     \bool_set_true:N \l__hwexam_assign_loadmodules_bool
5121   }
5122 }
5123 \cs_new_protected:Nn \__hwexam_assignment_args:n {
5124   \str_clear:N \l__hwexam_assign_id_str
5125   \int_set:Nn \l__hwexam_assign_number_int {-1}
5126   \tl_clear:N \l__hwexam_assign_title_tl
5127   \tl_clear:N \l__hwexam_assign_type_tl
5128   \tl_clear:N \l__hwexam_assign_given_tl
5129   \tl_clear:N \l__hwexam_assign_due_tl
5130   \bool_set_false:N \l__hwexam_assign_loadmodules_bool
5131   \keys_set:nn { hwexam / assignment }{ #1 }
5132 }

```

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.

```

5133 \newcommand\given@due[2]{
5134 \bool_lazy_all:nF {
5135 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
5136 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
5137 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
5138 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
5139 }{ #1 }
5140
5141 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
5142 \tl_if_empty:NF \l__hwexam_assign_given_tl {
5143 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
5144 }
5145 }{
5146 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
5147 }
5148
5149 \bool_lazy_or:nnF {
5150 \bool_lazy_and_p:nn {
5151 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
5152 }{
5153 \tl_if_empty_p:V \l__hwexam_assign_due_tl
5154 }
5155 }{
5156 \bool_lazy_and_p:nn {
5157 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
5158 }{
5159 \tl_if_empty_p:V \l__hwexam_assign_due_tl
5160 }
5161 }{ ,~ }
5162
5163 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
5164 \tl_if_empty:NF \l__hwexam_assign_due_tl {
5165 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
5166 }
5167 }{
5168 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
5169 }
5170
5171 \bool_lazy_all:nF {
5172 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
5173 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
5174 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
5175 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
5176 }{ #2 }
5177 }

```

`\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.

```

5178 \newcommand\assignment@title[3]{

```

```

5179 \tl_if_empty:NTF \l__hwexam_inclassassign_title_tl {
5180 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
5181 #1
5182 }{
5183 #2\l__hwexam_assign_title_tl#3
5184 }
5185 }{
5186 #2\l__hwexam_inclassassign_title_tl#3
5187 }
5188 }

```

(End definition for \assignment@title. This function is documented on page ??.)

**\assignment@number** Like \assignment@title only for the number, and no around part.

```

5189 \newcommand\assignment@number{
5190 \int_compare:nNnTF \l__hwexam_inclassassign_number_int = {-1} {
5191 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
5192 \int_use:N \l__hwexam_assign_number_int
5193 }
5194 }{
5195 \int_use:N \l__hwexam_inclassassign_number_int
5196 }
5197 }

```

(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.

```

5198 \newenvironment{assignment}[1][ ]{
5199 \__hwexam_assignment_args:n { #1 }
5200 %\sref@target
5201 \let\__hwexamnum\l__hwexam_assign_number_int
5202 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
5203 \stepcounter{assignment}
5204 }{
5205 \setcounter{assignment}{\int_use:N\__hwexamnum}
5206 }
5207 \setcounter{problem}{0}
5208 \def\current@section@level{\document@hwexamtype}
5209 %\sref@label@id{\document@hwexamtype \thesection}
5210 \begin{@assignment}
5211 }{
5212 \end{@assignment}
5213 }

```

In the multi-assignment case we just use the omdoc environment for suitable sectioning.

```

5214 \def\__hwexasstitle{
5215 \protect\document@hwexamtype~\arabic{assignment}
5216 \assignment@title{}\;{}{}\; -- \given@due{}\}
5217 }

```

```

5218 \ifmultiple
5219 \newenvironment{@assignment}{
5220 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
5221 \begin{omgroup}[loadmodules]{\__hwexasstitle}
5222 }{
5223 \begin{omgroup}{\__hwexasstitle}
5224 }
5225 }{
5226 \end{omgroup}
5227 }

```

for the single-page case we make a title block from the same components.

```

5228 \else
5229 \newenvironment{@assignment}{
5230 \begin{center}\bf
5231 \Large\@title\strut\
5232 \document@hwexamtype~\arabic{assignment}\assignment@title{\;}{:\;}{\}
5233 \large\given@due{--\;}{\;}{--}
5234 \end{center}
5235 }{}
5236 \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.

```

5237 \keys_define:nn { hwexam / inclassignment } {
5238 %id .str_set_x:N = \l__hwexam_assign_id_str,
5239 number .int_set:N = \l__hwexam_inclassign_number_int,
5240 title .tl_set:N = \l__hwexam_inclassign_title_tl,
5241 type .tl_set:N = \l__hwexam_inclassign_type_tl,
5242 given .tl_set:N = \l__hwexam_inclassign_given_tl,
5243 due .tl_set:N = \l__hwexam_inclassign_due_tl,
5244 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
5245 }
5246 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
5247 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
5248 \tl_clear:N \l__hwexam_inclassign_title_tl
5249 \tl_clear:N \l__hwexam_inclassign_type_tl
5250 \tl_clear:N \l__hwexam_inclassign_given_tl
5251 \tl_clear:N \l__hwexam_inclassign_due_tl
5252 \str_clear:N \l__hwexam_inclassign_mhrepos_str
5253 \keys_set:nn { hwexam / inclassignment }{ #1 }
5254 }
5255 \__hwexam_inclassignment_args:n {}
5256
5257 \newcommand\inputassignment[2][ ]{
5258 \__hwexam_inclassignment_args:n { #1 }
5259 \str_if_empty:NTF \l__hwexam_inclassign_mhrepos_str {
5260 \input{#2}
5261 }{
5262 \stex_in_repository:nn{\l__hwexam_inclassign_mhrepos_str}{

```



```

5263 \input{\mhp\path{\l__hwexam_inclasssign_mhrepos_str}{#2}}
5264 }
5265 }
5266 \__hwexam_inclasssignment_args:n {}
5267 }
5268 \newcommand\includeassignment[2][ ]{
5269 \newpage
5270 \inputassignment[#1]{#2}
5271 }

```

(End definition for \in\*assignment. This function is documented on page ??.)

## 35.4 Typesetting Exams

\quizheading

```

5272 \ExplSyntaxOff
5273 \newcommand\quizheading[1]{%
5274 \def\@tas{#1}%
5275 \large\noindent NAME: \hspace{8cm} MAILBOX:\|[2ex]%
5276 \ifx\@tas\empty\else%
5277 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\|[2ex]%
5278 \fi%
5279 }
5280 \ExplSyntaxOn

```

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

\testheading

```

5281 \keys_define:nn { hwexam / testheading } {
5282 min .tl_set:N = \l__hwexam_testheading_min_tl,
5283 duration .tl_set:N = \l__hwexam_testheading_duration_tl,
5284 reqpts .tl_set:N = \l__hwexam_testheading_reqpts_tl
5285 }
5286 \cs_new_protected:Nn \__hwexam_testheading_args:n {
5287 \tl_clear:N \l__hwexam_testheading_min_tl
5288 \tl_clear:N \l__hwexam_testheading_duration_tl
5289 \tl_clear:N \l__hwexam_testheading_reqpts_tl
5290 \keys_set:nn { hwexam / testheading }{ #1 }
5291 }
5292 \newenvironment{testheading}[1][ ]{
5293 \__hwexam_testheading_args:n{ #1 }
5294 \noindent\large{Name:~\hfill
5295 Matriculation Number:\hspace*{2cm}\strut}\|[1ex]
5296 \begin{center}
5297 \Large\textbf{\@title}\|[1ex]
5298 \large\@date\|[3ex]
5299 \end{center}
5300 \textbf{You~have~
5301 \tl_if_empty:NTF \l__hwexam_testheading_duration_tl {
5302 \l__hwexam_testheading_min_tl~minutes
5303 }{
5304 \l__hwexam_testheading_duration_tl
5305 }~

```

```

5306 (sharp)~for~the~test
5307 };\
5308 Write~the~solutions~to~the~sheet.
5309 \par\noindent
5310 \newcount\check@time\check@time=\l__hwexam_testheading_min_tl
5311 \advance\check@time by -\theassignment@totalmin
5312 The~estimated~time~for~solving~this~exam~is~
5313 {\theassignment@totalmin}-minutes,~
5314 leaving~you~{\the\check@time}-minutes~for~revising~
5315 your~exam.
5316
5317 \par\noindent
5318 \newcount\bonus@pts\bonus@pts=\theassignment@totalpts
5319 \advance\bonus@pts by -\l__hwexam_testheading_reqpts_tl
5320 You~can~reach~{\theassignment@totalpts}-points~if~you~
5321 solve~all~problems.~You~will~only~need~
5322 {\l__hwexam_testheading_reqpts_tl}-points~for~a~perfect~score,~
5323 i.e.~\ {\the\bonus@pts}-points~are~bonus~points.
5324 \vfill
5325 \begin{center}
5326 {
5327 \Large\em You~have~ample~time,~so~take~it~slow~
5328 and~avoid~rushing~to~mistakes!\}[2ex]
5329 Different~problems~test~different~skills~and~
5330 knowledge,~so~do~not~get~stuck~on~one~problem.
5331 }
5332 \vfill\par\resizebox{\textwidth}{!}{\correction@table}\}[3ex]
5333 \end{center}
5334 }{
5335 \newpage
5336 }

```

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

\testspace

```

5337 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

5338 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

5339 \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.

```

5340 <@=problems>
5341 \renewcommand\@problem[3]{
5342 \stepcounter{assignment@probs}
5343 \def\__problemspts{#2}

```

```

5344 \ifx\__problemspts\@empty\else
5345 \addtocounter{assignment@totalpts}{#2}
5346 \fi
5347 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
5348 \xdef\correction@probs{\correction@probs & #1}%
5349 \xdef\correction@pts{\correction@pts & #2}
5350 \xdef\correction@reached{\correction@reached & }
5351 }
5352 \@@=hwexam

```

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

**\correction@table** This macro generates the correction table

```

5353 \newcounter{assignment@probs}
5354 \newcounter{assignment@totalpts}
5355 \newcounter{assignment@totalmin}
5356 \def\correction@probs{\correction@probs@kw}%
5357 \def\correction@pts{\correction@pts@kw}%
5358 \def\correction@reached{\correction@reached@kw}%
5359 \def\after@correction@table{}%
5360 \stepcounter{assignment@probs}
5361 \newcommand\correction@table{
5362 \resizebox{\textwidth}{!}{%
5363 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
5364 &\multicolumn{\theassignment@probs}{c|}|%|
5365 {\footnotesize\correction@forgrading@kw} &\\ \hline
5366 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
5367 \correction@pts & \theassignment@totalpts & \\ \hline
5368 \correction@reached & & \[.7cm]\hline
5369 \end{tabular}}
5370 \ifx\after@correction@table\@empty\else\strut\par\noindent\after@correction@table\fi}
5371 \end{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\bierrfont=../assignments/bierglas
\font\denkerfont=../assignments/denker
\font\uhrfont=../assignments/uhr
\font\warnschildfont=../assignments/achtung

```

```

\newcommand\bierrglas{{\bierrfont\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{\bierrglas}

```