

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

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

2021-12-22

## **Abstract**

TODO

---

\*Version 3.0 (last revised 2021-12-22)

# 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>67</b>
19.1	Generic Path Handling . . . . .	67
19.2	PWD and kpsewhich . . . . .	69
19.3	File Hooks and Tracking . . . . .	70
19.4	MathHub Repositories . . . . .	71
<b>20</b>	<b>gTeX-References Implementation</b>	<b>77</b>
20.1	Document URIs and URLs . . . . .	77
20.2	Setting Reference Targets . . . . .	79
20.3	Using References . . . . .	80
<b>21</b>	<b>gTeX-Modules Implementation</b>	<b>81</b>
21.1	The module environment . . . . .	84
21.2	Invoking modules . . . . .	89
<b>22</b>	<b>gTeX-Module Inheritance Implementation</b>	<b>91</b>
22.1	SMS Mode . . . . .	91
22.2	Inheritance . . . . .	95

<b>23</b>	<b>STEX-Symbols Implementation</b>	<b>100</b>
23.1	Symbol Declarations . . . . .	100
23.2	Notations . . . . .	106
<b>24</b>	<b>STEX-Terms Implementation</b>	<b>114</b>
24.1	Symbol Invocations . . . . .	114
24.2	Terms . . . . .	116
24.3	Notation Components . . . . .	123
<b>25</b>	<b>STEX-Structural Features Implementation</b>	<b>125</b>
25.1	The feature environment . . . . .	125
25.2	Features . . . . .	127
<b>26</b>	<b>STEX-Statements Implementation</b>	<b>132</b>
26.1	Definitions . . . . .	133
26.2	Assertions . . . . .	134
26.3	Examples . . . . .	136
26.4	OMText . . . . .	137
<b>27</b>	<b>The Implementation</b>	<b>139</b>
27.1	Package Options . . . . .	139
27.2	Proofs . . . . .	139
27.3	Justifications . . . . .	145
<b>28</b>	<b>STEX-Others Implementation</b>	<b>147</b>
<b>29</b>	<b>STEX-Metatheory Implementation</b>	<b>148</b>
<b>30</b>	<b>Tikzinput Implementation</b>	<b>151</b>
<b>31</b>	<b>document-structure.sty Implementation</b>	<b>153</b>
31.1	The OMDoc Class . . . . .	153
31.2	Class Options . . . . .	153
31.3	Beefing up the document environment . . . . .	154
31.4	Implementation: OMDoc Package . . . . .	154
31.5	Package Options . . . . .	154
31.6	Document Structure . . . . .	156
31.7	Front and Backmatter . . . . .	159
31.8	Global Variables . . . . .	161
<b>32</b>	<b>MiKoSlides – Implementation</b>	<b>162</b>
32.1	Class and Package Options . . . . .	162
32.2	Notes and Slides . . . . .	164
32.3	Header and Footer Lines . . . . .	168
32.4	Frame Images . . . . .	169
32.5	Colors and Highlighting . . . . .	170
32.6	Sectioning . . . . .	171
32.7	Excursions . . . . .	173

<b>33 The Implementation</b>	<b>175</b>
33.1 Package Options . . . . .	175
33.2 Problems and Solutions . . . . .	176
33.3 Multiple Choice Blocks . . . . .	181
33.4 Including Problems . . . . .	182
33.5 Reporting Metadata . . . . .	183
<b>34 Implementation: The hwexam Class</b>	<b>185</b>
34.1 Class Options . . . . .	185
<b>35 Implementation: The hwexam Package</b>	<b>187</b>
35.1 Package Options . . . . .	187
35.2 Assignments . . . . .	188
35.3 Including Assignments . . . . .	191
35.4 Typesetting Exams . . . . .	192
35.5 Leftovers . . . . .	194

**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  $\textcolor{teal}{\$}\textcolor{teal}{add}\textcolor{teal}{\$}$  adds two elements, as in  $\textcolor{teal}{\$}\textcolor{teal}{add}\textcolor{teal}{ab}\textcolor{teal}{\$}$ .
```

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

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

### Example 7

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

$\textcolor{teal}{Multiplication}$  (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 \_x \int \_y \int \_z 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** ( $\langle\log\text{-}prefix\rangle*$ ) Logs debugging information with the given prefixes to the terminal, or all if **all** is given.

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

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

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

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

**image** ( $\langle\text{boolean}\rangle$ ) 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\text{-}prefix\rangle</math>} {<math>\langle\text{message}\rangle</math>}</code>
-----------------------------	---

---

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

---

<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> X <sub>2</sub> e and L <sup>A</sup> T <sub>E</sub> X <sub>3</sub> conditionals for L <sup>A</sup> T <sub>E</sub> X <sub>ML</sub> .
<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>X<sub>ML</sub> or S<sup>C</sup>A<sup>L</sup>L<sup>A</sup>T<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:N</code>	$\underline{TF}$ $\star$

---

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
<code>\c_stex_pwd_str</code>	(heuristically guessed) full path to the main file, based on the PWD and <code>\jobname</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`

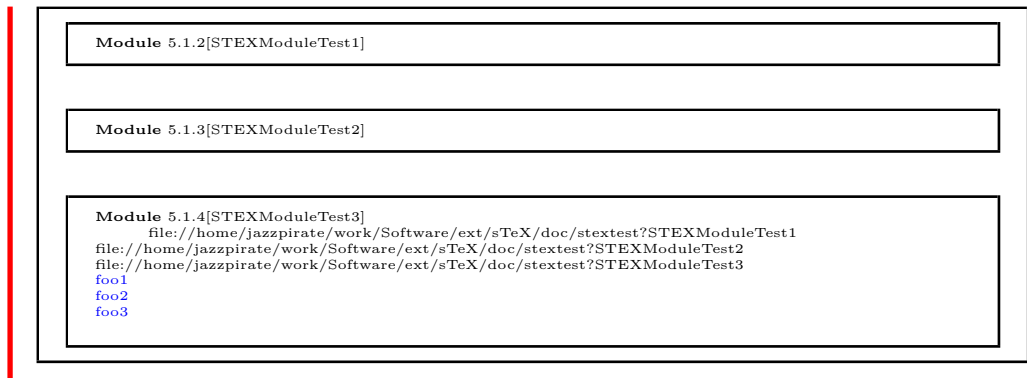
---

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:  $\rightarrow$  `\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo?foo}<`

Meaning:  $\rightarrow$  `\protect \bar <`

**Module 6.1.2[Importtest]**

Meaning:  $\rightarrow$  `\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo?foo}<`

**Module 6.1.3[Importtest2]**

Meaning:  $\rightarrow$  `\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]**  
Meaning: `>macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest1?foo}<`

**Module 6.1.6[UseTest3]**  
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?collec`,  
`http://mathhub.info/sTeX?Metatheory?fromto`, `http://mathhub.info/sTeX?Metatheory?apply`, `http://mathhub.info/sTeX?Metatheory?collec`,  
`http://mathhub.info/sTeX?Metatheory?seqtype`, `http://mathhub.info/sTeX?Metatheory?sequence-index`, `http://mathhub.info/sTeX?Metatheory?mathematical-structure`,  
`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

---

<u><code>\symdecl</code></u>	<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}_\comp{#2}}
\end{module}
```

Module 7.1.2[NotationTest]

---

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

---

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

### Test 13

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

Module 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!</code> [addition] 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/> <code>\infprec</code> <code>\neginfprec</code> <hr/>	Maximal and minimal notation precedences.
<hr/> <code>\dobrackets</code> <hr/>	<code>\dobrackets {⟨body⟩}</code>  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 <code>\withbrackets</code> .
<hr/> <code>\withbrackets</code> <hr/>	<code>\withbrackets ⟨left⟩ ⟨right⟩ {⟨body⟩}</code>  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 <code>\left</code> and <code>\right</code> 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]  
 $\langle x20x20a^b{}_c \rangle$  and  $\langle x20x20a^b{}_c \rangle$ .

### 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\mid [ #2 ] ^{#3} \comp\rangle } }
$\foobar{a}{b,c,d,e,f}g$ and $\foobar{foo}{a}{b,c}g$ and $\foobar{abc}$

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

```

Module 8.1.2[MathTest2]  
 $\langle x20x20a|[b,c,d,e,f]{}^g \rangle$  and  $\langle x20x20a|[b,c]{}^g \rangle$  and  $\langle x20x20a|[b]{}^c \rangle$   
 $a+(b\cdot c)$  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]
  some a and some b and also some c here.
  some a and some b and also some c here.

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

---

---

`\stex_highlight_term:nn`

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

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

---

`\comp`

`\comp{<args>}`

`\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**o**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 its 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  $\S\TeX$  collection, a version of  $\TeX$ / $\LaTeX$  that allows to markup  $\TeX$ / $\LaTeX$  documents semantically without leaving the document format, essentially turning  $\TeX$ / $\LaTeX$  into a document format for mathematical knowledge management (MKM).

This package supplies an infrastructure for writing OMDOC documents in  $\LaTeX$ . This includes a simple structure sharing mechanism for  $\S\TeX$  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  $\S\TeX$  sources, or after translation.

### 14.1 Introduction

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

The `omdoc` package supplies macros and environments that allow to label document fragments and to reference them later in the same document or in other documents. In essence, this enhances the document-as-trees model to documents-as-directed-acyclic-graphs (DAG) model. This structure can be used by MKM systems for added-value services, either directly from the  $\S\TeX$  sources, or after translation. Currently, trans-document referencing provided by this package can only be used in the  $\S\TeX$  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  $\text{\LaTeX}$  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><br><code>notes</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>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.





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-compatible}
22 \RequirePackage{morewrites}
23
24 Package options:
25 \keys_define:nn { stex } {
26   debug      .clist_set:N = \c_stex_debug_clist ,
27   showmods   .bool_set:N = \c_stex_showmods_bool ,
```

```

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

```

**\stex** The  $\TeX$  logo:

**\sTeX**

```

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

```

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

## 18.3 Messages and logging

```

40 <@@=stex_log>

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

```

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

```

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

```

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

Redirecting messages:

```

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

```

## 18.4 Persistence

75  $\langle @@=stex\_persist \rangle$

$\backslash c\_stex\_persist\_sms\_iow$  File variable used for the sms-File

```

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

```

(End definition for  $\backslash c\_stex\_persist\_sms\_iow$ .)

$\backslash stex\_add\_to\_sms:n$  Adds the provided code to the .sms-file of the document.

```

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

```

(End definition for  $\backslash stex\_add\_to\_sms:n$ . This function is documented on page 9.)

## 18.5 HTML Annotations

94  $\langle @@=stex\_annotate \rangle$   
95  $\backslash RequirePackage\{scalatex\}$

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

```

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

```

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

```

\latexml_if_p: 97 \ifcsname if@latexml\endcsname\else
\latexml_if:TF 98   \expandafter\newif\csname if@latexml\endcsname\@latexmlfalse
99 \fi

```

```

100
101 \prg_new_conditional:Nnn \latexml_if: {p, T, F, TF} {
102   \if@latexml
103     \prg_return_true:
104   \else:
105     \prg_return_false:
106   \fi:
107 }

```

(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
108 \tl_new:N \l__stex_annotate_arg_tl
109 \tl_const:Nx \c__stex_annotate_emptyarg_tl {
110   \scalatex_if:TF {
111     \scalatex_direct_HTML:n { \c_ampersand_str lrm; }
112   }{-}
113 }

```

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

```

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

```

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

\l\_stex\_html\_do\_output\_bool Whether to (locally) produce HTML output

```

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

```

(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

```

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

```

(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, S<sup>C</sup>A<sup>L</sup>T<sub>E</sub>X, p<sup>D</sup>f<sup>L</sup>at<sub>E</sub>x).

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

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

```

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

```

(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

```

219 <@@=stex_language>

```

`\c_stex_languages_prop`  
`\c_stex_language_abbrevs_prop`

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

```

220 \prop_const_from_keyval:Nn \c_stex_languages_prop {
221     en = english ,
222     de = ngerman ,
223     ar = arabic ,
224     bg = bulgarian ,
225     ru = russian ,
226     fi = finnish ,
227     ro = romanian ,
228     tr = turkish ,
229     fr = french
230 }

```

```

231
232 \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop {
233   english   = en ,
234   ngerman   = de ,
235   arabic    = ar ,
236   bulgarian = bg ,
237   russian   = ru ,
238   finnish   = fi ,
239   romanian  = ro ,
240   turkish   = tr ,
241   french    = fr
242 }
243 % todo: chinese simplified (zhs)
244 %       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:

```

245 \clist_if_empty:NF \c_stex_languages_clist {
246   \clist_clear:N \l_tmpa_clist
247   \clist_map_inline:Nn \c_stex_languages_clist {
248     \prop_get:NnNTF \c_stex_languages_prop { #1 } \l_tmpa_str {
249       \clist_put_right:No \l_tmpa_clist \l_tmpa_str
250     } {
251       \msg_error:nxx{stex}{error/unknownlanguage}{\l_tmpa_str}
252     }
253   }
254   \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
255   \RequirePackage[\clist_use:Nn \l_tmpa_clist,]{babel}
256 }

```

## 18.7 Activating/Deactivating Macros

`\stex_deactivate_macro:Nn`

```

257 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
258   \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
259   \def#1{
260     \msg_error:nnnn{stex}{error/deactivated-macro}{#1}{#2}
261   }
262 }

```

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

`\stex_reactivate_macro:N`

```

263 \cs_new_protected:Nn \stex_reactivate_macro:N {
264   \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname
265 }

```

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

```

266 </package>

```



## Chapter 19

# STEX -MathHub Implementation

```
267 <*package>
268
269 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
270
271 <@@=stex_path>
272
273 Warnings and error messages
274 \msg_new:nnn{stex}{error/norepository}{
275   No~archive~#1~found~in~#2
276 }
277 \msg_new:nnn{stex}{error/notinarchive}{
278   Not~currently~in~an~archive,~but~\detokenize{#1}~
279   needs~one!
280 }
281 \msg_new:nnn{stex}{error/nofile}{
282   \detokenize{#1}~could~not~find~file~#2
283 }
```

### 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
282 \cs_new_protected:Nn \stex_path_from_string:Nn {
283   \str_set:Nx \l_tmpa_str { #2 }
284   \str_if_empty:NTF \l_tmpa_str {
285     \seq_clear:N #1
286   }{
287     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
288     \sys_if_platform_windows:T{
289       \seq_clear:N \l_tmpa_tl
290       \seq_map_inline:Nn #1 {
291         \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
292         \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl

```

```

293     }
294     \seq_set_eq:NN #1 \l_tmpa_tl
295   }
296   \stex_path_canonicalize:N #1
297 }
298 }
299 \cs_generate_variant:Nn \stex_path_from_string:Nn
300 { 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`

```

301 \cs_new_protected:Nn \stex_path_to_string:NN {
302   \exp_args:NNe \str_set:Nn #2 { \seq_use:Nn #1 / }
303 }
304
305 \cs_new:Nn \stex_path_to_string:N {
306   \seq_use:Nn #1 /
307 }

```

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

`\c__stex_path_dot_str`  
`\c__stex_path_up_str`

. and .., respectively.

```

308 \str_const:Nn \c__stex_path_dot_str {.}
309 \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.

```

310 \cs_new_protected:Nn \stex_path_canonicalize:N {
311   \seq_if_empty:NF #1 {
312     \seq_clear:N \l_tmpa_seq
313     \seq_get_left:NN #1 \l_tmpa_tl
314     \str_if_empty:NT \l_tmpa_tl {
315       \seq_put_right:Nn \l_tmpa_seq {}
316     }
317     \seq_map_inline:Nn #1 {
318       \str_set:Nn \l_tmpa_tl { ##1 }
319       \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_dot_str {} {
320         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
321           \seq_if_empty:NTF \l_tmpa_seq {
322             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
323               \c__stex_path_up_str
324             }
325           }{
326             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
327             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
328               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
329                 \c__stex_path_up_str
330               }
331             }{
332               \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
333             }

```

```

334     }
335   }{
336     \str_if_empty:NF \l_tmpa_tl {
337       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
338     }
339   }
340 }
341 }
342 \seq_gset_eq:NN #1 \l_tmpa_seq
343 }
344 }

```

(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`

```

345 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
346   \seq_if_empty:NTF #1 {
347     \prg_return_false:
348   }{
349     \seq_get_left:NN #1 \l_tmpa_tl
350     \str_if_empty:NTF \l_tmpa_tl {
351       \prg_return_true:
352     }{
353       \prg_return_false:
354     }
355   }
356 }

```

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

## 19.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

357 \str_new:N\l_stex_kpsewhich_return_str
358 \cs_new_protected:Nn \stex_kpsewhich:n {
359   \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
360   \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
361   \tl_trim_spaces:N \l_stex_kpsewhich_return_str
362 }

```

(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`

```

363 \sys_if_platform_windows:TF{
364   \stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
365 }{
366   \stex_kpsewhich:n{-var-value~PWD}
367 }
368
369 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
370 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
371 \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

372 `<@@=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

373 `\seq_gclear_new:N\g__stex_files_stack`

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`

`\c_stex_mainfile_str`

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

375 `\stex_path_from_string:Nn \c_stex_mainfile_seq`

376 `\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`.

```

377 \seq_gclear_new:N\g_stex_currentfile_seq
378 \AddToHook{file/before}{
379   \stex_path_from_string:Nn\g_stex_currentfile_seq{\CurrentFilePath}
380   \stex_path_if_absolute:NTF\g_stex_currentfile_seq{
381     \exp_args:NNe\seq_put_right:Nn\g_stex_currentfile_seq{\CurrentFile}
382   }{
383     \stex_path_from_string:Nn\g_stex_currentfile_seq{
384       \c_stex_pwd_str/\CurrentFilePath/\CurrentFile
385     }
386   }
387   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
388   \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
389 }
390 \AddToHook{file/after}{
391   \seq_if_empty:NF\g__stex_files_stack{
392     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
393   }
394   \seq_if_empty:NTF\g__stex_files_stack{
395     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
396   }{
397     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
398     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
399   }
400 }
```

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

## 19.4 MathHub Repositories

```

401 <@@=stex_mathhub>

\mathhub
\c_stex_mathhub_seq
\c_stex_mathhub_str
402 \str_if_empty:NTF\mathhub{
403   \stex_kpsewhich:n{-var-value~MATHHUB}
404   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
405
406   \str_if_empty:NTF\c_stex_mathhub_str{
407     \msg_warning:nn{stex}{warning/nomathhub}
408   }{
409     \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
410     \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
411   }
412 }{
413   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
414   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
415     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
416       \c_stex_pwd_str/\mathhub
417     }
418   }
419   \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
420   \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
421 }

```

(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
422 \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
423   \str_set:Nx \l_tmpa_str { #1 }
424   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
425     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
426     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
427     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
428     \__stex_mathhub_find_manifest:N \l_tmpa_seq
429     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
430       \msg_error:nnnn{stex}{error/norepository}{#1}{
431         \stex_path_to_string:N \c_stex_mathhub_str
432       }
433     } {
434       \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
435     }
436   }
437 }

```

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

```

\l__stex_mathhub_manifest_file_seq
438 \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`:

```

439 \cs_new_protected:Nn \_stex_mathhub_find_manifest:N {
440   \seq_set_eq:NN \l_tmpa_seq #1
441   \bool_set_true:N \l_tmpa_bool
442   \bool_while_do:Nn \l_tmpa_bool {
443     \seq_if_empty:NTF \l_tmpa_seq {
444       \bool_set_false:N \l_tmpa_bool
445     }{
446       \file_if_exist:nTF{
447         \stex_path_to_string:N \l_tmpa_seq/MANIFEST.MF
448       }{
449         \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
450         \bool_set_false:N \l_tmpa_bool
451       }{
452         \file_if_exist:nTF{
453           \stex_path_to_string:N \l_tmpa_seq/META-INF/MANIFEST.MF
454         }{
455           \seq_put_right:Nn \l_tmpa_seq{META-INF}
456           \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
457           \bool_set_false:N \l_tmpa_bool
458         }{
459           \file_if_exist:nTF{
460             \stex_path_to_string:N \l_tmpa_seq/meta-inf/MANIFEST.MF
461           }{
462             \seq_put_right:Nn \l_tmpa_seq{meta-inf}
463             \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
464             \bool_set_false:N \l_tmpa_bool
465           }{
466             \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
467           }
468         }
469       }
470     }
471   }
472   \seq_set_eq:NN \l__stex_mathhub_manifest_file_seq \l_tmpa_seq
473 }

```

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

`\c_stex_mathhub_manifest_ior` File variable used for MANIFEST-files

```

474 \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:

```

475 \cs_new_protected:Nn \_stex_mathhub_parse_manifest:n {
476   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
477   \ior_open:Nn \c_stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
478   \ior_map_inline:Nn \c_stex_mathhub_manifest_ior {
479     \str_set:Nn \l_tmpa_str {##1}
480     \exp_args:NNoo \seq_set_split:Nnn
481       \l_tmpb_seq \c_colon_str \l_tmpa_str
482     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {

```

```

483 \exp_args:NNe \str_set:Nn \l_tmpb_tl {
484 \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
485 }
486 \exp_args:No \str_case:nnTF \l_tmpa_tl {
487 {id} {
488 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
489 { id } \l_tmpb_tl
490 }
491 {narration-base} {
492 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
493 { narr } \l_tmpb_tl
494 }
495 {url-base} {
496 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
497 { docurl } \l_tmpb_tl
498 }
499 {source-base} {
500 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
501 { ns } \l_tmpb_tl
502 }
503 {ns} {
504 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
505 { ns } \l_tmpb_tl
506 }
507 {dependencies} {
508 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
509 { deps } \l_tmpb_tl
510 }
511 }{}{}
512 }{}
513 }
514 \ior_close:N \c__stex_mathhub_manifest_ior
515 }

```

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

`\stex_set_current_repository:n`

```

516 \cs_new_protected:Nn \stex_set_current_repository:n {
517 \stex_require_repository:n { #1 }
518 \prop_set_eq:Nc \l_stex_current_repository_prop {
519 c_stex_mathhub_#1_manifest_prop
520 }
521 }

```

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

`\stex_require_repository:n`

```

522 \cs_new_protected:Nn \stex_require_repository:n {
523 \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
524 \stex_debug:nn{mathhub}{Opening~archive:~#1}
525 \__stex_mathhub_do_manifest:n { #1 }
526 \exp_args:Nx \stex_add_to_sms:n {
527 \prop_const_from_keyval:cn { c_stex_mathhub_#1_manifest_prop } {
528 id = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { id } ,
529 ns = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { ns } ,

```

```

530     narr = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { narr } ,
531     deps = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { deps }
532   }
533 }
534 }
535 }

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

536 \prop_new:N \l_stex_current_repository_prop
537
538 \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
539 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
540   \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
541 } {
542   \__stex_mathhub_parse_manifest:n { main }
543   \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
544   \l_tmpa_str
545   \prop_set_eq:cN { c_stex_mathhub_ \l_tmpa_str _manifest_prop }
546   \c_stex_mathhub_main_manifest_prop
547   \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
548   \stex_debug:nn{mathhub}{Current~repository:~
549   \prop_item:Nn \l_stex_current_repository_prop {id}
550 }
551 }

```

(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.

```

552 \cs_new_protected:Nn \stex_in_repository:nn {
553   \str_set:Nx \l_tmpa_str { #1 }
554   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
555   \str_if_empty:NTF \l_tmpa_str {
556     \exp_args:Ne \l_tmpa_cs{
557       \prop_item:Nn \l_stex_current_repository_prop { id }
558     }
559   }{
560     \stex_require_repository:n \l_tmpa_str
561     \str_set:Nx \l_tmpa_str { #1 }
562     \exp_args:Nne \use:nn {
563       \stex_set_current_repository:n \l_tmpa_str
564       \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
565     }{
566       \stex_set_current_repository:n {
567         \prop_item:Nn \l_stex_current_repository_prop { id }
568       }
569     }
570   }
571 }

```

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



**\inputref**  
**\inputref:nn**

```

572 \newif \ifinputref \inputreffalse
573
574 \cs_new_protected:Nn \inputref:nn {
575   \stex_in_repository:nn {#1} {
576     \ifinputref
577       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
578     \else
579       \inputreftrue
580       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
581       \inputreffalse
582     \fi
583   }
584 }
585 \NewDocumentCommand \inputref { 0{} m}{
586   \inputref:nn{ #1 }{ #2 }
587 }

```

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

**\mhp**

```

588 \def \mhp #1 #2 {
589   \exp_args:Ne \str_if_eq:nnTF{#1}{}{
590     \c_stex_mathhub_str /
591     \prop_item:Nn \l_stex_current_repository_prop { id }
592     / source / #2
593   }{
594     \c_stex_mathhub_str / #1 / source / #2
595   }
596 }

```

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

**\libinput**

```

597 \cs_new_protected:Npn \libinput #1 {
598   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
599     \msg_error:nnn{stex}{error/notinarchive}\libinput
600   }
601   \bool_set_false:N \l_tmpa_bool
602   \tl_clear:N \l_tmpa_tl
603   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
604   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
605   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str
606   \seq_pop_left:NNT \l_tmpb_seq \l_tmpb_str {
607     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
608     \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
609       / meta-inf / lib / #1.tex}{
610       \bool_set_true:N \l_tmpa_bool
611       \tl_put_right:Nx \l_tmpa_tl {
612         \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
613           / meta-inf / lib / #1.tex}
614       }
615     }{}
616   }

```

```

617 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
618 / \l_tmpa_str / lib / #1.tex
619 }{
620   \bool_set_true:N \l_tmpa_bool
621   \tl_put_right:Nx \l_tmpa_tl {
622     \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
623       / \l_tmpa_str / lib / #1.tex}
624   }
625 }{}
626 \bool_if:NF \l_tmpa_bool {
627   \msg_error:nnnn{stex}{error/nofile}\libinput{#1.tex}
628 }
629 \l_tmpa_tl
630 }

```

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

```

631 </package>

```

## Chapter 20

# STEX -References Implementation

```
632 <*package>
633
634 %%%%%%%%%% references.dtx %%%%%%%%%%
635
636 \RequirePackage{hyperref}
637 %\RequirePackage{cleveref}
638 <@@=stex_refs>
639
640 \Warnings and error messages
641
642 \iow_new:N \c__stex_refs_refs_iow
643 \AddToHook{begindocument}{
644   \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
645 }
646 \AddToHook{enddocument}{
647   \iow_close:N \c__stex_refs_refs_iow
648 }
649
650 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
651
652 \NewDocumentCommand \STEXreftitle { m } {
653   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
654 }
655
```

### 20.1 Document URIs and URLs

```
653 \seq_new:N \g__stex_refs_all_refs_seq
654
655 \str_new:N \l_stex_current_docns_str
656
657 \cs_new_protected:Nn \stex_get_document_uri: {
658   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
659   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
660   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
661   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
662 }
663
```

```

662 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
663
664 \str_clear:N \l_tmpa_str
665 \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
666   \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
667 }
668
669 \str_if_empty:NTF \l_tmpa_str {
670   \str_set:Nx \l_stex_current_docns_str {
671     file:/\stex_path_to_string:N \l_tmpa_seq
672   }
673 }{
674   \bool_set_true:N \l_tmpa_bool
675   \bool_while_do:Nn \l_tmpa_bool {
676     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
677     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
678       {source} { \bool_set_false:N \l_tmpa_bool }
679     }{}{
680       \seq_if_empty:NT \l_tmpa_seq {
681         \bool_set_false:N \l_tmpa_bool
682       }
683     }
684   }
685
686   \seq_if_empty:NTF \l_tmpa_seq {
687     \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
688   }{
689     \str_set:Nx \l_stex_current_docns_str {
690       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
691     }
692   }
693 }
694 }
695
696 \str_new:N \l_stex_current_docurl_str
697 \cs_new_protected:Nn \stex_get_document_url: {
698   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
699   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
700   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
701   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
702   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
703
704   \str_clear:N \l_tmpa_str
705   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
706     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
707       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
708     }
709   }
710
711   \str_if_empty:NTF \l_tmpa_str {
712     \str_set:Nx \l_stex_current_docurl_str {
713       file:/\stex_path_to_string:N \l_tmpa_seq
714     }
715   }{
716     \bool_set_true:N \l_tmpa_bool

```

```

716 \bool_while_do:Nn \l_tmpa_bool {
717   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
718   \exp_args:No \str_case:nnTF { \l_tmpb_str } {
719     {source} { \bool_set_false:N \l_tmpa_bool }
720   }{}{
721     \seq_if_empty:NT \l_tmpa_seq {
722       \bool_set_false:N \l_tmpa_bool
723     }
724   }
725 }
726
727 \seq_if_empty:NTF \l_tmpa_seq {
728   \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
729 }{
730   \str_set:Nx \l_stex_current_docurl_str {
731     \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
732   }
733 }
734 }
735 }

```

## 20.2 Setting Reference Targets

```

736 \str_const:Nn \c__stex_refs_url_str{URL}
737 \str_const:Nn \c__stex_refs_ref_str{REF}
738 % @currentlabel -> number
739 % @currentlabelname -> title
740 % @currentHref -> name.number <- id of some kind
741 % \theH# -> \arabic{section}
742 % \the# -> number
743 % \hyper@makecurrent{#}
744 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
745   \stex_get_document_uri:
746   \str_set:Nx \l_tmpa_str { #1 }
747   \str_if_empty:NT \l_tmpa_str {
748     \int_zero:N \l_tmpa_int
749     \bool_set_true:N \l_tmpa_bool
750     \bool_while_do:Nn \l_tmpa_bool {
751       \cs_if_exist:cTF {
752         sref_\l_stex_current_docns_str\c_hash_str REF_\int_use:N \l_tmpa_int _type
753       }{
754         \int_incr:N \l_tmpa_int
755       }{
756         \str_set:Nx \l_tmpa_str { REF_\int_use:N \l_tmpa_int }
757         \bool_set_false:N \l_tmpa_bool
758       }
759     }
760   }
761   \str_set:Nx \l_tmpa_str {
762     \l_stex_current_docns_str\c_hash_str\l_tmpa_str
763   }
764   \seq_gput_right:No \g__stex_refs_all_refs_seq \l_tmpa_str
765   \stex_if_smsmode:TF {
766     \stex_get_document_url:

```

```

767 \str_gset_eq:cN {sref_url_\l_tmpa_str_str}\l_stex_current_docurl_str
768 \str_gset_eq:cN {sref_\l_tmpa_str_type}\c__stex_refs_url_str
769 }{
770 \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~::~\expandafter{\@currentlabel~in~\exp_a
771 \exp_after:wN\label\exp_after:wN{sref_\l_tmpa_str}
772 \str_gset:cN {sref_\l_tmpa_str_type}\c__stex_refs_ref_str
773 }
774 }

775 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
776 \str_gset_eq:cN {sref_sym_#1_uri} \l_stex_current_docns_str
777 }

```

## 20.3 Using References

```

778 \keys_define:nn { stex / sref } {
779 linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
780 fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
781 pre           .tl_set:N = \l__stex_refs_pre_tl ,
782 post          .tl_set:N = \l__stex_refs_post_tl ,
783 indoc         .str_set_x:N = \l__stex_refs_repo_str ,
784 }
785
786 \cs_new_protected:Nn \__stex_refs_args:n {
787 \tl_clear:N \l__stex_refs_linktext_tl
788 \tl_clear:N \l__stex_refs_fallback_tl
789 \tl_clear:N \l__stex_refs_pre_tl
790 \tl_clear:N \l__stex_refs_post_tl
791 \str_clear:N \l__stex_refs_repo_str
792 \keys_set:nn { stex / sref } { #1 }
793 }
794
795 \end{package}

```

## Chapter 21

# STEX -Modules Implementation

```
796 <*package>
797
798 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
799
800 <@@=stex_modules>
      Warnings and error messages
801 \msg_new:nnn{stex}{error/unknownmodule}{
802   No~module~#1~found
803 }
804 \msg_new:nnn{stex}{error/syntax}{
805   Syntax~error:~#1
806 }
807 \msg_new:nnn{stex}{error/siglanguage}{
808   Module~#1~declares~signature~#2,~but~does~not~
809   declare~its~language
810 }
```

`\l_stex_current_module_prop` The current module:

```
811 \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

```
812 \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`  
`\g_stex_module_files_prop`

```
813 \seq_new:N \g_stex_modules_in_file_seq
814 \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
815 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
816   \prop_if_empty:NTF \l_stex_current_module_prop
817   \prg_return_false: \prg_return_true:
818 }

```

(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
819 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
820   \prop_if_exist:cTF { c_stex_module_#1_prop }
821   \prg_return_true: \prg_return_false:
822 }

```

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

```

\stex_add_to_current_module:n
\STEXexport
823 \cs_new_protected:Nn \stex_add_to_current_module:n {
824   \prop_get:NnN \l_stex_current_module_prop { content } \l_tmpa_tl
825   \tl_put_right:Nn \l_tmpa_tl { #1 }
826   \prop_put:Nno \l_stex_current_module_prop { content } { \l_tmpa_tl }
827 }
828 \cs_new_protected:Npn \STEXexport {
829   \begingroup
830   \newlinechar=-1\relax
831   \endlinechar=-1\relax
832   %\catcode'\ = 9\relax
833   \expandafter\endgroup\STEXexport:n
834 }
835 \cs_new_protected:Nn \STEXexport:n {
836   \ignorespaces #1
837   \stex_add_to_current_module:n { \ignorespaces #1 }
838   \stex_smsmode_set_codes:
839 }
840 \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
841 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
842   \str_set:Nx \l_tmpa_str { #1 }
843   \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
844   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
845   \prop_put:Nno \l_stex_current_module_prop { constants } \l_tmpa_seq
846 }

```

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

```

\stex_add_import_to_current_module:n
847 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
848   \str_set:Nx \l_tmpa_str { #1 }
849   \prop_get:NnN \l_stex_current_module_prop { imports } \l_tmpa_seq
850   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
851   \prop_put:Nno \l_stex_current_module_prop { imports } \l_tmpa_seq
852 }

```



(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).

```

853 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
854   \str_set:Nx \l_tmpa_str { #1 }
855   \seq_set_eq:NN \l_tmpa_seq #2
856   % split off file extension
857   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
858   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
859   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
860   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
861
862   \bool_set_true:N \l_tmpa_bool
863   \bool_while_do:Nn \l_tmpa_bool {
864     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
865     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
866       {source} { \bool_set_false:N \l_tmpa_bool }
867     }{}{
868       \seq_if_empty:NT \l_tmpa_seq {
869         \bool_set_false:N \l_tmpa_bool
870       }
871     }
872   }
873
874   \seq_if_empty:NTF \l_tmpa_seq {
875     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
876   }{
877     \str_set:Nx \l_stex_modules_ns_str {
878       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
879     }
880   }
881 }

```

(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`

```

882 \str_new:N \l_stex_modules_ns_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.

```

883 \cs_new_protected:Nn \stex_modules_current_namespace: {
884   \prop_get:NnNTF \l_stex_current_repository_prop { ns } \l_tmpa_str {
885     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
886   }{
887     % split off file extension
888     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
889     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
890     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
891     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
892     \seq_put_right:No \l_tmpa_seq \l_tmpb_str

```

```

893     \str_set:Nx \l_stex_modules_ns_str {
894         file:/\stex_path_to_string:N \l_tmpa_seq
895     }
896 }
897 }

```

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

## 21.1 The module environment

module arguments:

```

898 \keys_define:nn { stex / module } {
899     title          .str_set_x:N = \l_stex_module_title_str ,
900     ns             .str_set_x:N = \l_stex_module_ns_str ,
901     lang           .str_set_x:N = \l_stex_module_lang_str ,
902     sig            .str_set_x:N = \l_stex_module_sig_str ,
903     creators       .str_set_x:N = \l_stex_module_creators_str ,
904     contributors   .str_set_x:N = \l_stex_module_contributors_str ,
905     meta           .str_set_x:N = \l_stex_module_meta_str
906 }
907
908 \cs_new_protected:Nn \__stex_modules_args:n {
909     \str_clear:N \l_stex_module_title_str
910     \str_clear:N \l_stex_module_ns_str
911     \str_clear:N \l_stex_module_lang_str
912     \str_clear:N \l_stex_module_sig_str
913     \str_clear:N \l_stex_module_creators_str
914     \str_clear:N \l_stex_module_contributors_str
915     \str_clear:N \l_stex_module_meta_str
916     \keys_set:nn { stex / module } { #1 }
917 }
918
919 % module parameters here? In the body?
920

```

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

```

921 \cs_new_protected:Nn \stex_module_setup:nn {
922     \str_set:Nx \l_stex_module_name_str { #2 }
923     \__stex_modules_args:n { #1 }
924
925     First, we set up the name and namespace of the module.
926     Are we in a nested module?
927
928     \stex_if_in_module:TF {
929         % Nested module
930         \prop_get:NnN \l_stex_current_module_prop
931         { ns } \l_stex_module_ns_str
932         \str_set:Nx \l_stex_module_name_str {
933             \prop_item:Nn \l_stex_current_module_prop
934             { name } / \l_stex_module_name_str
935         }
936     }{
937         % not nested:
938         \str_if_empty:NT \l_stex_module_ns_str {

```

```

935     \stex_modules_current_namespace:
936     \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
937     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
938       / {\l_stex_module_ns_str}
939     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
940     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
941       \str_set:Nx \l_stex_module_ns_str {
942         \stex_path_to_string:N \l_tmpa_seq
943       }
944     }
945   }
946 }

```

Next, we determine the language of the module:

```

947 \str_if_empty:NT \l_stex_module_lang_str {
948   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
949   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
950   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
951   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
952   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
953     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
954       inferred~from~file~name}
955     \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
956   }
957 }
958
959 \str_if_empty:NF \l_stex_module_lang_str {
960   \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
961   \l_tmpa_str {
962     \ltx@ifpackageloaded{babel}{
963       \exp_args:Nx \selectlanguage { \l_tmpa_str }
964     }{}
965   } {
966     \msg_error:nnn{stex}{error/unknownlanguage}{\l_tmpa_str}
967   }
968 }

```

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

```

969 \str_if_empty:NTF \l_stex_module_sig_str {
970   \str_clear:N \l_tmpa_str
971   \seq_clear:N \l_tmpa_seq
972   \tl_clear:N \l_tmpa_tl
973   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
974     name      = \l_stex_module_name_str ,
975     ns        = \l_stex_module_ns_str ,
976     imports   = \exp_not:o { \l_tmpa_seq } ,
977     constants = \exp_not:o { \l_tmpa_seq } ,
978     content   = \exp_not:o { \l_tmpa_tl } ,
979     file      = \exp_not:o { \g_stex_currentfile_seq } ,
980     lang      = \l_stex_module_lang_str ,
981     sig       = \l_stex_module_sig_str ,
982     meta      = \l_stex_module_meta_str
983   }

```

```

984 }{
985   \str_if_empty:NT \l_stex_module_lang_str {
986     \msg_error:nnnn{stex}{error/siglanguage}{
987       \l_stex_module_ns_str?\l_stex_module_name_str
988     }\l_stex_module_sig_str}
989   }
990
991   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
992   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
993   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
994   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
995   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
996   \str_set:Nx \l_tmpa_str {
997     \stex_path_to_string:N \l_tmpa_seq /
998     \l_tmpa_str . \l_stex_module_sig_str .tex
999   }
1000   \IfFileExists \l_tmpa_str {
1001     \exp_args:No \stex_in_smsmode:nn { \l_tmpa_str } {
1002       \seq_clear:N \l_stex_all_modules_seq
1003       \prop_clear:N \l_stex_current_module_prop
1004       \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1005       \input { \l_tmpa_str }
1006     }
1007   }{
1008     \msg_error:nnn{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1009   }
1010   \stex_activate_module:n {
1011     \l_stex_module_ns_str ? \l_stex_module_name_str
1012   }
1013   \prop_set_eq:Nc \l_stex_current_module_prop {
1014     c_stex_module_
1015     \l_stex_module_ns_str ?
1016     \l_stex_module_name_str
1017     _prop
1018   }
1019 }

```

We load the metatheory:

```

1020 \str_if_empty:NT \l_stex_module_meta_str {
1021   \str_set:Nx \l_stex_module_meta_str {
1022     \c_stex_metatheory_ns_str ? Metatheory
1023   }
1024 }
1025 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1026   \exp_args:Nx \stex_add_to_current_module:n {
1027     \stex_activate_module:n {\l_stex_module_meta_str}
1028   }
1029   \stex_activate_module:n {\l_stex_module_meta_str}
1030 }
1031 }

```

*(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}

1032 \cs_new_protected:Nn \__stex_modules_begin_module:nn {
1033   \stex_reactivate_macro:N \STEXexport
1034   \stex_reactivate_macro:N \importmodule
1035   \stex_reactivate_macro:N \symdecl
1036   \stex_reactivate_macro:N \notation
1037   \stex_reactivate_macro:N \symdef
1038   \stex_module_setup:nn{#1}{#2}
1039
1040   \stex_debug:nn{modules}{
1041     New~module:\\
1042     Namespace:~\l_stex_module_ns_str\\
1043     Name:~\l_stex_module_name_str\\
1044     Language:~\l_stex_module_lang_str\\
1045     Signature:~\l_stex_module_sig_str\\
1046     Metatheory:~\l_stex_module_meta_str\\
1047     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1048   }
1049
1050   \seq_put_right:Nx \l_stex_all_modules_seq {
1051     \l_stex_module_ns_str ? \l_stex_module_name_str
1052   }
1053
1054   \seq_gput_right:Nx \g_stex_modules_in_file_seq
1055     { \l_stex_module_ns_str ? \l_stex_module_name_str }
1056
1057   \stex_if_smsmode:TF {
1058     \stex_smsmode_set_codes:
1059   } {
1060     \begin{stex_annotate_env} {theory} {
1061       \l_stex_module_ns_str ? \l_stex_module_name_str
1062     }
1063
1064     \stex_annotate_invisible:nnn{header}{} {
1065       \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1066       \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1067       \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1068         \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1069       }
1070     }
1071   }
1072   % TODO: Inherit metatheory for nested modules?
1073 }
1074 \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}

1075 \cs_new_protected:Nn \__stex_modules_end_module: {
1076   \str_set:Nx \l_tmpa_str {
1077     c_stex_module_
1078     \prop_item:Nn \l_stex_current_module_prop { ns } ?
1079     \prop_item:Nn \l_stex_current_module_prop { name }
1080     _prop

```

```

1081 }
1082 %^^A \prop_new:c { \l_tmpa_str }
1083 \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
1084 \stex_debug:nn{modules}{Closing~module~\prop_item:Nn \l_stex_current_module_prop { name }}
1085 }

```

(End definition for `\_stex_modules_end_module:.`)

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

```

1086 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1087 \NewDocumentEnvironment { @module } { 0{} m } {
1088   \par
1089   \_stex_modules_begin_module:nn{#1}{#2}
1090 } {
1091   \_stex_modules_end_module:
1092   \stex_if_smsmode:TF {
1093     \exp_args:Nx \stex_add_to_sms:n {
1094       \prop_gset_from_keyval:cn {
1095         c_stex_module_
1096         \prop_item:Nn \l_stex_current_module_prop { ns } ?
1097         \prop_item:Nn \l_stex_current_module_prop { name }
1098         _prop
1099       } {
1100         name      = \prop_item:cn { \l_tmpa_str } { name } ,
1101         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
1102         imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
1103         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
1104         content   = \prop_item:cn { \l_tmpa_str } { content } ,
1105         file      = \prop_item:cn { \l_tmpa_str } { file } ,
1106         lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
1107         sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
1108         meta      = \prop_item:cn { \l_tmpa_str } { meta }
1109       }
1110     }
1111   }{
1112     \end{stex_annotate_env}
1113   }
1114 }

```

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

```

1115 \cs_if_exist:NTF \thesection {
1116   \newcounter{module}[section]
1117 }{
1118   \newcounter{module}
1119 }
1120
1121 \bool_if:NT \c_stex_showmods_bool {
1122   \latexml_if:F { \RequirePackage{mdframed} }
1123 }
1124
1125 \cs_new_protected:Nn \stex_modules_heading: {
1126   \stepcounter{module}
1127   \par
1128   \bool_if:NT \c_stex_showmods_bool {

```

```

1129 \noindent{\textbf{Module} ~
1130 \cs_if_exist:NT \thesection {\thesection.}
1131 \themodule ~ [\l_stex_module_name_str]
1132 }
1133 \str_if_empty:NTF \l_stex_module_title_str {
1134 }{
1135 \quad(\l_stex_module_title_str)\hfill
1136 }\par
1137 }
1138 \edef\@currentlabel{Module~\thesection.\themodule~[\l_stex_module_name_str]}
1139 % TODO
1140 \stex_ref_new_doc_target:n \l_stex_module_name_str
1141 }

```

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

Finally:

```

1142 \NewDocumentEnvironment { module } { 0 } { m } {
1143 \bool_if:NT \c_stex_showmods_bool {
1144 \begin{mdframed}
1145 }
1146 \begin{@module}[#1]{#2}
1147 \stex_modules_heading:
1148 }{
1149 \end{@module}
1150 \bool_if:NT \c_stex_showmods_bool {
1151 \end{mdframed}
1152 }
1153 }

```

## 21.2 Invoking modules

`\STEXModule`  
`\stex_invoke_module:n`

```

1154 \NewDocumentCommand \STEXModule { m } {
1155 \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1156 \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1157 \tl_set:Nn \l_tmpa_tl {
1158 \msg_error:nnn{stex}{error/unknownmodule}{#1}
1159 }
1160 \seq_map_inline:Nn \l_stex_all_modules_seq {
1161 \str_set:Nn \l_tmpb_str { ##1 }
1162 \str_if_eq:eeT { \l_tmpa_str } {
1163 \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1164 } {
1165 \seq_map_break:n {
1166 \tl_set:Nn \l_tmpa_tl {
1167 \stex_invoke_module:n { ##1 }
1168 }
1169 }
1170 }
1171 }
1172 \l_tmpa_tl
1173 }
1174

```

```

1175 \cs_new_protected:Nn \stex_invoke_module:n {
1176   \stex_debug:nn{modules}{Invoking~module~#1}
1177   \peek_charcode_remove:NTF ! {
1178     \__stex_modules_invoke_uri:nN { #1 }
1179   } {
1180     \peek_charcode_remove:NTF ? {
1181       \__stex_modules_invoke_symbol:nn { #1 }
1182     } {
1183       \msg_error:nnn{stex}{error/syntax}{
1184         ?~or~!~expected~after~
1185         \c_backslash_str STEXModule{#1}
1186       }
1187     }
1188   }
1189 }
1190
1191 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1192   \str_set:Nn #2 { #1 }
1193 }
1194
1195 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1196   \stex_invoke_symbol:n{#1?#2}
1197 }

```

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

`\stex_activate_module:n`

```

1198 \cs_new_protected:Nn \stex_activate_module:n {
1199   \stex_debug:nn{modules}{Activating~module~#1}
1200   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1201     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1202     \prop_item:cn { c_stex_module_#1_prop } { content }
1203   }
1204 }

```

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

```

1205 </package>

```



## Chapter 22

# STEX -Module Inheritance Implementation

```
1206 <*package>
1207
1208 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1209
```

### 22.1 SMS Mode

```
1210 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1211 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1212 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1213 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1214
1215 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1216   \makeatletter
1217   \makeatother
1218   \ExplSyntaxOn
1219   \ExplSyntaxOff
1220 }
1221
1222 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1223   \symdef
1224   \importmodule
1225   \notation
1226   \symdecl
1227   \STEXexport
1228 }
1229
1230 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1231   \tl_to_str:n {
1232     module,
1233     @module
```

```

1234 }
1235 }

```

(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

```

```

1236 \bool_new:N \g__stex_smsmode_bool
1237 \bool_set_false:N \g__stex_smsmode_bool
1238 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1239   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1240 }

```

(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

```

```

1241 \bool_new:N \g__stex_smsmode_catcode_bool
1242 \bool_set_false:N \g__stex_smsmode_catcode_bool
1243 \prg_new_conditional:Nnn \__stex_smsmode_if_catcodes: { p, T, F, TF } {
1244   \bool_if:NTF \g__stex_smsmode_catcode_bool
1245   \prg_return_true: \prg_return_false:
1246 }

```

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

```

\stex_smsmode_set_codes:

```

```

1247 \cs_new_protected:Nn \stex_smsmode_set_codes: {
1248   \stex_if_smsmode:T {
1249     \__stex_smsmode_if_catcodes:F {
1250       \bool_gset_true:N \g__stex_smsmode_catcode_bool
1251       \exp_after:wN \char_gset_active_eq:NN
1252       \c_backslash_str \__stex_smsmode_cs:
1253       \tex_global:D \char_set_catcode_active:N \
1254       \tex_global:D \char_set_catcode_other:N $
1255       \tex_global:D \char_set_catcode_other:N ^
1256       \tex_global:D \char_set_catcode_other:N _
1257       \tex_global:D \char_set_catcode_other:N &
1258       \tex_global:D \char_set_catcode_other:N ##
1259     }
1260   }
1261 } \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.

```

1262 \cs_new_protected:Nn \__stex_smsmode_unset_codes: {
1263   \__stex_smsmode_if_catcodes:T {
1264     \bool_gset_false:N \g__stex_smsmode_catcode_bool
1265     \exp_after:wN \tex_global:D \exp_after:wN
1266     \char_set_catcode_escape:N \c_backslash_str
1267     \tex_global:D \char_set_catcode_math_toggle:N $
1268     \tex_global:D \char_set_catcode_math_superscript:N ^
1269     \tex_global:D \char_set_catcode_math_subscript:N _
1270     \tex_global:D \char_set_catcode_alignment:N &
1271     \tex_global:D \char_set_catcode_parameter:N ##
1272   }
1273 } \iffalse $ \fi % to make syntax highlighting work again

```

(End definition for `\_stex_smsmode_unset_codes:`.)

`\stex_in_smsmode:nn`

```

1274 \cs_new_protected:Nn \stex_in_smsmode:nn {
1275   \vbox_set:Nn \l_tmpa_box {
1276     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1277     \bool_gset_true:N \g__stex_smsmode_bool
1278     \stex_smsmode_set_codes:
1279     #2
1280     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1281     \stex_if_smsmode:F {
1282       \__stex_smsmode_unset_codes:
1283     }
1284   }
1285   \box_clear:N \l_tmpa_box
1286 }

```

(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:

```

1287 \cs_new_protected:Nn \_stex_smsmode_cs: {
1288   \str_clear:N \l_tmpa_str
1289   \peek_analysis_map_inline:n {
1290     % #1: token (one expansion)
1291     % #2: charcode
1292     % #3 catcode
1293     \token_if_eq_charcode:NNTF ##3 B {
1294       % token is a letter
1295       \exp_args:NNNo \str_put_right:Nn \l_tmpa_str { ##1 }
1296     } {
1297       \str_if_empty:NTF \l_tmpa_str {
1298         % we don't allow (or need) single non-letter CSs
1299         % for now
1300         \peek_analysis_map_break:
1301       }{
1302         \str_if_eq:onTF \l_tmpa_str { begin } {
1303           \peek_analysis_map_break:n {
1304             \exp_after:wN \_stex_smsmode_checkbegin:n ##1
1305           }
1306         } {
1307           \str_if_eq:onTF \l_tmpa_str { end } {
1308             \peek_analysis_map_break:n {
1309               \exp_after:wN \_stex_smsmode_checkend:n ##1
1310             }
1311           } {
1312             \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
1313             \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1314               \g_stex_smsmode_allowedmacros_tl
1315               { \use:c{\l_tmpa_str} } {
1316               \stex_debug:nn{modules}{Executing-1:~\l_tmpa_str}
1317               \peek_analysis_map_break:n {
1318                 \exp_after:wN \l_tmpa_tl ##1
1319               }

```

```

1320     } {
1321         \exp_args:NNo \exp_args:NNo \tl_if_in:NnTF
1322         \g_stex_smsmode_allowedmacros_escape_tl
1323         { \use:c{\l_tmpa_str} } {
1324             \__stex_smsmode_unset_codes:
1325             \stex_debug:nn{modules}{Executing~2:~\l_tmpa_str}
1326             % TODO \__stex_smsmode_rescan_cs:
1327             \int_compare:nNnTF {##2} = {92} {
1328                 \peek_analysis_map_break:n {
1329                     \__stex_smsmode_unset_codes:
1330                     \__stex_smsmode_rescan_cs:
1331                 }
1332             } {
1333                 \peek_analysis_map_break:n {
1334                     \exp_after:wN \l_tmpa_tl ##1
1335                 }
1336             }
1337         } {
1338             \int_compare:nNnTF {##2} = {92} {
1339                 \peek_analysis_map_break:n { \__stex_smsmode_cs: }
1340             } {
1341                 \peek_analysis_map_break:n { \exp_after:wN\relax ##1 }
1342             }
1343         }
1344     }
1345 }
1346 }
1347 }
1348 }
1349 }
1350 }

```

(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:

```

1351 \cs_new_protected:Nn \__stex_smsmode_rescan_cs: {
1352     \str_clear:N \l_tmpb_str
1353     \peek_analysis_map_inline:n {
1354         \token_if_eq_charcode:NNTF ##3 B {
1355             % token is a letter
1356             \exp_args:NNo \str_put_right:Nn \l_tmpb_str { ##1 }
1357         } {
1358             \peek_analysis_map_break:n {
1359                 \exp_after:wN \use:c \exp_after:wN {
1360                     \exp_after:wN \l_tmpa_str\exp_after:wN
1361                 } \use:c { \l_tmpb_str \exp_after:wN } ##1
1362             }
1363         }
1364     }
1365 }

```

(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.

```

1366 \cs_new_protected:Nn \__stex_smsmode_checkbegin:n {
1367   \str_set:Nn \l_tmpa_str { #1 }
1368   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1369     \__stex_smsmode_unset_codes:
1370     \begin{#1}
1371   }
1372 }
```

(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.

```

1373 \cs_new_protected:Nn \__stex_smsmode_checkend:n {
1374   \str_set:Nn \l_tmpa_str { #1 }
1375   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1376     \end{#1}
1377   }
1378 }
```

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

## 22.2 Inheritance

1379 `\stex_importmodule`

`\stex_import_module_uri:nn`

```

1380 \cs_new_protected:Nn \stex_import_module_uri:nn {
1381   \str_set:Nx \l__stex_importmodule_archive_str { #1 }
1382   \str_set:Nn \l__stex_importmodule_path_str { #2 }
1383   \str_if_empty:NT \l__stex_importmodule_archive_str {
1384     \prop_if_empty:NF \l_stex_current_repository_prop {
1385       \prop_get:NnN \l_stex_current_repository_prop { id } \l__stex_importmodule_archive_str
1386     }
1387   }
1388
1389   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l__stex_importmodule_path_str }
1390   \seq_pop_right:NN \l_tmpb_seq \l__stex_importmodule_name_str
1391   \str_set:Nx \l__stex_importmodule_path_str { \seq_use:Nn \l_tmpb_seq ? }
1392
1393   \str_if_empty:NTF \l__stex_importmodule_archive_str {
1394     \stex_modules_current_namespace:
1395     \str_if_empty:NF \l__stex_importmodule_path_str {
1396       \str_set:Nx \l_stex_module_ns_str {
1397         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1398       }
1399     }
1400   }{
1401     \stex_require_repository:n \l__stex_importmodule_archive_str
1402     \prop_get:cnN { c_stex_mathhub\l__stex_importmodule_archive_str _manifest_prop } { ns }
1403     \l_stex_module_ns_str
1404     \str_if_empty:NF \l__stex_importmodule_path_str {
1405       \str_set:Nx \l_stex_module_ns_str {
1406         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1407       }
1408     }
1409   }
```

```

1408     }
1409   }
1410 }

```

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

```

\l_stex_importmodule_name_str Store the return values of \stex_import_module_uri:nn.
\l_stex_importmodule_archive_str 1411 \str_new:N \l__stex_importmodule_name_str
\l_stex_importmodule_path_str 1412 \str_new:N \l__stex_importmodule_archive_str
\l_stex_importmodule_file_str 1413 \str_new:N \l__stex_importmodule_path_str
1414 \str_new:N \g__stex_importmodule_file_str

```

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

```

\stex_import_require_module:nnnn {<ns>} {<archive-ID>} {<path>} {<name>}
1415 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1416   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1417
1418     % archive
1419     \str_set:Nx \l_tmpa_str { #2 }
1420     \str_if_empty:NTF \l_tmpa_str {
1421       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1422     } {
1423       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1424       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1425       \seq_put_right:Nn \l_tmpa_seq { source }
1426     }
1427
1428     % path
1429     \str_set:Nx \l_tmpb_str { #3 }
1430     \str_if_empty:NTF \l_tmpb_str {
1431       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1432
1433       \ltx@ifpackageloaded{babel} {
1434         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1435           { \language } \l_tmpb_str {
1436           \msg_error:nnn{stex}{error/unknownlanguage}{\language}
1437         }
1438       } {
1439         \str_clear:N \l_tmpb_str
1440       }
1441
1442       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1443       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1444         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1445       }{
1446         \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1447         \IfFileExists{ \l_tmpa_str.tex }{
1448           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1449         }{
1450           % try english as default
1451           \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1452           \IfFileExists{ \l_tmpa_str.en.tex }{
1453             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }

```

```

1454     }{
1455         \msg_error:nnn{stex}{error/unknownmodule}{#1?#4}
1456     }
1457 }
1458 }
1459
1460 } {
1461     \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1462     \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1463
1464     \ltx@ifpackageloaded{babel} {
1465         \exp_args:NnX \prop_get:NnNF \c_stex_language_abbrevs_prop
1466             { \language } \l_tmpb_str {
1467             \msg_error:nnn{stex}{error/unknownlanguage}{\language}
1468         }
1469     } {
1470         \str_clear:N \l_tmpb_str
1471     }
1472
1473     \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1474
1475     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1476     \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1477         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1478     }{
1479         \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1480         \IfFileExists{ \l_tmpa_str/#4.tex }{
1481             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1482         }{
1483             % try english as default
1484             \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1485             \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1486                 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1487             }{
1488                 \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1489                 \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1490                     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1491                 }{
1492                     \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1493                     \IfFileExists{ \l_tmpa_str.tex }{
1494                         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1495                     }{
1496                         % try english as default
1497                         \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1498                         \IfFileExists{ \l_tmpa_str.en.tex }{
1499                             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1500                         }{
1501                             \msg_error:nnn{stex}{error/unknownmodule}{#1?#4}
1502                         }
1503                     }
1504                 }
1505             }
1506         }
1507     }

```

```

1508     }
1509
1510     \seq_set_eq:NN \l_tmpa_seq \g_stex_modules_in_file_seq
1511     \seq_clear:N \g_stex_modules_in_file_seq
1512     % \exp_args:Nnx \use:nn {
1513         \exp_args:No \stex_in_smsmode:nn { \g__stex_importmodule_file_str } {
1514             \seq_clear:N \l_stex_all_modules_seq
1515             \prop_clear:N \l_stex_current_module_prop
1516             \str_set:Nx \l_tmpb_str { #2 }
1517             \str_if_empty:NF \l_tmpb_str {
1518                 \stex_set_current_repository:n { #2 }
1519             }
1520             \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1521             \input { \g__stex_importmodule_file_str }
1522         }
1523     % }{
1524
1525     % }
1526     \prop_gput:Noo \g_stex_module_files_prop
1527     \g__stex_importmodule_file_str \g_stex_modules_in_file_seq
1528     \seq_set_eq:NN \g_stex_modules_in_file_seq \l_tmpa_seq
1529
1530     \stex_if_module_exists:nF { #1 ? #4 } {
1531         \msg_error:nnn{stex}{error/unknownmodule}{
1532             #1?#4~(in~file~\g__stex_importmodule_file_str)
1533         }
1534     }
1535 }
1536 \stex_activate_module:n { #1 ? #4 }
1537 }

```

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

## `\importmodule`

```

1538 \NewDocumentCommand \importmodule { O{} m } {
1539     \stex_import_module_uri:nn { #1 } { #2 }
1540     \stex_debug:nn{modules}{Importing~module:~
1541         \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1542     }
1543     \stex_if_smsmode:F {
1544         \stex_import_require_module:nnnn
1545         { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1546         { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1547         \stex_annotate_invisible:nnn
1548         {import} { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } {}
1549     }
1550     \exp_args:Nx \stex_add_to_current_module:n {
1551         \stex_import_require_module:nnnn
1552         { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1553         { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1554     }
1555     \exp_args:Nx \stex_add_import_to_current_module:n {
1556         \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1557     }

```



```

1558 \stex_smsmode_set_codes:
1559 }
1560 \stex_deactivate_macro:Nn \importmodule {module-environments}

```

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

## **\usemodule**

```

1561 \NewDocumentCommand \usemodule { 0{} m } {
1562   \stex_if_smsmode:F {
1563     \stex_import_module_uri:nn { #1 } { #2 }
1564     \stex_import_require_module:nnnn
1565     { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1566     { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1567     \stex_annotate_invisible:nnn
1568     {usemodule} {\l_stex_module_ns_str ? \l__stex_importmodule_name_str} {}
1569   }
1570   \stex_smsmode_set_codes:
1571 }

```

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

```

1572 \endpackage

```

## Chapter 23

# STEX -Symbols Implementation

```
1573 <*package>
1574
1575 %%%%%%%%%%%%% symbols.dtx %%%%%%%%%%%%%
1576
Warnings and error messages
1577
```

### 23.1 Symbol Declarations

```
1578 <@@=stex_symdecl>

\l_stex_all_symbols_seq Stores all available symbols
1579 \seq_new:N \l_stex_all_symbols_seq

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

\STEXsymbol

1580 \NewDocumentCommand \STEXsymbol { m } {
1581   \stex_get_symbol:n { #1 }
1582   \exp_args:No
1583   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1584 }

(End definition for \STEXsymbol. This function is documented on page 27.)
symdecl arguments:

1585 \keys_define:nn { stex / symdecl } {
1586   name      .str_set:N = \l_stex_symdecl_name_str ,
1587   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1588   args      .str_set:N = \l_stex_symdecl_args_str ,
1589   type      .tl_set:N   = \l_stex_symdecl_type_tl ,
1590   align     .str_set:N   = \l_stex_symdecl_align_str , % TODO(?)
1591   gfc       .str_set:N   = \l_stex_symdecl_gfc_str , % TODO(?)
1592   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
1593   def       .tl_set:N   = \l_stex_symdecl_definiens_tl
1594 }
```

```

1595
1596 \bool_new:N \l_stex_symdecl_make_macro_bool
1597
1598 \cs_new_protected:Nn \__stex_symdecl_args:n {
1599   \str_clear:N \l_stex_symdecl_name_str
1600   \str_clear:N \l_stex_symdecl_args_str
1601   \bool_set_false:N \l_stex_symdecl_local_bool
1602   \tl_clear:N \l_stex_symdecl_type_tl
1603   \tl_clear:N \l_stex_symdecl_definiens_tl
1604
1605   \keys_set:nn { stex / symdecl } { #1 }
1606 }

```

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

```

1607
1608 \NewDocumentCommand \symdecl { s O{} m } {
1609   \__stex_symdecl_args:n { #2 }
1610   \IfBooleanTF #1 {
1611     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1612   } {
1613     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1614   }
1615   \stex_symdecl_do:n { #3 }
1616   \stex_smsmode_set_codes:
1617 }
1618 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

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

```

1619 \cs_new_protected:Nn \stex_symdecl_do:n {
1620   \stex_if_in_module:F {
1621     % TODO throw error? some default namespace?
1622   }
1623
1624   \str_if_empty:NT \l_stex_symdecl_name_str {
1625     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1626   }
1627
1628   \prop_if_exist:cT { g_stex_symdecl_
1629     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1630     \prop_item:Nn \l_stex_current_module_prop {name} ?
1631     \l_stex_symdecl_name_str
1632     _prop
1633   }{
1634     % TODO throw error (beware of circular dependencies)
1635   }
1636
1637   \prop_clear:N \l_tmpa_prop
1638   \prop_put:Nnx \l_tmpa_prop { module } {
1639     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1640     \prop_item:Nn \l_stex_current_module_prop {name}
1641   }

```

```

1642 \seq_clear:N \l_tmpa_seq
1643 \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
1644 \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1645 \prop_put:Nno \l_tmpa_prop { local } \l_stex_symdecl_local_bool
1646 \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1647
1648 \exp_args:No \stex_add_constant_to_current_module:n {
1649   \l_stex_symdecl_name_str
1650 }
1651
1652 % arity/args
1653 \int_zero:N \l_tmpb_int
1654
1655 \bool_set_true:N \l_tmpa_bool
1656 \str_map_inline:Nn \l_stex_symdecl_args_str {
1657   \token_case_meaning:NnF ##1 {
1658     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1659     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1660     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1661     {\tl_to_str:n a} {
1662       \bool_set_false:N \l_tmpa_bool
1663       \int_incr:N \l_tmpb_int
1664     }
1665     {\tl_to_str:n B} {
1666       \bool_set_false:N \l_tmpa_bool
1667       \int_incr:N \l_tmpb_int
1668     }
1669   }{
1670     \msg_set:nnn{stex}{error/wrongargs}{
1671       args~value~in~symbol~declaration~for~
1672       \prop_item:Nn \l_stex_current_module_prop {ns} ?
1673       \prop_item:Nn \l_stex_current_module_prop {name} ?
1674       \l_stex_symdecl_name_str ~
1675       needs~to~be~
1676       i,~a,~b~or~B,~but~##1~given
1677     }
1678     \msg_error:nn{stex}{error/wrongargs}
1679   }
1680 }
1681 \bool_if:NTF \l_tmpa_bool {
1682   % possibly numeric
1683   \str_if_empty:NTF \l_stex_symdecl_args_str {
1684     \prop_put:Nnn \l_tmpa_prop { args } {}
1685     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1686   }{
1687     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1688     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1689     \str_clear:N \l_tmpa_str
1690     \int_step_inline:nn \l_tmpa_int {
1691       \str_put_right:Nn \l_tmpa_str i
1692     }
1693     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1694   }
1695 } {

```

```

1696     \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1697     \prop_put:Nnx \l_tmpa_prop { arity }
1698       { \str_count:N \l_stex_symdecl_args_str }
1699   }
1700   \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
1701
1702
1703   % semantic macro
1704
1705   \bool_if:NT \l_stex_symdecl_make_macro_bool {
1706     \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1707       \prop_item:Nn \l_tmpa_prop { module } ?
1708       \prop_item:Nn \l_tmpa_prop { name }
1709     } }
1710
1711     \bool_if:NF \l_stex_symdecl_local_bool {
1712       \exp_args:Nx \stex_add_to_current_module:n {
1713         \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1714           \prop_item:Nn \l_tmpa_prop { module } ?
1715           \prop_item:Nn \l_tmpa_prop { name }
1716         } }
1717       }
1718     }
1719   }
1720
1721   % add to all symbols
1722
1723   \bool_if:NF \l_stex_symdecl_local_bool {
1724     \exp_args:Nx \stex_add_to_current_module:n {
1725       \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1726         \prop_item:Nn \l_tmpa_prop { module } ?
1727         \prop_item:Nn \l_tmpa_prop { name }
1728       }
1729     }
1730   }
1731
1732   \stex_debug:nn{symbols}{New~symbol:~
1733     \prop_item:Nn \l_tmpa_prop { module } ?
1734     \prop_item:Nn \l_tmpa_prop { name } ^^J
1735     Type:~\exp_not:o { \l_stex_symdecl_type_tl } ^^J
1736     Args:~\prop_item:Nn \l_tmpa_prop { args }
1737   }
1738
1739   % circular dependencies require this:
1740
1741   \prop_if_exist:cF {
1742     g_stex_symdecl_
1743     \prop_item:Nn \l_tmpa_prop { module } ?
1744     \prop_item:Nn \l_tmpa_prop { name }
1745     _prop
1746   } {
1747     \prop_gset_eq:cN {
1748       g_stex_symdecl_
1749       \prop_item:Nn \l_tmpa_prop { module } ?

```

```

1750     \prop_item:Nn \l_tmpa_prop { name }
1751     _prop
1752   } \l_tmpa_prop
1753 }
1754
1755 \stex_if_smsmode:TF {
1756   \bool_if:NF \l_stex_symdecl_local_bool {
1757     \exp_args:Nx \stex_add_to_sms:n {
1758       \prop_gset_from_keyval:cn {
1759         g_stex_symdecl_
1760         \prop_item:Nn \l_tmpa_prop { module } ?
1761         \prop_item:Nn \l_tmpa_prop { name }
1762         _prop
1763       } {
1764         name      = \prop_item:Nn \l_tmpa_prop { name }      ,
1765         module    = \prop_item:Nn \l_tmpa_prop { module }    ,
1766         notations = \prop_item:Nn \l_tmpa_prop { notations } ,
1767         local     = \prop_item:Nn \l_tmpa_prop { local }     ,
1768         type      = \prop_item:Nn \l_tmpa_prop { type }      ,
1769         args      = \prop_item:Nn \l_tmpa_prop { args }      ,
1770         arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
1771         assocs    = \prop_item:Nn \l_tmpa_prop { assocs }
1772       }
1773       \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1774         \prop_item:Nn \l_tmpa_prop { module } ?
1775         \prop_item:Nn \l_tmpa_prop { name }
1776       }
1777     }
1778   }
1779 }{
1780   \exp_args:NNx \seq_put_right:Nn \l_stex_all_symbols_seq {
1781     \prop_item:Nn \l_tmpa_prop { module } ?
1782     \prop_item:Nn \l_tmpa_prop { name }
1783   }
1784   \stex_if_do_html:T {
1785     \stex_annotate_invisible:nnn {symdecl} {
1786       \prop_item:Nn \l_tmpa_prop { module } ?
1787       \prop_item:Nn \l_tmpa_prop { name }
1788     } {
1789       \stex_annotate_invisible:nnn{type}{}{\l_stex_symdecl_type_tl$}
1790       \stex_annotate_invisible:nnn{args}{}{
1791         \prop_item:Nn \l_tmpa_prop { args }
1792       }
1793       \stex_annotate_invisible:nnn{macroname}{}{#1}
1794       \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
1795         \stex_annotate_invisible:nnn{definiens}{}
1796         {\l_stex_symdecl_definiens_tl$}
1797       }
1798     }
1799   }
1800 }
1801 }

```

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

`\stex_get_symbol:n`

```
1802 \str_new:N \l_stex_get_symbol_uri_str
1803
1804 \cs_new_protected:Nn \stex_get_symbol:n {
1805   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
1806     \__stex_symdecl_get_symbol_from_cs:n { #1 }
1807   }{
1808     % argument is a string
1809     % is it a command name?
1810     \cs_if_exist:cTF { #1 }{
1811       \cs_set_eq:Nc \l_tmpa_tl { #1 }
1812       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
1813       \str_if_empty:NNTF \l_tmpa_str {
1814         \exp_args:Nx \cs_if_eq:NNTF {
1815           \tl_head:N \l_tmpa_tl
1816         } \stex_invoke_symbol:n {
1817           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
1818         }{
1819           \__stex_symdecl_get_symbol_from_string:n { #1 }
1820         }
1821       } {
1822         \__stex_symdecl_get_symbol_from_string:n { #1 }
1823       }
1824     }{
1825       % argument is not a command name
1826       \__stex_symdecl_get_symbol_from_string:n { #1 }
1827       % \l_stex_all_symbols_seq
1828     }
1829   }
1830 }
1831
1832 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
1833   \str_set:Nn \l_tmpa_str { #1 }
1834   \bool_set_false:N \l_tmpa_bool
1835   \stex_if_in_module:T {
1836     \prop_get:NnN \l_stex_current_module_prop
1837     { constants } \l_tmpa_seq
1838     \exp_args:NNo \seq_if_in:NnT \l_tmpa_seq { \l_tmpa_str } {
1839       \bool_set_true:N \l_tmpa_bool
1840       \str_set:Nx \l_stex_get_symbol_uri_str {
1841         \prop_item:Nn \l_stex_current_module_prop { ns } ?
1842         \prop_item:Nn \l_stex_current_module_prop { name } ? #1
1843       }
1844     }
1845   }
1846   \bool_if:NF \l_tmpa_bool {
1847     \tl_set:Nn \l_tmpa_tl {
1848       \msg_set:nnn{stex}{error/unknownsymbol}{
1849         No~symbol~#1~found!
1850       }
1851     }
1852     \msg_error:nn{stex}{error/unknownsymbol}
1853   }
1854   \str_set:Nn \l_tmpa_str { #1 }
1855   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
```

```

1855 \seq_map_inline:Nn \l_stex_all_symbols_seq {
1856   \str_set:Nn \l_tmpb_str { ##1 }
1857   \str_if_eq:eeT { \l_tmpa_str } {
1858     \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1859   } {
1860     \seq_map_break:n {
1861       \tl_set:Nn \l_tmpa_tl {
1862         \str_set:Nn \l_stex_get_symbol_uri_str {
1863           ##1
1864         }
1865       }
1866     }
1867   }
1868 }
1869 \l_tmpa_tl
1870 }
1871 }
1872
1873 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
1874   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
1875   { \tl_tail:N \l_tmpa_tl }
1876   \tl_if_single:NTF \l_tmpa_tl {
1877     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
1878       \exp_after:wN \str_set:Nn \exp_after:wN
1879       \l_stex_get_symbol_uri_str \l_tmpa_tl
1880     }{
1881       % TODO
1882       % tail is not a single group
1883     }
1884   }{
1885     % TODO
1886     % tail is not a single group
1887   }
1888 }

```

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

## 23.2 Notations

```

1889 <@@=stex_notation>
1890 notation arguments:
1891 \keys_define:nn { stex / notation } {
1892   lang .tl_set_x:N = \l__stex_notation_lang_str ,
1893   variant .tl_set_x:N = \l__stex_notation_variant_str ,
1894   prec .tl_set_x:N = \l__stex_notation_prec_str ,
1895   op .tl_set:N = \l__stex_notation_op_tl ,
1896   unknown .code:n = \str_set:Nx
1897     \l__stex_notation_variant_str \l_keys_key_str
1898 }
1899
1900 \cs_new_protected:Nn \__stex_notation_args:n {
1901   \str_clear:N \l__stex_notation_lang_str
1902   \str_clear:N \l__stex_notation_variant_str

```



```

1902 \str_clear:N \l__stex_notation_prec_str
1903 \tl_clear:N \l__stex_notation_op_tl
1904
1905 \keys_set:nn { stex / notation } { #1 }
1906 }

```

## **\notation**

```

1907 \NewDocumentCommand \notation { 0{ } m } {
1908   \__stex_notation_args:n { #1 }
1909   \tl_clear:N \l_stex_symdecl_definiens_tl
1910   \stex_get_symbol:n { #2 }
1911   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
1912 }
1913 \stex_deactivate_macro:Nn \notation {module~environments}

```

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

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

```

1914 \cs_new_protected:Nn \stex_notation_do:nn {
1915   \prop_set_eq:Nc \l_tmpa_prop {
1916     g_stex_symdecl_ #1 _prop
1917   }
1918
1919   \prop_clear:N \l_tmpb_prop
1920   \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
1921   \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
1922   \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
1923
1924   % precedences
1925   \seq_clear:N \l_tmpb_seq
1926   \exp_args:NNno
1927   \str_if_empty:NTF \l__stex_notation_prec_str {
1928     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1929     \int_compare:nNnTF \l_tmpa_str = 0 {
1930       \exp_args:NNnx
1931       \prop_put:Nno \l_tmpb_prop { opprec }
1932       { \neginfprec }
1933     }{
1934       \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
1935     }
1936   } {
1937     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
1938       \exp_args:NNnx
1939       \prop_put:Nno \l_tmpb_prop { opprec }
1940       { \neginfprec }
1941       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1942       \int_step_inline:nn { \l_tmpa_str } {
1943         \exp_args:NNx
1944         \seq_put_right:Nn \l_tmpb_seq { \infprec }
1945       }
1946     }{
1947       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
1948       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
1949         \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
1950         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {

```

```

1951         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
1952         \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
1953         \seq_map_inline:Nn \l_tmpa_seq {
1954             \seq_put_right:Nn \l_tmpb_seq { ##1 }
1955         }
1956     }
1957     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1958 }{
1959     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1960     \int_compare:nNnTF \l_tmpa_str = 0 {
1961         \exp_args:NNnx
1962         \prop_put:Nno \l_tmpb_prop { opprec }
1963         { \infprec }
1964     }{
1965         \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
1966     }
1967 }
1968 }
1969 }
1970
1971 \seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
1972 \int_step_inline:nn { \l_tmpa_str } {
1973     \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
1974         \exp_args:NNx
1975         \seq_put_right:Nn \l_tmpb_seq {
1976             \prop_item:Nn \l_tmpb_prop { opprec }
1977         }
1978     }
1979 }
1980
1981 \prop_put:Nno \l_tmpb_prop { argprec } \l_tmpb_seq
1982 \tl_clear:N \l_tmpa_tl
1983
1984 \int_compare:nNnTF \l_tmpa_str = 0 {
1985     \exp_args:NNe
1986     \cs_set:Npn \l__stex_notation_macrocode_cs {
1987         \_stex_term_math_oms:nnnn { #1 }
1988         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1989         { \prop_item:Nn \l_tmpb_prop { opprec } }
1990         { \exp_not:n { #2 } }
1991     }
1992     \__stex_notation_final:
1993 }{
1994     \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
1995     \str_if_in:NnTF \l_tmpb_str b {
1996         \exp_args:Nne \use:nn
1997         {
1998             \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1999             \cs_set:Npn \l_tmpa_str { {
2000                 \_stex_term_math_omb:nnnn { #1 }
2001                 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2002                 { \prop_item:Nn \l_tmpb_prop { opprec } }
2003                 { \exp_not:n { #2 } }
2004             }
2005         }
2006     }

```

```

2005   }{
2006     \str_if_in:NnTF \l_tmpb_str B {
2007       \exp_args:Nne \use:nn
2008       {
2009         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2010         \cs_set:Npn \l_tmpa_str } { {
2011           \_stex_term_math_omb:nnnn { #1 }
2012           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2013           { \prop_item:Nn \l_tmpb_prop { opprec } }
2014           { \exp_not:n { #2 } }
2015         } }
2016       }{
2017         \exp_args:Nne \use:nn
2018         {
2019           \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2020           \cs_set:Npn \l_tmpa_str } { {
2021             \_stex_term_math_oma:nnnn { #1 }
2022             { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2023             { \prop_item:Nn \l_tmpb_prop { opprec } }
2024             { \exp_not:n { #2 } }
2025           } }
2026         }
2027       }
2028
2029       \int_zero:N \l_tmpa_int
2030       \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2031       \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2032       \__stex_notation_arguments:
2033     }
2034   }

```

(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

```

2035 \cs_new_protected:Nn \__stex_notation_arguments: {
2036   \int_incr:N \l_tmpa_int
2037   \str_if_empty:NnTF \l_tmpa_str {
2038     \__stex_notation_final:
2039   }{
2040     \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2041     \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2042     \str_if_eq:NnTF \l_tmpb_str a {
2043       \__stex_notation_argument_assoc:n
2044     }{
2045       \str_if_eq:NnTF \l_tmpb_str B {
2046         \__stex_notation_argument_assoc:n
2047       }{
2048         \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2049         \tl_put_right:Nx \l_tmpa_tl {
2050           { \_stex_term_math_arg:nnn
2051             { \int_use:N \l_tmpa_int }
2052             { \l_tmpb_str }
2053             { ####\int_use:N \l_tmpa_int }
2054           }

```

```

2055     }
2056     \__stex_notation_arguments:
2057   }
2058 }
2059 }
2060 }

```

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

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

```

2061 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
2062   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2063   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
2064   \tl_put_right:Nx \l_tmpa_tl {
2065     { \stex_term_math_assoc_arg:nnnn
2066       { \int_use:N \l_tmpa_int }
2067       { \l_tmpb_str }
2068       \exp_args:No \exp_not:n
2069       {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
2070       { ####\int_use:N \l_tmpa_int }
2071     }
2072   }
2073   \__stex_notation_arguments:
2074 }

```

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

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

```

2075 \cs_new_protected:Nn \__stex_notation_final: {
2076   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
2077   \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
2078   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2079   \exp_args:Nne \use:nn
2080   {
2081     \cs_generate_from_arg_count:cNnn {
2082       stex_notation_ \l_tmpa_str \c_hash_str
2083       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2084       _cs
2085     }
2086     \cs_gset:Npn \l_tmpb_str { { {
2087       \exp_after:wN \exp_after:wN \exp_after:wN
2088       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2089       { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2090     } } }
2091
2092     \tl_if_empty:NF \l__stex_notation_op_tl {
2093       \cs_gset:cpx {
2094         stex_op_notation_ \l_tmpa_str \c_hash_str
2095         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2096         _cs
2097       } {
2098         \stex_term_oms:nnn {
2099           \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2100           \l__stex_notation_lang_str

```

```

2101     }{
2102         \l_tmpa_str
2103     }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2104 }
2105 }
2106
2107
2108
2109 \stex_debug:nn{symbols}{
2110     Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2111     ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
2112     Operator~precedence:~
2113     \prop_item:Nn \l_tmpb_prop { opprec }^^J
2114     Argument~precedences:~
2115     \seq_use:Nn \l_tmpa_seq {,~}^^J
2116     Notation: \cs_meaning:c {
2117         stex_notation_ \l_tmpa_str \c_hash_str
2118         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2119         _cs
2120     }
2121 }
2122
2123 \prop_gset_eq:cN {
2124     g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2125     \c_hash_str \l__stex_notation_lang_str _prop
2126 } \l_tmpb_prop
2127
2128 \exp_args:Nx
2129 \stex_add_to_current_module:n {
2130     \prop_get:cnN {
2131         g_stex_symdecl_
2132         \prop_item:Nn \l_tmpb_prop { symbol }
2133         _prop
2134     } { notations } \exp_not:N \l_tmpa_seq
2135     \seq_put_right:Nn \exp_not:N \l_tmpa_seq {
2136         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2137     }
2138     \prop_put:cno {
2139         g_stex_symdecl_
2140         \prop_item:Nn \l_tmpb_prop { symbol }
2141         _prop
2142     } { notations } \exp_not:N \l_tmpa_seq
2143 }
2144
2145 \stex_if_smsmode:TF {
2146     \stex_smsmode_set_codes:
2147     \exp_args:Nx \stex_add_to_sms:n {
2148         \prop_gset_from_keyval:cn {
2149             g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2150             \c_hash_str \l__stex_notation_lang_str _prop
2151         } {
2152             symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2153             language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2154             variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,

```

```

2155         opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2156         argprec     = \prop_item:Nn \l_tmpb_prop { argprec }     ,
2157     }
2158 }
2159 }{
2160   \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
2161   \seq_put_right:Nx \l_tmpa_seq {
2162     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2163   }
2164   \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
2165   \prop_set_eq:cN {
2166     g_stex_symdecl_ \l_tmpa_str _prop
2167   } \l_tmpa_prop
2168
2169   % HTML annotations
2170   \stex_if_do_html:T {
2171     \stex_annotate_invisible:nnn { notation }
2172     { \prop_item:Nn \l_tmpb_prop { symbol } } {
2173       \stex_annotate_invisible:nnn { notationfragment }
2174       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
2175       \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2176       \stex_annotate_invisible:nnn { precedence }
2177       { \prop_item:Nn \l_tmpb_prop { opprec } ;
2178         \seq_use:Nn \l_tmpa_seq { x }
2179       }{}
2180
2181       \int_zero:N \l_tmpa_int
2182       \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2183       \tl_clear:N \l_tmpa_tl
2184       \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{}{
2185         \int_incr:N \l_tmpa_int
2186         \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2187         \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2188         \str_if_eq:VnTF \l_tmpb_str a {
2189           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2190             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2191             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2192           } }
2193         }{
2194           \str_if_eq:VnTF \l_tmpb_str B {
2195             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2196               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2197               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2198             } }
2199           }{
2200             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2201               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2202             } }
2203           }
2204         }
2205       }
2206       \stex_annotate_invisible:nnn { notationcomp }{}{
2207         $ \exp_args:Nno \use:nn { \use:c {
2208           stex_notation_ \prop_item:Nn \l_tmpb_prop { symbol }

```

```

2209         \c_hash_str \l__stex_notation_variant_str
2210         \c_hash_str \l__stex_notation_lang_str _cs
2211     } } { \l_tmpa_tl } $
2212   }
2213 }
2214 }
2215 }
2216 }

```

(End definition for \\_stex\_notation\_final:.)

**\symdef**

```

2217 \keys_define:nn { stex / symdef } {
2218   name .str_set_x:N = \l_stex_symdecl_name_str ,
2219   local .bool_set:N = \l_stex_symdecl_local_bool ,
2220   args .str_set_x:N = \l_stex_symdecl_args_str ,
2221   type .tl_set:N = \l_stex_symdecl_type_tl ,
2222   def .tl_set:N = \l_stex_symdecl_definiens_tl ,
2223   op .tl_set:N = \l__stex_notation_op_tl ,
2224   lang .str_set_x:N = \l__stex_notation_lang_str ,
2225   variant .str_set_x:N = \l__stex_notation_variant_str ,
2226   prec .str_set_x:N = \l__stex_notation_prec_str ,
2227   unknown .code:n = \str_set:Nx
2228     \l__stex_notation_variant_str \l_keys_key_str
2229 }
2230
2231 \cs_new_protected:Nn \_stex_notation_symdef_args:n {
2232   \str_clear:N \l_stex_symdecl_name_str
2233   \str_clear:N \l_stex_symdecl_args_str
2234   \bool_set_false:N \l_stex_symdecl_local_bool
2235   \tl_clear:N \l_stex_symdecl_type_tl
2236   \tl_clear:N \l_stex_symdecl_definiens_tl
2237   \str_clear:N \l__stex_notation_lang_str
2238   \str_clear:N \l__stex_notation_variant_str
2239   \str_clear:N \l__stex_notation_prec_str
2240   \tl_clear:N \l__stex_notation_op_tl
2241
2242   \keys_set:nn { stex / symdef } { #1 }
2243 }
2244
2245 \NewDocumentCommand \symdef { 0{} m } {
2246   \_stex_notation_symdef_args:n { #1 }
2247   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2248   \stex_symdecl_do:n { #2 }
2249   \exp_args:Nx \stex_notation_do:nn {
2250     \prop_item:Nn \l_tmpa_prop { module } ?
2251     \prop_item:Nn \l_tmpa_prop { name }
2252   }
2253 }
2254 \stex_deactivate_macro:Nn \symdef {module~environments}

```

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

```

2255 \endpackage

```

## Chapter 24

# STEX -Terms Implementation

```
2256 <*package>
2257
2258 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2259
2260 <@@=stex_terms>
2261
2262   Warnings and error messages
2263   \msg_new:nnn{stex}{error/nonotation}{
2264     Symbol~#1~invoked,~but~has~no~notation#2!
2265   }
2266   \msg_new:nnn{stex}{error/notationarg}{
2267     Error~in~parsing~notation~#1
2268   }
2269
```

### 24.1 Symbol Invocations

Arguments:

```
2268 \keys_define:nn { stex / terms } {
2269   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2270   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2271   unknown .code:n = \str_set:Nx
2272     \l__stex_terms_variant_str \l_keys_key_str
2273 }
2274
2275 \cs_new_protected:Nn \__stex_terms_args:n {
2276   \str_clear:N \l__stex_terms_lang_str
2277   \str_clear:N \l__stex_terms_variant_str
2278   \str_clear:N \l__stex_terms_prec_str
2279   \tl_clear:N \l__stex_terms_op_tl
2280
2281   \keys_set:nn { stex / terms } { #1 }
2282 }
```

`\stex_invoke_symbol:n` Invokes a semantic macro



```

2283 \cs_new_protected:Nn \stex_invoke_symbol:n {
2284   \if_mode_math:
2285     \exp_after:wN \__stex_terms_invoke_math:n
2286   \else:
2287     \exp_after:wN \__stex_terms_invoke_text:n
2288   \fi: { #1 }
2289 }

```

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

`\__stex_terms_invoke_math:n`

```

2290 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2291   \peek_charcode_remove:NTF ! {
2292     \peek_charcode:NTF [ {
2293       \__stex_terms_invoke_op:nw { #1 }
2294     }{
2295       \__stex_terms_invoke_op:nw { #1 } []
2296     }
2297   }{
2298     \peek_charcode_remove:NTF * {
2299       \__stex_terms_invoke_text:n { #1 }
2300     }{
2301       \peek_charcode:NTF [ {
2302         \__stex_terms_invoke_math:nw { #1 }
2303       }{
2304         \__stex_terms_invoke_math:nw { #1 } []
2305       }
2306     }
2307   }
2308 }

```

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

`\__stex_terms_invoke_op:nw`

```

2309 \cs_new_protected:Npn \__stex_terms_invoke_op:nw #1 [#2] {
2310   \__stex_terms_args:n { #2 }
2311   \cs_if_exist:cTF {
2312     stex_op_notation_ #1 \c_hash_str
2313     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2314   }{
2315     \csname stex_op_notation_ #1 \c_hash_str
2316     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2317   \endcsname
2318   }{
2319     % TODO throw error
2320   }
2321 }

```

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

`\__stex_terms_invoke_math:nw`

```

2322 \cs_new_protected:Npn \__stex_terms_invoke_math:nw #1 [#2] {
2323   \__stex_terms_args:n { #2 }
2324   \prop_set_eq:Nc \l_tmpa_prop {
2325     g_stex_symdecl_ #1 _prop

```

```

2326 }
2327 \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
2328 \seq_if_empty:NTF \l_tmpa_seq {
2329   \msg_error:nnnn{stex}{error/nonotation}{#1}{s}
2330 } {
2331   \seq_if_in:NxTF \l_tmpa_seq
2332     { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
2333     \use:c{
2334       stex_notation_ #1 \c_hash_str
2335       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2336     }_cs
2337   }
2338 }{
2339   \str_if_empty:NTF \l__stex_terms_variant_str {
2340     \str_if_empty:NTF \l__stex_terms_lang_str {
2341       \seq_get_left:NN \l_tmpa_seq \l_tmpa_str
2342       \use:c{
2343         stex_notation_ #1 \c_hash_str \l_tmpa_str
2344       }_cs
2345     }
2346   }{
2347     \msg_error:nn{stex}{error/nonotation}{#1}{
2348       ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2349     }
2350   }
2351 }{
2352   \msg_error:nn{stex}{error/nonotation}{#1}{
2353     ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2354   }
2355 }
2356 }
2357 }
2358 }

```

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

`\__stex_terms_invoke_text:n`

```

2359 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
2360   \peek_charcode_remove:NTF ! {
2361     \stex_term_custom:nn { #1 } { }
2362   }{
2363     \prop_set_eq:Nc \l_tmpa_prop {
2364       g_stex_symdecl_ #1 _prop
2365     }
2366     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2367     \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2368   }
2369 }

```

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

## 24.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
2370 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2371 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2372 \int_new:N \l__stex_terms_downprec
2373 \int_set_eq:NN \l__stex_terms_downprec \infprec

(End definition for \infprec, \neginfprec, and \l__stex_terms_downprec. These variables are docu-
mented on page 28.)

```

Bracketing:

```

\l__stex_terms_left_bracket_str
\l__stex_terms_right_bracket_str
2374 \tl_set:Nn \l__stex_terms_left_bracket_str (
2375 \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

```

2376 \cs_new_protected:Nn \_stex_terms_maybe_brackets:nn {
2377   \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
2378     \bool_if:NTF \l_stex_inarray_bool { #2 }{
2379       \dobrackets { #2 }
2380     }
2381   }{ #2 }
2382 }

(End definition for \_stex_terms_maybe_brackets:nn.)

```

\dobrackets

```

2383 %\RequirePackage{scalerel}
2384 \cs_new_protected:Npn \dobrackets #1 {
2385   %\ThisStyle{\if D\m@switch
2386   %   \exp_args:Nnx \use:nn
2387   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
2388   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
2389   %   \else
2390   %   \exp_args:Nnx \use:nn
2391   %   { \l__stex_terms_left_bracket_str #1 }
2392   %   { \l__stex_terms_right_bracket_str }
2393   %\fi}
2394 }

```

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

\withbrackets

```

2395 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2396   \exp_args:Nnx \use:nn
2397   {
2398     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
2399     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
2400     #3
2401   }
2402   {
2403     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
2404     {\l__stex_terms_left_bracket_str}
2405     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str

```

```

2406     {\l__stex_terms_right_bracket_str}
2407   }
2408 }

```

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

#### `\STEXinvisible`

```

2409 \cs_new_protected:Npn \STEXinvisible #1 {
2410   \stex_annotate_invisible:n { #1 }
2411 }

```

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

OMDoc terms:

#### `\_stex_term_math_oms:nnnn`

```

2412 \cs_new_protected:Nn \_stex_term_oms:nnn {
2413   \stex_annotate:nnn{ OMID }{ #2 }{
2414     \stex_highlight_term:nn { #1 } { #3 }
2415   }
2416 }
2417
2418 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2419   \__stex_terms_maybe_brackets:nn { #3 }{
2420     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2421   }
2422 }

```

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

#### `\_stex_term_math_oma:nnnn`

```

2423 \cs_new_protected:Nn \_stex_term_oma:nnn {
2424   \stex_annotate:nnn{ OMA }{ #2 }{
2425     \stex_highlight_term:nn { #1 } { #3 }
2426   }
2427 }
2428
2429 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2430   \__stex_terms_maybe_brackets:nn { #3 }{
2431     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2432   }
2433 }

```

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

#### `\_stex_term_math_omb:nnnn`

```

2434 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2435   \stex_annotate:nnn{ OMBIND }{ #2 }{
2436     \stex_highlight_term:nn { #1 } { #3 }
2437   }
2438 }
2439
2440 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2441   \__stex_terms_maybe_brackets:nn { #3 }{
2442     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2443   }
2444 }

```

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

`\_stex_term_math_arg:nnn`

```

2445 \cs_new_protected:Nn \_stex_term_arg:nn {
2446   \stex_unhighlight_term:n {
2447     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2448   }
2449 }
2450 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
2451   \exp_args:Nnx \use:nn
2452     { \int_set:Nn \l__stex_terms_downprec { #2 }
2453       \_stex_term_arg:nn { #1 }{ #3 }
2454     }
2455   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2456 }

```

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

`\_stex_term_math_assoc_arg:nnnn`

```

2457 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
2458   \seq_set_split:Nnn \l_tmpa_seq , { #4 }
2459   \int_compare:nNnTF { \seq_count:N \l_tmpa_seq } < 2 {
2460     \tl_set:Nn \l_tmpa_tl { #4 }
2461   }{
2462     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
2463     \seq_reverse:N \l_tmpa_seq
2464     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_tl
2465     \tl_set:No \l_tmpa_tl { \l_tmpb_tl }
2466
2467     \seq_map_inline:Nn \l_tmpa_seq {
2468       \exp_args:Nno \tl_set:No \l_tmpa_tl {
2469         \exp_args:Nno
2470         \l_tmpa_cs { ##1 } \l_tmpa_tl
2471       }
2472     }
2473
2474   }
2475   \exp_args:Nnno
2476   \_stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2477 }

```

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

`\stex_term_custom:nn`

```

2478 \cs_new_protected:Nn \stex_term_custom:nn {
2479   \str_set:Nn \l__stex_terms_custom_uri { #1 }
2480   \str_set:Nn \l_tmpa_str { #2 }
2481   \tl_clear:N \l_tmpa_tl
2482   \int_zero:N \l_tmpa_int
2483   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2484   \__stex_terms_custom_loop:
2485 }

```

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

\\_stex\_terms\_custom\_loop:

```

2486 \cs_new_protected:Nn \_stex_terms_custom_loop: {
2487   \bool_set_false:N \l_tmpa_bool
2488   \bool_while_do:nn {
2489     \str_if_eq_p:ee X {
2490       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2491     }
2492   }{
2493     \int_incr:N \l_tmpa_int
2494   }
2495
2496   \peek_charcode:NTF [ {
2497     % notation/text component
2498     \_stex_terms_custom_component:w
2499   } {
2500     \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2501       % all arguments read => finish
2502       \_stex_terms_custom_final:
2503     } {
2504       % arguments missing
2505       \peek_charcode_remove:NTF * {
2506         % invisible, specific argument position or both
2507         \peek_charcode:NTF [ {
2508           % visible specific argument position
2509           \_stex_terms_custom_arg:wn
2510         } {
2511           % invisible
2512           \peek_charcode_remove:NTF * {
2513             % invisible specific argument position
2514             \_stex_terms_custom_arg_inv:wn
2515           } {
2516             % invisible next argument
2517             \_stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2518           }
2519         }
2520       } {
2521         % next normal argument
2522         \_stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
2523       }
2524     }
2525   }
2526 }

```

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

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

```

2527 \cs_new_protected:Npn \_stex_terms_custom_arg_inv:wn [ #1 ] #2 {
2528   \bool_set_true:N \l_tmpa_bool
2529   \_stex_terms_custom_arg:wn [ #1 ] { #2 }
2530 }

```

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

\\_stex\_terms\_custom\_arg:wn

```

2531 \cs_new_protected:Npn \__stex_terms_custom_arg:wn [ #1 ] #2 {
2532   \str_set:Nx \l_tmpb_str {
2533     \str_item:Nn \l_tmpa_str { #1 }
2534   }
2535   \str_case:VnTF \l_tmpb_str {
2536     { X } {
2537       \msg_error:nnn{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2538     }
2539     { i } { \__stex_terms_custom_set_X:n { #1 } }
2540     { b } { \__stex_terms_custom_set_X:n { #1 } }
2541     { a } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
2542     { B } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
2543   }{}{
2544     \msg_error:nnn{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2545   }
2546
2547   \bool_if:nTF \l_tmpa_bool {
2548     \tl_put_right:Nx \l_tmpa_tl {
2549       \stex_annotate_invisible:n {
2550         \stex_term_arg:nn { \int_eval:n { #1 } }
2551         \exp_not:n { { #2 } }
2552       }
2553     }
2554   } {
2555     \tl_put_right:Nx \l_tmpa_tl {
2556       \stex_term_arg:nn { \int_eval:n { #1 } }
2557       \exp_not:n { { #2 } }
2558     }
2559   }
2560
2561   \__stex_terms_custom_loop:
2562 }

```

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

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

```

2563 \cs_new_protected:Nn \__stex_terms_custom_set_X:n {
2564   \str_set:Nx \l_tmpa_str {
2565     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
2566     X
2567     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
2568   }
2569 }

```

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

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

```

2570 \cs_new_protected:Npn \__stex_terms_custom_component:w [ #1 ] {
2571   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
2572   \__stex_terms_custom_loop:
2573 }

```

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

`\_stex_terms_custom_final:`

```

2574 \cs_new_protected:Nn \_stex_terms_custom_final: {
2575   \int_compare:nNnTF \l_tmpb_int = 0 {
2576     \exp_args:Nnno \_stex_term_oms:nnn
2577   }{
2578     \str_if_in:NnTF \l_tmpa_str {b} {
2579       \exp_args:Nnno \_stex_term_ombind:nnn
2580     } {
2581       \exp_args:Nnno \_stex_term_oma:nnn
2582     }
2583   }
2584   { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
2585 }

```

(End definition for `\_stex_terms_custom_final:.`)

**`\symref`**

**`\symname`**

```

2586 \NewDocumentCommand \symref { m m }{
2587   \let\compemph_uri_prev:\compemph@uri
2588   \let\compemph@uri\symrefemph@uri
2589   \STEXsymbol{#1}![#2]
2590   \let\compemph@uri\compemph_uri_prev:
2591 }
2592
2593 \keys_define:nn { stex / symname } {
2594   post      .str_set_x:N      = \l_stex_symname_post_str
2595 }
2596
2597 \cs_new_protected:Nn \stex_symname_args:n {
2598   \str_clear:N \l_stex_symname_post_str
2599   \keys_set:nn { stex / symname } { #1 }
2600 }
2601
2602 \NewDocumentCommand \symname { 0{} m }{
2603   \stex_symname_args:n { #1 }
2604   \stex_get_symbol:n { #2 }
2605   \str_set:Nx \l_tmpa_str {
2606     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
2607   }
2608   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
2609
2610   \let\compemph_uri_prev:\compemph@uri
2611   \let\compemph@uri\symrefemph@uri
2612   \exp_args:NNx \use:nn
2613   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }![
2614     \l_tmpa_str \l_stex_symname_post_str
2615   ] }
2616   \let\compemph@uri\compemph_uri_prev:
2617 }

```

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



## 24.3 Notation Components

2618  $\langle @@=\text{stex\_notationcomps} \rangle$

$\backslash\text{stex\_highlight\_term:nn}$

```

2619
2620 \str_new:N \l__stex_notationcomps_highlight_uri_str
2621 \cs_new_protected:Nn \stex_highlight_term:nn {
2622   \exp_args:Nnx
2623   \use:nn {
2624     \str_set:Nx \l__stex_notationcomps_highlight_uri_str { #1 }
2625     #2
2626   } {
2627     \str_set:Nx \exp_not:N \l__stex_notationcomps_highlight_uri_str
2628     { \l__stex_notationcomps_highlight_uri_str }
2629   }
2630 }
2631
2632 \cs_new_protected:Nn \stex_unhighlight_term:n {
2633   % \latexml_if:TF {
2634   %   #1
2635   % } {
2636   %   \scalatex_if:TF {
2637   %     #1
2638   %   } {
2639     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
2640   % }
2641   % }
2642 }
```

(End definition for  $\backslash\text{stex\_highlight\_term:nn}$ . This function is documented on page 29.)

```

\comp
\compemph@uri
\compemph
\defemph
\defemph@uri
\symrefemph
\symrefemph@uri
2643 \cs_new_protected:Npn \comp #1 {
2644   \str_if_empty:NF \l__stex_notationcomps_highlight_uri_str {
2645     \scalatex_if:TF {
2646       \stex_annotate:nnn { comp } { \l__stex_notationcomps_highlight_uri_str } { #1 }
2647     } {
2648       \exp_args:Nnx \compemph@uri { #1 } { \l__stex_notationcomps_highlight_uri_str }
2649     }
2650   }
2651 }
2652
2653 \cs_new_protected:Npn \compemph@uri #1 #2 {
2654   \compemph{ #1 }
2655 }
2656
2657
2658 \cs_new_protected:Npn \compemph #1 {
2659   \textcolor{blue}{#1}
2660 }
2661
2662 \cs_new_protected:Npn \defemph@uri #1 #2 {
2663   \defemph{#1}
2664 }
```

```

2665
2666 \cs_new_protected:Npn \defemph #1 {
2667   \textbf{#1}
2668 }
2669
2670 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
2671   \symrefemph{#1}
2672 }
2673
2674 \cs_new_protected:Npn \symrefemph #1 {
2675   \textbf{#1}
2676 }

```

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

### `\ellipses`

```

2677 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix
\parrayline
\parraylineh
\parraycell
2678 \bool_new:N \l_stex_inarray_bool
2679 \bool_set_false:N \l_stex_inarray_bool
2680 \NewDocumentCommand \parray { m m } {
2681   \begingroup
2682   \bool_set_true:N \l_stex_inarray_bool
2683   \begin{array}{#1}
2684     #2
2685   \end{array}
2686   \endgroup
2687 }
2688
2689 \NewDocumentCommand \prmatrix { m } {
2690   \begingroup
2691   \bool_set_true:N \l_stex_inarray_bool
2692   \begin{matrix}
2693     #1
2694   \end{matrix}
2695   \endgroup
2696 }
2697
2698 \def \parrayline #1 #2 {
2699   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
2700 }
2701
2702 \def \parraylineh #1 #2 {
2703   #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
2704 }
2705
2706 \def \parraycell #1 {
2707   #1 \bool_if:NT \l_stex_inarray_bool {\&}
2708 }

```

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

```

2709 \endpackage

```

## Chapter 25

# STEX -Structural Features Implementation

```
2710 <*package>
2711
2712 %%%%%%%%%%% features.dtx %%%%%%%%%%%
2713
2714 <@@=stex_features>
      Warnings and error messages
2715
```

### 25.1 The feature environment

structural@feature

```
2716
2717 \NewDocumentEnvironment{structural@feature}{ m m m }{
2718   \stex_if_in_module:F {
2719     \msg_set:nnn{stex}{error/nomodule}{
2720       Structural~Feature~has~to~occur~in~a~module:\\
2721       Feature~#2~of~type~#1\\
2722       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
2723     }
2724     \msg_error:nn{stex}{error/nomodule}
2725   }
2726
2727   \str_set:Nx \l_stex_module_name_str {
2728     \prop_item:Nn \l_stex_current_module_prop
2729     { name } / #2 - feature
2730   }
2731
2732   \str_set:Nx \l_stex_module_ns_str {
2733     \prop_item:Nn \l_stex_current_module_prop
2734     { ns }
2735   }
2736
```

```

2737
2738 \str_clear:N \l_tmpa_str
2739 \seq_clear:N \l_tmpa_seq
2740 \tl_clear:N \l_tmpa_tl
2741 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
2742   origname = #2,
2743   name     = \l_stex_module_name_str ,
2744   ns       = \l_stex_module_ns_str ,
2745   imports  = \exp_not:o { \l_tmpa_seq } ,
2746   constants = \exp_not:o { \l_tmpa_seq } ,
2747   content  = \exp_not:o { \l_tmpa_tl } ,
2748   file     = \exp_not:o { \g_stex_currentfile_seq } ,
2749   lang     = \l_stex_module_lang_str ,
2750   sig      = \l_tmpa_str ,
2751   meta     = \l_tmpa_str ,
2752   feature  = #1 ,
2753 }
2754
2755 \stex_if_smsmode:TF {
2756   \stex_smsmode_set_codes:
2757 } {
2758   \begin{stex_annotate_env}{ feature:#1 }{}
2759   \stex_annotate_invisible:nnn{header}{}{ #3 }
2760 }
2761 }{
2762   \str_set:Nx \l_tmpa_str {
2763     c_stex_feature_
2764     \prop_item:Nn \l_stex_current_module_prop { ns } ?
2765     \prop_item:Nn \l_stex_current_module_prop { name }
2766     _prop
2767   }
2768   \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
2769   \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
2770   \stex_if_smsmode:TF {
2771     \exp_args:Nx \stex_add_to_sms:n {
2772       \prop_gset_from_keyval:cn {
2773         c_stex_feature_
2774         \prop_item:Nn \l_stex_current_module_prop { ns } ?
2775         \prop_item:Nn \l_stex_current_module_prop { name }
2776         _prop
2777       } {
2778         origname = #2,
2779         name     = \prop_item:cn { \l_tmpa_str } { name } ,
2780         ns       = \prop_item:cn { \l_tmpa_str } { ns } ,
2781         imports  = \prop_item:cn { \l_tmpa_str } { imports } ,
2782         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
2783         content  = \prop_item:cn { \l_tmpa_str } { content } ,
2784         file     = \prop_item:cn { \l_tmpa_str } { file } ,
2785         lang     = \prop_item:cn { \l_tmpa_str } { lang } ,
2786         sig      = \prop_item:cn { \l_tmpa_str } { sig } ,
2787         meta     = \prop_item:cn { \l_tmpa_str } { meta } ,
2788         feature  = \prop_item:cn { \l_tmpa_str } { feature }
2789       }
2790     }

```

```

2791 } {
2792     \end{stex_annotate_env}
2793 }
2794 }
2795

```

## 25.2 Features

structure

```

2796
2797 \prop_new:N \l_stex_all_structures_prop
2798
2799 \keys_define:nn { stex / features / structure } {
2800     name .str_set_x:N = \l__stex_features_structure_name_str ,
2801 }
2802
2803 \cs_new_protected:Nn \__stex_features_structure_args:n {
2804     \str_clear:N \l__stex_features_structure_name_str
2805     \keys_set:nn { stex / features / structure } { #1 }
2806 }
2807
2808 %\stex_new_feature:nnnn { structure } { 0{ } m } {
2809 % \__stex_features_structure_args:n { ##1 }
2810 % \str_if_empty:NT \l__stex_features_structure_name_str {
2811 %     \str_set:Nx \l__stex_features_structure_name_str { ##2 }
2812 % }
2813 %} {
2814 %
2815 %}
2816
2817 \NewDocumentEnvironment{mathstructure}{ 0{ } m }{
2818     \__stex_features_structure_args:n { #1 }
2819     \str_if_empty:NT \l__stex_features_structure_name_str {
2820         \str_set:Nx \l__stex_features_structure_name_str { #2 }
2821     }
2822     \exp_args:Nnnx
2823     \begin{structural@feature}{ structure }
2824         { \l__stex_features_structure_name_str }{}
2825         \seq_clear:N \l_tmpa_seq
2826         \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
2827     }{
2828
2829         \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
2830         \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
2831         \str_set:Nx \l_tmpa_str {
2832             \prop_item:Nn \l_stex_current_module_prop { ns } ?
2833             \prop_item:Nn \l_stex_current_module_prop { name }
2834         }
2835         \seq_map_inline:Nn \l_tmpa_seq {
2836             \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
2837         }
2838         \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
2839         \exp_args:Nnx

```

```

2840 \AddToHookNext { env / mathstructure / after }{
2841 \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}{
2842 \_stex_term_math_oms:nnnn { \l_tmpa_str }{}{0}{}}
2843 }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
2844 \STEXexport {
2845 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2846 {\prop_item:Nn \l_stex_current_module_prop { origname }}
2847 {\l_tmpa_str}
2848 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2849 {#2}{\l_tmpa_str}
2850 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
2851 % \prop_item:Nn \l_stex_current_module_prop { origname },
2852 % \l_tmpa_str
2853 % }
2854 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
2855 % #2,\l_tmpa_str
2856 % }
2857 % \tl_set:cx { #2 } {
2858 % \stex_invoke_structure:n { \l_tmpa_str }
2859 % }
2860 % }
2861
2862 \end{structural@feature}
2863 % \g_stex_last_feature_prop
2864 }

```

\instantiate

```

2865 \seq_new:N \l__stex_features_structure_field_seq
2866 \str_new:N \l__stex_features_structure_field_str
2867 \str_new:N \l__stex_features_structure_def_tl
2868 \prop_new:N \l__stex_features_structure_prop
2869 \NewDocumentCommand \instantiate { m O{} m }{
2870 \stex_smsmode_set_codes:
2871 \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
2872 \prop_set_eq:Nc \l__stex_features_structure_prop {
2873 c_stex_feature_\l_tmpa_str _prop
2874 }
2875 \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
2876 \seq_map_inline:Nn \l__stex_features_structure_field_seq {
2877 \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
2878 \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
2879 \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
2880 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
2881 {!} \l_tmpa_tl
2882 \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
2883 \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
2884 \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
2885 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
2886 }{
2887 \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
2888 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
2889 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
2890 \l_tmpa_tl
2891 \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {

```

```

2892         \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
2893         \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
2894     }{
2895         \tl_clear:N \l_tmpb_tl
2896     }
2897 }
2898 }{
2899     \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
2900     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
2901         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
2902         \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
2903         \tl_clear:N \l_tmpa_tl
2904     }{
2905         % TODO throw error
2906     }
2907 }
2908 % \l_tmpa_str: name
2909 % \l_tmpa_tl: definiens
2910 % \l_tmpb_tl: notation
2911 \tl_if_empty:NT \l__stex_features_structure_field_str {
2912     % TODO throw error
2913 }
2914 \str_clear:N \l_tmpb_str
2915
2916 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
2917 \seq_map_inline:Nn \l_tmpa_seq {
2918     \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
2919     \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
2920     \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
2921         \seq_map_break:n {
2922             \str_set:Nn \l_tmpb_str { ####1 }
2923         }
2924     }
2925 }
2926 \prop_get:cnN { g_stex_symdecl_ \l_tmpb_str _prop } {args}
2927     \l_tmpb_str
2928
2929 \tl_if_empty:NTF \l_tmpb_tl {
2930     \tl_if_empty:NF \l_tmpa_tl {
2931         \exp_args:Nx \use:n {
2932             \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
2933         }
2934     }
2935 }{
2936     \tl_if_empty:NTF \l_tmpa_tl {
2937         \exp_args:Nx \use:n {
2938             \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\
2939         }
2940     }
2941 }{
2942     \exp_args:Nx \use:n {
2943         \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
2944         \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
2945     }

```

```

2946     }
2947   }
2948   % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
2949   % \prop_item:Nn \l_stex_current_module_prop {name} ?
2950   % #3/\l_stex_features_structure_field_str
2951   % \par
2952   % \expandafter\present\csname
2953   %   g_stex_symdecl_
2954   %   \prop_item:Nn \l_stex_current_module_prop {ns} ?
2955   %   \prop_item:Nn \l_stex_current_module_prop {name} ?
2956   %   #3/\l_stex_features_structure_field_str
2957   %   _prop
2958   % \endcsname
2959 }
2960
2961 \tl_clear:N \l__stex_features_structure_def_tl
2962
2963 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
2964 \seq_map_inline:Nn \l_tmpa_seq {
2965   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
2966   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
2967   \exp_args:Nx \use:n {
2968     \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
2969
2970     }
2971   }
2972
2973   \prop_if_exist:cF {
2974     g_stex_symdecl_
2975     \prop_item:Nn \l_stex_current_module_prop {ns} ?
2976     \prop_item:Nn \l_stex_current_module_prop {name} ?
2977     #3/\l_tmpa_str
2978     _prop
2979   }{
2980     \prop_get:cnN { g_stex_symdecl_ ##1 _prop } {args}
2981     \l_tmpb_str
2982     \exp_args:Nx \use:n {
2983       \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
2984     }
2985   }
2986 }
2987
2988 \symdecl*[type={\STEXsymbol{module-type}}{
2989   \_stex_term_math_oms:nnnn {
2990     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
2991     \prop_item:Nn \l__stex_features_structure_prop {name}
2992     }{}{0}{}
2993   }{}{#3}
2994
2995   % TODO: -> sms file
2996
2997   \tl_set:cx{ #3 }{
2998     \stex_invoke_structure:nnn {
2999       \prop_item:Nn \l_stex_current_module_prop {ns} ?

```



```

3000     \prop_item:Nn \l_stex_current_module_prop {name} ? #3
3001   } {
3002     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3003     \prop_item:Nn \l__stex_features_structure_prop {name}
3004   }
3005 }
3006
3007 }

```

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

\stex\_invoke\_structure:nnn

```

3008 % #1: URI of the instance
3009 % #2: URI of the instantiated module
3010 \cs_new_protected:Nn \stex_invoke_structure:nnn {
3011   \tl_if_empty:nTF{ #3 }{
3012     \prop_set_eq:Nc \l__stex_features_structure_prop {
3013       c_stex_feature_ #2 _prop
3014     }
3015     \tl_clear:N \l_tmpa_tl
3016     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3017     \seq_map_inline:Nn \l_tmpa_seq {
3018       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3019       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3020       \cs_if_exist:cT {
3021         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
3022       }{
3023         \tl_if_empty:NF \l_tmpa_tl {
3024           \tl_put_right:Nn \l_tmpa_tl {,}
3025         }
3026         \tl_put_right:Nx \l_tmpa_tl {
3027           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
3028         }
3029       }
3030     }
3031     \exp_args:No \mathstrut \l_tmpa_tl
3032   }{
3033     \stex_invoke_symbol:n{#1/#3}
3034   }
3035 }

```

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

```

3036 </package>

```

## Chapter 26

# STEX -Statements Implementation

```
3037 <*package>
3038
3039 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3040
3041 \protected\def\ignorespacesandpars{
3042   \begingroup\catcode13=10\relax
3043   \@ifnextchar\par{
3044     \endgroup\expandafter\ignorespacesandpars\@gobble
3045   }{
3046     \endgroup
3047   }
3048 }
3049
3050 <@@=stex_statements>
3051
3052   Warnings and error messages
3053
3054 \def\titleemph#1{\textbf{#1}}
3055
symboldoc
3056 \NewDocumentEnvironment{symboldoc}{m}{
3057   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3058   \seq_clear:N \l_tmpb_seq
3059   \seq_map_inline:Nn \l_tmpa_seq {
3060     \str_if_eq:nnF{ ##1 }{}{
3061       \stex_get_symbol:n { ##1 }
3062       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3063         \l_stex_get_symbol_uri_str
3064       }
3065     }
3066   }
3067   \par
3068   \exp_args:Nnnx
3069   \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
3070 }
```

```

3068 \end{stex_annotate_env}
3069 }

3070 \seq_new:N \g_stex_statements_patched_seq
3071
3072 \cs_new_protected:Nn \stex_statements_set_patched:n {
3073   \seq_put_right:Nn \g_stex_statements_patched_seq {#1}
3074 }
3075
3076 \cs_new_protected:Nn \stex_statements_patch:nn {
3077   \seq_if_in:NnF \g_stex_statements_patched_seq {#1} {
3078     \AddToHook{begindocument}{
3079       \cs_if_exist:cTF{end#1}{
3080         \AddToHook{env/#1/before}[stex]{\use:c{__stex_statements_#2_begin:n}{}}
3081         \AddToHook{env/#1/after}[stex]{\use:c{__stex_statements_#2_end:}}
3082       }{
3083         \NewDocumentEnvironment{#1}{0{}}{
3084           \use:c{__stex_statements_#2_begin:n}{ }
3085         }{
3086           \use:c{__stex_statements_#2_end:}
3087         }
3088       }
3089     }
3090   }
3091 }

```

## 26.1 Definitions

definition

```

3092
3093 \NewDocumentCommand \definiendum { 0{ } m m } {
3094   \stex_get_symbol:n { #2 }
3095   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3096   \scalatex_if:TF {
3097     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { #3 }
3098   } {
3099     \exp_args:Nnx \defemph@uri { #3 } { \l_stex_get_symbol_uri_str }
3100   }
3101 }
3102 \stex_deactivate_macro:Nn \definiendum {definition~environments}
3103 \NewDocumentCommand \definame { 0{ } m } {
3104   % TODO: root
3105   \stex_get_symbol:n { #2 }
3106   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3107   \str_set:Nx \l_tmpa_str {
3108     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3109   }
3110   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3111   \scalatex_if:TF {
3112     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3113       \l_tmpa_str
3114     }
3115   } {

```

```

3116     \defemph@uri {
3117         \l_tmpa_str
3118     } { \l_stex_get_symbol_uri_str }
3119 }
3120 }
3121 \stex_deactivate_macro:Nn \definame {definition-environments}
3122
3123 \cs_new_protected:Nn \__stex_statements_defi_begin:n {
3124     \stex_reactivate_macro:N \definiendum
3125     \stex_reactivate_macro:N \definame
3126     \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3127     \seq_clear:N \l_tmpb_seq
3128     \seq_map_inline:Nn \l_tmpa_seq {
3129         \str_if_eq:nnF{ ##1 }{}{
3130             \stex_get_symbol:n { ##1 }
3131             \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3132                 \l_stex_get_symbol_uri_str
3133             }
3134         }
3135     }
3136     \stex_smsmode_set_codes:
3137     \exp_args:Nnnx
3138     \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
3139 }
3140
3141 \cs_new_protected:Nn \__stex_statements_defi_end: {
3142     \end{stex_annotate_env}
3143 }

```

Hook:

```

3144 \stex_statements_patch:nn{definition}{defi}

inline:
3145 \NewDocumentCommand \inlinedef { m } {
3146     \begingroup
3147     \stex_reactivate_macro:N \definiendum
3148     \stex_reactivate_macro:N \definame
3149     \stex_ref_new_doc_target:n{
3150         #1
3151     }
3152 }

```

## 26.2 Assertions

**assertion**

```

3153 \cs_new_protected:Nn \__stex_statements_assertion_begin:n {
3154     \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3155     \seq_clear:N \l_tmpb_seq
3156     \seq_map_inline:Nn \l_tmpa_seq {
3157         \str_if_eq:nnF{ ##1 }{}{
3158             \stex_get_symbol:n { ##1 }
3159             \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3160                 \l_stex_get_symbol_uri_str

```

```

3161     }
3162   }
3163 }
3164 \titleemph{Assertion}~
3165 \stex_smsmode_set_codes:
3166 \exp_args:Nnnx
3167 \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
3168 }
3169
3170 \cs_new_protected:Nn \__stex_statements_assertion_end: {
3171   \end{stex_annotate_env}
3172 }

```

Hook:

```

3173 \stex_statements_patch:nn{assertion}{assertion}

inline:
3174 \NewDocumentCommand \inlineass { m } {
3175   \begingroup
3176   \stex_ref_new_doc_target:n{
3177     #1
3178   }
3179 }

```

theorem

```

3180 \cs_new_protected:Nn \__stex_statements_theorem_begin:n {
3181   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3182   \seq_clear:N \l_tmpb_seq
3183   \seq_map_inline:Nn \l_tmpa_seq {
3184     \str_if_eq:nnF{ ##1 }{}{
3185       \stex_get_symbol:n { ##1 }
3186       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3187         \l_stex_get_symbol_uri_str
3188       }
3189     }
3190   }
3191   \titleemph{Theorem}~
3192   \stex_smsmode_set_codes:
3193   \exp_args:Nnnx
3194   \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
3195 }
3196
3197 \cs_new_protected:Nn \__stex_statements_theorem_end: {
3198   \end{stex_annotate_env}
3199 }

```

Hook:

```

3200 \stex_statements_patch:nn{theorem}{theorem}

```

lemma

```

3201 \cs_new_protected:Nn \__stex_statements_lemma_begin:n {
3202   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3203   \seq_clear:N \l_tmpb_seq

```

```

3204 \seq_map_inline:Nn \l_tmpa_seq {
3205 \str_if_eq:nnF{ ##1 }{}{
3206   \stex_get_symbol:n { ##1 }
3207   \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3208     \l_stex_get_symbol_uri_str
3209   }
3210 }
3211 }
3212 \titleemph{Lemma}~
3213 \stex_smsmode_set_codes:
3214 \exp_args:Nnnx
3215 \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
3216 }
3217
3218 \cs_new_protected:Nn \__stex_statements_lemma_end: {
3219   \end{stex_annotate_env}
3220 }

```

Hook:

```

3221 \stex_statements_patch:nn{lemma}{lemma}

```

**axiom**

```

3222 \cs_new_protected:Nn \__stex_statements_axiom_begin:n {
3223   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3224   \seq_clear:N \l_tmpb_seq
3225   \seq_map_inline:Nn \l_tmpa_seq {
3226     \str_if_eq:nnF{ ##1 }{}{
3227       \stex_get_symbol:n { ##1 }
3228       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3229         \l_stex_get_symbol_uri_str
3230       }
3231     }
3232   }
3233   \titleemph{Axiom}~
3234   \stex_smsmode_set_codes:
3235   \exp_args:Nnnx
3236   \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
3237 }
3238
3239 \cs_new_protected:Nn \__stex_statements_axiom_end: {
3240   \end{stex_annotate_env}
3241 }

```

Hook:

```

3242 \stex_statements_patch:nn{axiom}{axiom}

```

## 26.3 Examples

**example**

```

3243 \cs_new_protected:Nn \__stex_statements_example_begin:n {
3244   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3245   \seq_clear:N \l_tmpb_seq

```

```

3246 \seq_map_inline:Nn \l_tmpa_seq {
3247   \str_if_eq:nnF{ ##1 }{}{
3248     \stex_get_symbol:n { ##1 }
3249     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3250       \l_stex_get_symbol_uri_str
3251     }
3252   }
3253 }
3254 \titleemph{Example}~
3255 \stex_smsmode_set_codes:
3256 \exp_args:Nnnx
3257 \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpb_seq {,}}
3258 }
3259
3260 \cs_new_protected:Nn \__stex_statements_example_end: {
3261   \end{stex_annotate_env}
3262 }

```

Hook:

```

3263 \stex_statements_patch:nn{example}{example}

inline:
3264 \NewDocumentCommand \inlineex { m } {
3265   \begingroup
3266   \stex_ref_new_doc_target:n{
3267     #1
3268   }
3269 }

```

## 26.4 OMText

```

3270 \keys_define:nn { stex / omtext } {
3271   id      .str_set_x:N = \l_stex_omtext_id_str ,
3272   title   .tl_set:N    = \l_stex_omtext_title_tl ,
3273   type    .tl_set_x:N  = \l_stex_omtext_type_tl ,
3274   for     .tl_set_x:N  = \l_stex_omtext_for_tl ,
3275   from    .tl_set_x:N  = \l_stex_omtext_from_tl ,
3276   start   .tl_set:N    = \l_stex_omtext_start_tl ,
3277 }
3278 \cs_new_protected:Nn \stex_omtext_args:n {
3279   \tl_clear:N \l_stex_omtext_title_tl
3280   \tl_clear:N \l_stex_omtext_start_tl
3281   \keys_set:nn { stex / omtext } { #1 }
3282 }
3283 \newif\if@in@omtext\@in@omtextfalse
3284 \NewDocumentEnvironment {omtext} { 0 } { } {
3285   \stex_omtext_args:n { #1 }
3286   \tl_if_empty:NTF \l_stex_omtext_start_tl {
3287     \tl_if_empty:NF \l_stex_omtext_title_tl {
3288       \titleemph{\l_stex_omtext_title_tl}:~
3289     }
3290   }{
3291     \titleemph{\l_stex_omtext_start_tl}~

```

```

3292 }
3293 \@in@omtexttrue
3294
3295 \stex_ref_new_doc_target:n \l_stex_omtext_id_str
3296 \stex_smsmode_set_codes:
3297 \ignorespacesandpars
3298 }{}
3299 \end{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>

```
3300 <*package>
3301 <@@=stex_sproof>
3302
3303 %%%%%%%%%%% sproof.dtx %%%%%%%%%%%
3304
```

### 27.2 Proofs

We first define some keys for the proof environment.

```
3305 \keys_define:nn { stex / spf } {
3306   id          .str_set:N = \l__stex_sproof_spf_id_str,
3307   display     .tl_set:N  = \l__stex_sproof_spf_display_tl,
3308   for         .tl_set:N  = \l__stex_sproof_spf_for_tl ,
3309   from        .tl_set:N  = \l__stex_sproof_spf_from_tl ,
3310   proofend    .tl_set:N  = \l__stex_sproof_spf_proofend_tl,
3311   type        .tl_set:N  = \l__stex_sproof_spf_type_tl,
3312   title       .tl_set:N  = \l__stex_sproof_spf_title_tl,
3313   continues   .tl_set:N  = \l__stex_sproof_spf_continues_tl,
3314   functions   .tl_set:N  = \l__stex_sproof_spf_functions_tl,
3315   method      .tl_set:N  = \l__stex_sproof_spf_method_tl
3316 }
3317 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
3318   \str_clear:N \l__stex_sproof_spf_id_str
3319   \tl_clear:N \l__stex_sproof_spf_display_tl
3320   \tl_clear:N \l__stex_sproof_spf_for_tl
3321   \tl_clear:N \l__stex_sproof_spf_from_tl
3322   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
3323   \tl_clear:N \l__stex_sproof_spf_type_tl
3324   \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>XML

```

3325 \tl_clear:N \l__stex_sproof_spf_continues_tl
3326 \tl_clear:N \l__stex_sproof_spf_functions_tl
3327 \tl_clear:N \l__stex_sproof_spf_method_tl
3328 \keys_set:nn { stex / spf }{ #1 }
3329 }

```

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

```

3330 \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.

```

3331 \newcount\count_ten
3332 \newenvironment{pst@with@label}[1]{
3333   \edef\pst@label{#1}
3334   \advance\count_ten by 1\relax
3335   \count_ten=1
3336 }{
3337   \advance\count_ten by -1\relax
3338 }

```

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

```

3339 \def\the@pst@label{
3340   \pst@make@label\pst@label{\number\count_ten}\l__stex_sproof_pstlabel_postfix_tl
3341 }

```

(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.

```

3342 \keys_define:nn { stex / pstlabel }{
3343   prefix      .tl_set:N   = \l__stex_sproof_pstlabel_prefix_tl,
3344   delimiter   .tl_set:N   = \l__stex_sproof_pstlabel_delimiter_tl,
3345   postfix     .tl_set:N   = \l__stex_sproof_pstlabel_postfix_tl
3346 }
3347 \cs_new_protected:Nn \__stex_sproof_pstlabel_args:n {

```

---

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

```

3348 \tl_set:Nn \l__stex_sproof_pstlabel_prefix_tl {P}
3349 \tl_set:Nn \l__stex_sproof_pstlabel_delimiter_tl {.}
3350 \tl_clear:N \l__stex_sproof_pstlabel_postfix_tl
3351 }
3352 \__stex_sproof_pstlabel_args:n {}
3353 \newcommand\setpstlabelstyle[1]{
3354   \__stex_sproof_pstlabel_args:n {#1}
3355 }
3356 \newcommand\setpstlabelstyledefault{%
3357   \__stex_sproof_pstlabel_args:n{prefix=P,delimiter=.,postfix={}}
3358 }

```

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

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

```

3359 \ExplSyntaxOff
3360 \def\pst@make@label@long#1#2{\@for\@I:=#1\do{\expandafter\expandafter\expandafter\@I\csname
3361 \def\pst@make@label@angles#1#2{\ensuremath{\@for\@I:=#1\do{\rangle}}#2}
3362 \def\pst@make@label@short#1#2{#2}
3363 \def\pst@make@label@empty#1#2{}
3364 \ExplSyntaxOn
3365 \def\pstlabelstyle#1{%
3366   \def\pst@make@label{\use:c{pst@make@label@#1}}%
3367 }%
3368 \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.

```

3369 \def\next@pst@label{%
3370   \global\advance\count\count10 by 1%
3371 }%

```

(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

```

3372 \def\sproof@box{
3373   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
3374 }
3375 \def\spf@proofend{\sproof@box}
3376 \def\sproofend{
3377   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
3378     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
3379   }
3380 }
3381 \def\sProofEndSymbol#1{\def\sproof@box{#1}}

```

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

**spf@\*@kw**

```

3382 \def\spf@proofsketch@kw{Proof Sketch}
3383 \def\spf@proof@kw{Proof}
3384 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

3385 \cs_if_exist:NT \bbl@loaded {
3386   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
3387   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
3388     \input{proof-ngerman.lda}
3389   }
3390   \clist_if_in:NnT \l_tmpa_clist {finnish}{
3391     \input{proof-finnish.lda}
3392   }
3393   \clist_if_in:NnT \l_tmpa_clist {french}{
3394     \input{proof-french.lda}
3395   }
3396   \clist_if_in:NnT \l_tmpa_clist {russian}{
3397     \input{proof-russian.lda}
3398   }
3399 }
3400
```

**spfsketch**

```

3401 \newcommand\spfsketch[2][]{
3402   \__stex_sproof_spf_args:n{#1}
3403   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
3404     \titleemph{
3405       \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {
3406         \spf@proofsketch@kw
3407       }{
3408         \l__stex_sproof_spf_type_tl
3409       }
3410     }:
3411   }
3412   {-#2}
3413   %\sref@label@id{this \ifx\spf@type\@empty\spf@proofsketch@kw\else\spf@type\fi}
3414   \sproofend
3415 }
```

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

EdN:11  
EdN:12

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

```

3416 \newenvironment{spfeq}[2][]{
3417   \__stex_sproof_spf_args:n{#1}
3418   %\sref@target
3419   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
3420     \titleemph{
3421       \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {
3422         \spf@proof@kw
3423       }{
3424         \l__stex_sproof_spf_type_tl
3425       }
3426     }:

```

<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

```

3427 }
3428 {~#2}
3429 \begin{displaymath}\begin{array}{rcll}
3430 }{
3431 \end{array}\end{displaymath}
3432 }

```

(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.

```

3433 \newenvironment{spf@proof}[2][\{
3434 \_stex_sproof_spf_args:n{#1}
3435 %\sref@target
3436 \count_ten=10
3437 \par\noindent
3438 \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3439 \titleemph{
3440 \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {
3441 \spf@proof@kw
3442 }{
3443 \l__stex_sproof_spf_type_tl
3444 }
3445 } :
3446 }
3447 {~#2}
3448 %\sref@label@id{this \ifx\spf@type\empty\spf@proof@kw\else\spf@type\fi}
3449 \def\pst@label{}
3450 \newcount\pst@count% initialize the labeling mechanism
3451 \begin{description}\begin{pst@with@label}{\l__stex_sproof_pstlabel_prefix_tl}
3452 }{
3453 \end{pst@with@label}\end{description}
3454 }
3455 \newenvironment{sproof}[2][\{ \begin{spf@proof}[#1]{#2} \} \sproofend \end{spf@proof} \}
3456 \newenvironment{sProof}[2][\{ \begin{spf@proof}[#1]{#2} \} \end{spf@proof} \}

```

**\spfidea**

```

3457 \newcommand\spfidea[2][\{
3458 \_stex_sproof_spf_args:n{#1}
3459 \titleemph{
3460 \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {Proof~Idea}{
3461 \l__stex_sproof_spf_type_tl
3462 } :
3463 }~#2
3464 \sproofend
3465 }

```

(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

```

3466 \newenvironment{spfstep}[1][]{
3467   \_stex_sproof_spf_args:n{#1}
3468   \@in@omtexttrue
3469   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3470     \item[\the@pst@label]
3471   }
3472   \tl_if_empty:NF \l__stex_sproof_spf_title_tl {
3473     {(\titleemph{\l__stex_sproof_spf_title_tl})\enspace}
3474   }
3475   %\sref@label@id{\pst@label}
3476   \ignorespacesandpars
3477 }{
3478   \next@pst@label\ignorespacesandpars
3479 }

```

sproofcomment

```

3480 \newenvironment{sproofcomment}[1][]{
3481   \_stex_sproof_spf_args:n{#1}
3482   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3483     \item[\the@pst@label]
3484   }
3485 }{
3486   \next@pst@label
3487 }

```

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.

```

3488 \newenvironment{subproof}[2][]{
3489   \_stex_sproof_spf_args:n{#1}
3490   \def\@test{#2}
3491   \ifx\@test\empty\else
3492     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3493       \item[\the@pst@label]
3494     }{#2}
3495   \fi
3496   \begin{pst@with@label}{\pst@label,\number\count_ten}
3497 }{
3498   \end{pst@with@label}\next@pst@label
3499 }

```

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

```

3500 \newenvironment{spfcases}[2][]{
3501   \def\@test{#1}
3502   \ifx\@test\empty
3503     \begin{subproof}[method=by-cases]{#2}
3504   \else
3505     \begin{subproof}[#1,method=by-cases]{#2}
3506   \fi
3507 }{

```

---

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

```

3508 \end{subproof}
3509 }

```

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

```

3510 \newenvironment{spfcase}[2] [] {
3511   \__stex_sproof_spf_args:n{#1}
3512   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3513     \item[\the@pst@label]
3514   }
3515   \def\@test{#2}
3516   \ifx\@test\@empty
3517   \else
3518     {\titleemph{#2}:~}
3519   \fi
3520   \begin{pst@with@label}{\pst@label,\number\count_ten}
3521 }{
3522   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3523     \sproofend
3524   }
3525   \end{pst@with@label}
3526   \next@pst@label
3527 }

```

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

```

3528 \newcommand\spfcasesketch[3] [] {
3529   \__stex_sproof_spf_args:n{#1}
3530   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3531     \item[\the@pst@label]
3532   }
3533   \def\@test{#2}
3534   \ifx\@test\@empty
3535   \else
3536     {\titleemph{#2}:~}
3537   \fi#3
3538   \next@pst@label
3539 }%

```

## 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.

```

3540 \keys_define:nn { stex / just }{
3541   id          .str_set:x:N = \l__stex_sproof_just_id_str,
3542   method      .tl_set:N   = \l__stex_sproof_just_method_tl,
3543   premises    .tl_set:N   = \l__stex_sproof_just_premises_tl,
3544   args        .tl_set:N   = \l__stex_sproof_just_args_tl
3545 }

```

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**

```
3546 \newenvironment{justification}[1] [] {}{}
```

**\premise**

```
3547 \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.

```
3548 \newcommand\justarg[2] [] {#2}
```

```
3549 \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

```
3550 <*package>
3551
3552 %%%%%%%%%%% others.dtx %%%%%%%%%%%
3553
3554 <@@=stex_others>
    Warnings and error messages
3555 % None

\MSC Math subject classifier

3556 \NewDocumentCommand \MSC {m} {
3557 % TODO
3558 }

(End definition for \MSC. This function is documented on page 10.)
    Patching tikzinput, if loaded
3559 \@ifpackageloaded{tikzinput}{
3560 \RequirePackage{stex-tikzinput}
3561 }{}
3562 </package>
```

## Chapter 29

# STEX -Metatheory Implementation

```
3563 <*package>
3564 <@@=stex_modules>
3565
3566 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
3567
3568 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
3569 \begingroup
3570 \stex_module_setup:nn{
3571   ns=\c_stex_metatheory_ns_str,
3572   meta=NONE
3573 }{Metatheory}
3574 \stex_reactivate_macro:N \symdecl
3575 \stex_reactivate_macro:N \notation
3576 \stex_reactivate_macro:N \symdef
3577 \ExplSyntaxOff
3578 \csname stex_suppress_html:n\endcsname{
3579   % is-a (a:A, a \in A, a is an A, etc.)
3580   \symdecl[args=ai]{isa}
3581   \notation[typed]{isa}{#1 \comp{:} #2}{#1 \comp, #2}
3582   \notation[in]{isa}{#1 \comp\in #2}{#1 \comp, #2}
3583   \notation[pred]{isa}{#2\comp(#1 \comp)}{#1 \comp, #2}
3584
3585   % bind (\forall, \Pi, \lambda etc.)
3586   \symdecl[args=Bi]{bind}
3587   \notation[forall]{bind}{\comp\forall #1.\;#2}{#1 \comp, #2}
3588   \notation[\Pi]{bind}{\comp\prod_{#1}#2}{#1 \comp, #2}
3589   \notation[deffun]{bind}{\comp( #1 \comp{}\;\to\;} #2}{#1 \comp, #2}
3590
3591   % dummy variable
3592   \symdecl{dummyvar}
3593   \notation[underscore]{dummyvar}{\comp\_}
3594   \notation[dot]{dummyvar}{\comp\cdot}
3595   \notation[dash]{dummyvar}{\comp{\rm --}}
3596
3597   %fromto (function space, Hom-set, implication etc.)
```

```

3598 \symdecl[args=ai]{fromto}
3599 \notation[xarrow]{fromto}{#1 \comp\to #2}{#1 \comp\times #2}
3600 \notation[arrow]{fromto}{#1 \comp\to #2}{#1 \comp\to #2}
3601
3602 % mapto (lambda etc.)
3603 %\symdecl[args=Bi]{mapto}
3604 %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
3605 %\notation[lambda]{mapto}{\comp\lambda #1 \comp. \; #2}{#1 \comp, #2}
3606 %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp. \; #2}{#1 \comp, #2}
3607
3608 % function/operator application
3609 \symdecl[args=ia]{apply}
3610 \notation[prec=0;0x\neginfprec,parens]{apply}{#1 \comp( #2 \comp)}{#1 \comp, #2}
3611 \notation[prec=0;0x\neginfprec,lambda]{apply}{#1 \; #2 }{#1 \; #2}
3612
3613 % ‘type’ of all collections (sets, classes, types, kinds)
3614 \symdecl{collection}
3615 \notation[U]{collection}{\comp{\mathcal{U}}{}}
3616 \notation[set]{collection}{\comp{\textsf{Set}}{}}
3617
3618 % sequences
3619 \symdecl[args=1]{seqtype}
3620 \notation[kleene]{seqtype}{#1^{\comp\ast}}
3621
3622 \symdef[args=2,li]{sequence-index}{#1_{#2}}
3623 \notation[ui]{sequence-index}{#1^{#2}}
3624
3625 %\symdef[args=3,li]{sequence-from-to}{#1_{#2}\comp{\,\ellipses,}#1_{#3}}
3626 %\notation[ui]{sequence-from-to}{#1^{#2}\comp{\,\ellipses,}#1^{#3}}
3627 % ^ superceded by \aseqfromto and \livar/\uivar
3628
3629 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{\,\ellipses}}{#1\comp,#2}
3630 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{\,\ellipses,}#2}{#1\comp,#2}
3631 \symdef[args=aai,prec=nobrackets]{aseqfromtovia}{#1\comp{\,\ellipses,}#2\comp{\,\ellipses,}#3}
3632
3633 % letin (‘let’, local definitions, variable substitution)
3634 \symdecl[args=bii]{letin}
3635 \notation[let]{letin}{\comp{\rm let}}{\;#1\comp{=}\;#2\; \comp{\rm in}}{\;#3}
3636 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
3637 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
3638
3639 % structures
3640 \symdecl*[args=1]{module-type}
3641 \notation{module-type}{\mathtt{MOD} #1}
3642 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
3643 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{#1 \comp, #2}
3644
3645 }
3646 \ExplSyntaxOn
3647 \stex_add_to_current_module:n{
3648   \let\nappa\apply
3649   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
3650   \def\livar{\csname sequence-index\endcsname[li]}
3651   \def\uivar{\csname sequence-index\endcsname[ui]}

```

```

3652     \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
3653     \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
3654     \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
3655   }
3656   \__stex_modules_end_module:
3657 \endgroup
3658 </package>

```

## Chapter 30

# Tikzinput Implementation

```
3659 <*package>
3660
3661 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
3662
3663 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
3664 \RequirePackage{l3keys2e}
3665
3666 \keys_define:nn { tikzinput } {
3667   image .bool_set:N = \c_tikzinput_image_bool,
3668   image .default:n = false ,
3669   unknown .code:n = {}
3670 }
3671
3672 \ProcessKeysOptions { tikzinput }
3673
3674 \bool_if:NTF \c_tikzinput_image_bool {
3675   \RequirePackage{graphicx}
3676
3677   \providecommand\usetikzlibrary[]{}
3678   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
3679 }{
3680   \RequirePackage{tikz}
3681   \RequirePackage{standalone}
3682
3683   \newcommand \tikzinput [2] [] {
3684     \setkeys{Gin}{#1}
3685     \ifx \Gin@ewidth \Gin@exclamation
3686       \ifx \Gin@eheight \Gin@exclamation
3687         \input { #2 }
3688       \else
3689         \resizebox{!}{ \Gin@eheight }{
3690           \input { #2 }
3691         }
3692       \fi
3693     \else
3694       \ifx \Gin@eheight \Gin@exclamation
3695         \resizebox{ \Gin@ewidth }{!}{
3696           \input { #2 }
```

```

3697     }
3698     \else
3699         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
3700             \input { #2 }
3701         }
3702     \fi
3703 \fi
3704 }
3705 }
3706
3707 \newcommand \ctikzinput [2] [] {
3708     \begin{center}
3709         \tikzinput [ #1 ] { #2 }
3710     \end{center}
3711 }
3712
3713 \@ifpackageloaded{stex}{
3714     \RequirePackage{stex-tikzinput}
3715 }{}
3716
3717 </package>
3718 <*stex>
3719 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
3720 \RequirePackage{stex}
3721 \RequirePackage{tikzinput}
3722
3723 \newcommand\mhtikzinput [2] [] {%
3724     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
3725     \stex_in_repository:nn\Gin@mhrepos{
3726         \tikzinput [ #1 ] {\mhp{##1}{#2}}
3727     }
3728 }
3729 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [ #1 ] { #2 }\end{center}}
3730 </stex>

```

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

## 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.

```
3731 \*cls
3732 \@@=document_structure
3733 \ProvidesExplClass{omdoc}{2020/10/19}{1.4}{OMDoc Documents}
3734 \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`

```
3735 \keys_define:nn{ document-structure / pkg }{
3736   class      .str_set_x:N = \c_document_structure_class_str,
3737   minimal    .bool_set:N = \c_document_structure_minimal_bool,
3738   report     .code:n      = {
3739     \ClassWarning{omdoc}{the option 'report' is deprecated, use 'class=report', instead}
3740     \str_set:Nn \c_document_structure_class_str {report}
3741   },
3742   book       .code:n      = {
3743     \ClassWarning{omdoc}{the option 'book' is deprecated, use 'class=book', instead}
3744     \str_set:Nn \c_document_structure_class_str {book}
3745   },
3746   bookpart   .code:n      = {
3747     \ClassWarning{omdoc}{the option 'bookpart' is deprecated, use 'class=book,topsect=chapter}
3748     \str_set:Nn \c_document_structure_class_str {book}
3749     \str_set:Nn \c_document_structure_topsect_str {chapter}
3750   },
```

```

3751 docopt      .str_set_x:N = \c_document_structure_docopt_str,
3752 unknown     .code:n      = {
3753   \PassOptionsToPackage{ \CurrentOption }{ omdoc }
3754 }
3755 }
3756 \ProcessKeysOptions{ document-structure / pkg }
3757 \str_if_empty:NT \c_document_structure_class_str {
3758   \str_set:Nn \c_document_structure_class_str {article}
3759 }
3760 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
3761   {\c_document_structure_class_str}
3762

```

### 31.3 Beefing up the document environment

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

```

3763 \RequirePackage{omdoc}
3764 \bool_if:NF \c_document_structure_minimal_bool {
3765   \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>

```

3766 \keys_define:nn { document-structure / document }{
3767   id .str_set_x:N = \c_document_structure_document_id_str
3768 }
3769 \let\__document_structure_orig_document=\document
3770 \renewcommand{\document}[1][]{
3771   \keys_set:nn{ document-structure / document }{ #1 }
3772   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
3773   \__document_structure_orig_document
3774 }

```

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

```

3775 }
3776 \</cls>

```

### 31.4 Implementation: OMDoc Package

```

3777 \*package>
3778 \ProvidesExplPackage{omdoc}{2020/10/19}{1.4}{OMDoc document Structure}
3779 \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



```

3780
3781 \keys_define:nn{ document-structure / pkg }{
3782   class      .str_set_x:N = \c_document_structure_class_str,
3783   topsect    .str_set_x:N = \c_document_structure_topsect_str,
3784   % showignores .bool_set:N = \c_document_structure_showignores_bool,
3785 }
3786 \ProcessKeysOptions{ document-structure / pkg }
3787 \str_if_empty:NT \c_document_structure_class_str {
3788   \str_set:Nn \c_document_structure_class_str {article}
3789 }
3790 \str_if_empty:NT \c_document_structure_topsect_str {
3791   \str_set:Nn \c_document_structure_topsect_str {section}
3792 }

```

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

```

3793 \RequirePackage{xspace}
3794 \RequirePackage{comment}
3795 \@ifpackageloaded{babel}{\RequirePackage[base]{babel}}

```

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

```

3796 \@ifpackageloaded{babel}{
3797   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
3798   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
3799     \input{omdoc-ngerman.ldf}
3800   }
3801 }{}
3802 %\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.

```

3803 \int_new:N \l_document_structure_section_level_int
3804 \str_case:VnF \c_document_structure_topsect_str {
3805   {part}{
3806     \int_set:Nn \l_document_structure_section_level_int {0}
3807   }
3808   {chapter}{
3809     \int_set:Nn \l_document_structure_section_level_int {1}
3810   }
3811 }{
3812   \str_case:VnF \c_document_structure_class_str {
3813     {book}{
3814       \int_set:Nn \l_document_structure_section_level_int {0}
3815     }
3816     {report}{
3817       \int_set:Nn \l_document_structure_section_level_int {0}
3818     }
3819   }{
3820     \int_set:Nn \l_document_structure_section_level_int {2}
3821   }
3822 }

```

## 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

```
3823 \def\current@section@level{document}%
3824 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
3825 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
3826 \cs_new_protected:Npn \skipomgroup {
3827   \ifcase\l_document_structure_section_level_int
3828   \or\stepcounter{part}
3829   \or\stepcounter{chapter}
3830   \or\stepcounter{section}
3831   \or\stepcounter{subsection}
3832   \or\stepcounter{subsubsection}
3833   \or\stepcounter{paragraph}
3834   \or\stepcounter{subparagraph}
3835   \fi
3836 }
```

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

`blindomgroup`

```
3837 \newcommand\at@begin@blindomgroup[1]{%
3838 \newenvironment{blindomgroup}
3839 {
3840   \int_incr:N\l_document_structure_section_level_int
3841   \at@begin@blindomgroup\l_document_structure_section_level_int
3842 }{}}
```

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

```
3843 \newcommand\omgroup@nonum[2]{%
3844   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
3845   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
3846 }
```

*(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.

```
3847 \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.

```

3848 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
3849   \@nameuse{#1}{#2}
3850 }{
3851   \cs_if_exist:NTF\rdfmata@sectioning{
3852     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
3853   }{
3854     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
3855   }
3856 }
3857 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
3858 }

```

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

omgroup

```

3859 \keys_define:nn { document-structure / omgroup }{
3860   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
3861   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
3862   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
3863   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
3864   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
3865   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
3866   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
3867   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
3868   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
3869   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
3870 }
3871 \cs_new_protected:Nn \__document_structure_omgroup_args:n {
3872   \str_clear:N \l__document_structure_omgroup_id_str
3873   \str_clear:N \l__document_structure_omgroup_date_str
3874   \clist_clear:N \l__document_structure_omgroup_creators_clist
3875   \clist_clear:N \l__document_structure_omgroup_contributors_clist
3876   \tl_clear:N \l__document_structure_omgroup_srccite_tl
3877   \tl_clear:N \l__document_structure_omgroup_type_tl
3878   \tl_clear:N \l__document_structure_omgroup_short_tl
3879   \tl_clear:N \l__document_structure_omgroup_display_tl
3880   \tl_clear:N \l__document_structure_omgroup_intro_tl
3881   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
3882   \keys_set:nn { document-structure / omgroup } { #1 }
3883 }

```

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.

```

3884 \newif\if@mainmatter\@mainmattertrue
3885 \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.

```

3886 \keys_define:nn { document-structure / sectioning }{
3887   name .str_set_x:N = \l__document_structure_sect_name_str ,
3888   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
3889   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
3890   num .bool_set:N = \l__document_structure_sect_num_bool ,
3891 }

```

```

3892 \cs_new_protected:Nn \l__document_structure_sect_args:n {
3893   \str_clear:N \l__document_structure_sect_name_str
3894   \str_clear:N \l__document_structure_sect_ref_str
3895   \bool_set_false:N \l__document_structure_sect_clear_bool
3896   \bool_set_false:N \l__document_structure_sect_num_bool
3897   \keys_set:nn { document-structure / sectioning } { #1 }
3898 }
3899 \newcommand\omdoc@sectioning[3][]{
3900   \l__document_structure_sect_args:n {#1}
3901   \let\omdoc@sect@name\l__document_structure_sect_name_str
3902   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
3903   \if@mainmatter% numbering not overridden by frontmatter, etc.
3904     \bool_if:NTF \l__document_structure_sect_num_bool {
3905       \omgroup@num{#2}{#3}
3906     }{
3907       \omgroup@nonum{#2}{#3}
3908     }
3909     \def\current@section@level{\omdoc@sect@name}
3910   \else
3911     \omgroup@nonum{#2}{#3}
3912   \fi
3913 }% 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.

```

3914 \newcommand\omgroup@redefine@addtocontents[1]{%
3915   %\edef\__document_structureimport{#1}%
3916   %\@for\@I:=\__document_structureimport\do{%
3917     %\edef\@path{\csname module@\@I @path\endcsname}%
3918     %\@ifundefined{tf@toc}\relax%
3919     % {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
3920   %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
3921   %\def\addcontentsline##1##2##3{%
3922     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
3923   %\else% hyperref.sty not loaded
3924   %\def\addcontentsline##1##2##3{%
3925     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
3926   %\fi
3927 }% 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.

```

3928 \int_new:N \l__document_structure_omgroup_level_int
3929 \newenvironment{omgroup}[2][]{% keys, title
3930 {
3931   \l__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.

```

3932 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
3933   \omgroup@redefine@addtocontents{
3934     %\@ifundefined{module@id}\used@modules%
3935     %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}

```

```

3936     }
3937 }

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

3938 \int_incr:N \l_document_structure_omgroup_level_int
3939 \int_incr:N \l_document_structure_section_level_int
3940 \ifcase\l_document_structure_section_level_int
3941   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
3942   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
3943   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
3944   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
3945   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
3946   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
3947   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
3948 \fi
3949 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
3950 \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str
3951 }% for customization
3952 {}

```

and finally, we localize the sections

```

3953 \newcommand\omdoc@part@kw{Part}
3954 \newcommand\omdoc@chapter@kw{Chapter}
3955 \newcommand\omdoc@section@kw{Section}
3956 \newcommand\omdoc@subsection@kw{Subsection}
3957 \newcommand\omdoc@subsubsection@kw{Subsubsection}
3958 \newcommand\omdoc@paragraph@kw{paragraph}
3959 \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`

```

3960 \providecommand\printindex{\IfFileExists{\jobname.ind}{\input{\jobname.ind}}{}}

```

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

some classes (e.g. `book.cls`) already have `\frontmatter`, `\mainmatter`, and `\backmatter` macros. As we want to define `frontmatter` and `backmatter` environments, we save their behavior (possibly defining it) in `orig@*matter` macros and make them undefined (so that we can define the environments).

```

3961 \cs_if_exist:NTF\frontmatter{
3962   \let\__document_structure_orig_frontmatter\frontmatter
3963   \let\frontmatter\relax
3964 }{
3965   \tl_set:Nn\__document_structure_orig_frontmatter{
3966     \clearpage
3967     \@mainmatterfalse
3968     \pagenumbering{roman}
3969   }
3970 }
3971 \cs_if_exist:NTF\backmatter{

```

```

3972 \let\__document_structure_orig_backmatter\backmatter
3973 \let\backmatter\relax
3974 }{
3975 \tl_set:Nn\__document_structure_orig_backmatter{
3976 \clearpage
3977 \@mainmatterfalse
3978 \pagenumbering{roman}
3979 }
3980 }

```

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.

```

3981 \newenvironment{frontmatter}{
3982 \__document_structure_orig_frontmatter
3983 }{
3984 \cs_if_exist:NTF\mainmatter{
3985 \mainmatter
3986 }{
3987 \clearpage
3988 \@mainmattertrue
3989 \pagenumbering{arabic}
3990 }
3991 }

```

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

```

3992 \newenvironment{backmatter}{
3993 \__document_structure_orig_backmatter
3994 }{
3995 \cs_if_exist:NTF\mainmatter{
3996 \mainmatter
3997 }{
3998 \clearpage
3999 \@mainmattertrue
4000 \pagenumbering{arabic}
4001 }
4002 }

```

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

```

4003 \@mainmattertrue\pagenumbering{arabic}

```

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

```

4004 \newcommand\afterprematurestop{}
4005 \def\prematurestop@endomgroup{
4006 \int_compare:nNf \l_document_structure_omgroup_level_int = 0 {
4007 \end{omgroup}
4008 \int_decr:N \l_document_structure_omgroup_level_int
4009 \prematurestop@endomgroup
4010 }
4011 }
4012 \providecommand\prematurestop{

```

```

4013 \message{Stopping sTeX processing prematurely}
4014 \prematuarestop@endomgroup
4015 \afterprematuarestop
4016 \end{document}
4017 }

```

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

## 31.8 Global Variables

**\setSGvar** set a global variable

```

4018 \RequirePackage{etoolbox}
4019 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

**\useSGvar** use a global variable

```

4020 \newrobustcmd\useSGvar[1]{%
4021 \@ifundefined{sTeX@Gvar@#1}
4022 {\PackageError{omdoc}
4023 {The sTeX Global variable #1 is undefined}
4024 {set it with \protect\setSGvar}}
4025 \@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.

```

4026 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
4027 \@ifundefined{sTeX@Gvar@#1}
4028 {\PackageError{omdoc}
4029 {The sTeX Global variable #1 is undefined}
4030 {set it with \protect\setSGvar}}
4031 {\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.

```
4032 \*cls)
4033 \@@=mikoslides)
4034 \ProvidesExplClass{mikoslides}{2020/12/06}{1.3}{MiKo slides Class}
4035 \RequirePackage{l3keys2e,expl-keystr-compatible}
4036
4037 \keys_define:nn{mikoslides / cls}{
4038   class .code:n = {
4039     \PassOptionsToClass{\CurrentOption}{omdoc}
4040     \str_if_eq:nnT{#1}{book}{
4041       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
4042     }
4043     \str_if_eq:nnT{#1}{report}{
4044       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
4045     }
4046   },
4047   notes .bool_set:N = \c__mikoslides_notes_bool ,
4048   slides .code:n = { \bool_set_false:N \c__mikoslides_notes_bool },
4049   unknown .code:n = {
4050     \PassOptionsToClass{\CurrentOption}{omdoc}
4051     \PassOptionsToClass{\CurrentOption}{beamer}
4052     \PassOptionsToPackage{\CurrentOption}{mikoslides}
4053   }
4054 }
4055 \ProcessKeysOptions{ mikoslides / cls }
4056 \bool_if:NTF \c__mikoslides_notes_bool {
4057   \PassOptionsToPackage{notes=true}{mikoslides}
4058 }{
4059   \PassOptionsToPackage{notes=false}{mikoslides}
4060 }
4061 \</cls)
```



now we do the same for the mikoslides package.

```

4062 <*package>
4063 \ProvidesExplPackage{mikoslides}{2020/12/06}{1.3}{MiKo slides Package}
4064 \RequirePackage{l3keys2e,expl-keystr-compat}
4065
4066 \keys_define:nn{mikoslides / pkg}{
4067   topsect      .str_set_x:N = \c__mikoslides_topsect_str,
4068   defaulttopsect .str_set_x:N = \c__mikoslides_defaulttopsec_str,
4069   notes        .bool_set:N = \c__mikoslides_notes_bool ,
4070   slides       .code:n      = { \bool_set_false:N \c__mikoslides_notes_bool },
4071   sectocframes .bool_set:N = \c__mikoslides_sectocframes_bool ,
4072   frameimages  .bool_set:N = \c__mikoslides_frameimages_bool ,
4073   fiboxed      .bool_set:N = \c__mikoslides_fiboxed_bool ,
4074   nopproblems  .bool_set:N = \c__mikoslides_nopproblems_bool,
4075   unknown      .code:n      = {
4076     \PassOptionsToClass{\CurrentOption}{stex}
4077     \PassOptionsToClass{\CurrentOption}{tikzinput}
4078   }
4079 }
4080 \ProcessKeysOptions{ mikoslides / pkg }
4081 \newif\ifnotes
4082 \bool_if:NTF \c__mikoslides_notes_bool {
4083   \notesttrue
4084 }{
4085   \notesfalse
4086 }
4087

```

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

```

4088 \str_if_empty:NTF \c__mikoslides_topsect_str {
4089   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_defaulttopsec_str
4090 }{
4091   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_topsect_str
4092 }
4093 </package>

```

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

```

4094 <*cls>
4095 \bool_if:NTF \c__mikoslides_notes_bool {
4096   \LoadClass{omdoc}
4097 }{
4098   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
4099   \newcounter{Item}
4100   \newcounter{paragraph}
4101   \newcounter{subparagraph}
4102   \newcounter{Hfootnote}
4103   \RequirePackage{omdoc}
4104 }

```

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

```

4105 \RequirePackage{mikoslides}
4106 </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.

```

4107 <*package>
4108 \RequirePackage{stex-compatibility}
4109 \RequirePackage{stex-tikzinput}
4110 \bool_if:NT \c__mikoslides_notes_bool {
4111   \RequirePackage{a4wide}
4112   \RequirePackage{marginnote}
4113   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
4114   \RequirePackage{mdframed}
4115   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
4116   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
4117 }
4118 \RequirePackage{etoolbox}
4119 \RequirePackage{amssymb}
4120 \RequirePackage{amsmath}
4121 \RequirePackage{comment}
4122 \RequirePackage{textcomp}
4123 \RequirePackage{url}
4124 \RequirePackage{graphicx}
4125 \RequirePackage{pgf}

```

## 32.2 Notes and Slides

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

```

4126 \bool_if:NT \c__mikoslides_notes_bool {
4127   \renewcommand\usetheme[2] [] {\usepackage[#1]{beamernotestheme#2}}
4128 }

```

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

```

4129 \newcounter{slide}
4130 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
4131 \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.

```

4132 \bool_if:NTF \c__mikoslides_notes_bool {
4133   \renewenvironment{note}{\ignorespaces}{\}
4134 }{
4135   \excludecomment{note}
4136 }

```

---

<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.

```
4137 \bool_if:NT \c__mikoslides_notes_bool {
4138   \newlength{\slideframewidth}
4139   \setlength{\slideframewidth}{1.5pt}
```

**frame** We first define the keys.

```
4140 \cs_new_protected:Nn \__mikoslides_do_yes_param:Nn {
4141   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
4142     \bool_set_true:N #1
4143   }{
4144     \bool_set_false:N #1
4145   }
4146 }
4147 \keys_define:nn{mikoslides / frame}{
4148   label .str_set_x:N = \l__mikoslides_frame_label_str,
4149   allowframebreaks .code:n = {
4150     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowframebreaks_bool { #1 }
4151   },
4152   allowdisplaybreaks .code:n = {
4153     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowdisplaybreaks_bool { #1 }
4154   },
4155   fragile .code:n = {
4156     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_fragile_bool { #1 }
4157   },
4158   shrink .code:n = {
4159     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_shrink_bool { #1 }
4160   },
4161   squeeze .code:n = {
4162     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_squeeze_bool { #1 }
4163   },
4164   t .code:n = {
4165     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_t_bool { #1 }
4166   },
4167 }
4168 \cs_new_protected:Nn \__mikoslides_frame_args:n {
4169   \str_clear:N \l__mikoslides_frame_label_str
4170   \bool_set_true:N \l__mikoslides_frame_allowframebreaks_bool
4171   \bool_set_true:N \l__mikoslides_frame_allowdisplaybreaks_bool
4172   \bool_set_true:N \l__mikoslides_frame_fragile_bool
4173   \bool_set_true:N \l__mikoslides_frame_shrink_bool
4174   \bool_set_true:N \l__mikoslides_frame_squeeze_bool
4175   \bool_set_true:N \l__mikoslides_frame_t_bool
4176   \keys_set:nn { mikoslides / frame }{ #1 }
4177 }
```

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

```
4178 \renewenvironment{frame}[1][]{
4179   \__mikoslides_frame_args:n{#1}
4180   \sffamily
4181   \stepcounter{slide}
4182   \def\@currentlabel{\theslide}
4183   \str_if_empty:NF \l__mikoslides_frame_label_str {
4184     \label{\l__mikoslides_frame_label_str}
```

4185 }  
 4186

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

4186 \def\itemize@level{outer}  
 4187 \def\itemize@outer{outer}  
 4188 \def\itemize@inner{inner}  
 4189 \renewcommand\newpage{\addtocounter{framenum}{1}}  
 4190 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}  
 4191 \renewenvironment{itemize}{  
 4192 \ifx\itemize@level\itemize@outer  
 4193 \def\itemize@label{\rhd\$}  
 4194 \fi  
 4195 \ifx\itemize@level\itemize@inner  
 4196 \def\itemize@label{\$\scriptstyle\rhd\$}  
 4197 \fi  
 4198 \begin{list}  
 4199 {\itemize@label}  
 4200 {\setlength{\labelsep}{.3em}  
 4201 \setlength{\labelwidth}{.5em}  
 4202 \setlength{\leftmargin}{1.5em}  
 4203 }  
 4204 \edef\itemize@level{\itemize@inner}  
 4205 }{  
 4206 \end{list}  
 4207 }

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

4208 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth=100pt]  
 4209 }{  
 4210 \medskip\miko@slidelabel\end{mdframed}  
 4211 }

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

\frametitle

4212 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip  
 4213 }

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

EdN:18

\pause 18

4214 \bool\_if:NT \c\_\_mikoslides\_notes\_bool {  
 4215 \newcommand\pause{  
 4216 }  
 4217 }

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

nomtext

4217 \bool\_if:NTF \c\_\_mikoslides\_notes\_bool {  
 4218 \newenvironment{nomtext}[1][\begin{omtext}[#1]}{\end{omtext}}  
 4219 }{  
 4220 \excludecomment{nomtext}  
 4221 }  
 4222 }

---

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

nomgroup

```
4222 \bool_if:NTF \c__mikoslides_notes_bool {
4223   \newenvironment{nomgroup}[2] [] {\begin{omgroup}[#1]{#2}}{\end{omgroup}}
4224 }{
4225   \excludecomment{nomgroup}
4226 }
```

ndefinition

```
4227 \bool_if:NTF \c__mikoslides_notes_bool {
4228   \newenvironment{ndefinition}[1] [] {\begin{definition}[#1]}{\end{definition}}
4229 }{
4230   \excludecomment{ndefinition}
4231 }
```

nassertion

```
4232 \bool_if:NTF \c__mikoslides_notes_bool {
4233   \newenvironment{nassertion}[1] [] {\begin{assertion}[#1]}{\end{assertion}}
4234 }{
4235   \excludecomment{nassertion}
4236 }
```

nsproof

```
4237 \bool_if:NTF \c__mikoslides_notes_bool {
4238   \newenvironment{nsproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
4239 }{
4240   \excludecomment{nsproof}
4241 }
```

nexample

```
4242 \bool_if:NTF \c__mikoslides_notes_bool {
4243   \newenvironment{nexample}[1] [] {\begin{example}[#1]}{\end{example}}
4244 }{
4245   \excludecomment{nexample}
4246 }
```

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

```
4247 \def\inputref@preskip{\smallskip}
4248 \def\inputref@postskip{\medskip}
```

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

\inputref\*

```
4249 \let\orig@inputref\inputref
4250 \def\inputref{@ifstar\ninputref\orig@inputref}
4251 \newcommand\ninputref[2] [] {
4252   \bool_if:NT \c__mikoslides_notes_bool {
4253     \orig@inputref[#1]{#2}
4254   }
4255 }
```

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

```

4256 \newlength{\slidelogoheight}
4257
4258 \bool_if:NTF \c__mikoslides_notes_bool {
4259   \setlength{\slidelogoheight}{.4cm}
4260 }{
4261   \setlength{\slidelogoheight}{1cm}
4262 }
4263 \newsavebox{\slidelogo}
4264 \sbox{\slidelogo}{\text{\TeX}}
4265 \newrobustcmd{\setslidelogo}[1]{
4266   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
4267 }
```

(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.

```

4268 \def\source{Michael Kohlhase}% customize locally
4269 \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.

```

4270 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
4271 \newsavebox{\cclogo}
4272 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
4273 \newif\ifcchref\cchreffalse
4274 \AtBeginDocument{
4275   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
4276 }
4277 \def\licensing{
4278   \ifcchref
4279     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
4280   \else
4281     {\usebox{\cclogo}}
4282   \fi
4283 }
4284 \newrobustcmd{\setlicensing}[2][]{
4285   \def@url{#1}
4286   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
4287   \ifx@url@empty
4288     \def\licensing{{\usebox{\cclogo}}}
4289   \else
4290     \def\licensing{
```

```

4291     \ifcchref
4292     \href{#1}{\usebox{\cclogo}}
4293     \else
4294     {\usebox{\cclogo}}
4295     \fi
4296   }
4297 \fi
4298 }

```

(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>

```

4299 \newrobustcmd\miko@slidelabel{
4300   \vbox to \slidelogoheight{
4301     \vss\hbox to \slidewidth
4302     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
4303   }
4304 }

```

(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.

```

4305 \def\Gin@mhrepos{}
4306 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
4307 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
4308 \newrobustcmd\frameimage[2][]{
4309   \stepcounter{slide}
4310   \bool_if:NT \c__mikoslides_frameimages_bool {
4311     \def\Gin@ewidth{}\setkeys{Gin}{#1}
4312     \bool_if:NF \c__mikoslides_notes_bool { \vfill }
4313     \begin{center}
4314       \bool_if:NTF \c__mikoslides_fiboxed_bool {
4315         \fbox{
4316           \ifx\Gin@ewidth\@empty
4317             \ifx\Gin@mhrepos\@empty
4318               \mhgraphics[width=\slidewidth,#1]{#2}
4319             \else
4320               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
4321             \fi
4322           \else% Gin@ewidth empty
4323             \ifx\Gin@mhrepos\@empty
4324               \mhgraphics[#1]{#2}
4325             \else
4326               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
4327             \fi
4328           \fi% Gin@ewidth empty
4329         }
4330       }{
4331         \ifx\Gin@ewidth\@empty

```

---

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

```

4332         \ifx\Gin@mhrepos\empty
4333             \mhgraphics[width=\slidewidth,#1]{#2}
4334         \else
4335             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
4336         \fi
4337         \ifx\Gin@mhrepos\empty
4338             \mhgraphics[#1]{#2}
4339         \else
4340             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
4341         \fi
4342     \fi% Gin@ewidth empty
4343 }
4344 \end{center}
4345 \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
4346 \bool_if:NF \c__mikoslides_notes_bool { \vfill }
4347 }
4348 } % 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.

```

4349 \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

```

4350 \AddToHook{begindocument}{
4351     \definecolor{green}{rgb}{0,.5,0}
4352     \definecolor{purple}{cmyk}{.3,1,0,.17}
4353 }

```

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`.

```

4354 % \def\STpresent#1{\textcolor{blue}{#1}}
4355 \def\defemph#1{\textcolor{magenta}{#1}}
4356 \def\symrefemph#1{\textcolor{cyan}{#1}}
4357 \def\compemph#1{\textcolor{blue}{#1}}
4358 \def\titleemph#1{\textcolor{blue}{#1}}
4359 \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.

```

4360 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
4361 \def\smalltextwarning{
4362     \pgfuseimage{miko@small@dbend}
4363     \xspace
4364 }
4365 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}

```



```

4366 \newrobustcmd\textwarning{
4367   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
4368   \xspace
4369 }
4370 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}
4371 \newrobustcmd\bigtextwarning{
4372   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
4373   \xspace
4374 }

```

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

```

4375 \newrobustcmd\putgraphicsat[3]{
4376   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
4377 }
4378 \newrobustcmd\putat[2]{
4379   \begin{picture}(0,0)\put(#1){#2}\end{picture}
4380 }

```

## 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`.

```

4381 \bool_if:NT \c__mikoslides_sectocframes_bool {
4382   \str_if_eq:VnTF \__mikoslidestopsect{part}{
4383     \newcounter{chapter}\counterwithin*{section}{chapter}
4384   }{
4385     \str_if_eq:VnT\__mikoslidestopsect{chapter}{
4386       \newcounter{chapter}\counterwithin*{section}{chapter}
4387     }
4388   }
4389 }

```

`\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
4390 \def\part@prefix{}
4391 \@ifpackageloaded{omdoc}{}{
4392   \str_case:VnF \__mikoslidestopsect {
4393     {part}{
4394       \int_set:Nn \l_document_structure_section_level_int {0}
4395       \def\thesection{\arabic{chapter}.\arabic{section}}
4396       \def\part@prefix{\arabic{chapter}.}
4397     }
4398     {chapter}{
4399       \int_set:Nn \l_document_structure_section_level_int {1}
4400       \def\thesection{\arabic{chapter}.\arabic{section}}
4401       \def\part@prefix{\arabic{chapter}.}
4402     }
4403   }{
4404     \int_set:Nn \l_document_structure_section_level_int {2}
4405     \def\part@prefix{}

```

```

4406 }
4407 }
4408
4409 \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

```

4410 \renewenvironment{omgroup}[2][]{
4411   \__document_structure_omgroup_args:n { #1 }
4412   \int_incr:N \l_document_structure_omgroup_level_int
4413   \int_incr:N \l_document_structure_section_level_int
4414   \bool_if:NT \c__mikoslides_sectocframes_bool {
4415     \stepcounter{slide}
4416     \begin{frame}[noframenumbering]
4417     \vfill\Large\centering
4418     \red{
4419       \ifcase\l_document_structure_section_level_int\or
4420         \stepcounter{part}
4421         \def\__mikoslideslabel{\omdoc@part@kw~\Roman{part}}
4422         \def\currentsectionlevel{\omdoc@part@kw}
4423       \or
4424         \stepcounter{chapter}
4425         \def\__mikoslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
4426         \def\currentsectionlevel{\omdoc@chapter@kw}
4427       \or
4428         \stepcounter{section}
4429         \def\__mikoslideslabel{\part@prefix\arabic{section}}
4430         \def\currentsectionlevel{\omdoc@section@kw}
4431       \or
4432         \stepcounter{subsection}
4433         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
4434         \def\currentsectionlevel{\omdoc@subsection@kw}
4435       \or
4436         \stepcounter{subsubsection}
4437         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
4438         \def\currentsectionlevel{\omdoc@subsubsection@kw}
4439       \or
4440         \stepcounter{mparagraph}
4441         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{msubsection}.\arabic{mparagraph}}
4442         \def\currentsectionlevel{\omdoc@paragraph@kw}
4443       \fi% end ifcase
4444       \__mikoslideslabel%\sref@label@id\__mikoslideslabel
4445       \quad #2%
4446     }%
4447     \vfill%
4448     \end{frame}%
4449   }
4450   \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str%
4451 }{}
4452 }

```

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

```

4453 \def\inserttheorembodyfont{\normalfont}
4454 \bool_if:NF \c__mikoslides_notes_bool {
4455   \defbeamertemplate{theorem begin}{miko}
4456   {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
4457     \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
4458     \inserttheorempunctuation\inserttheorembodyfont\space}
4459   \defbeamertemplate{theorem end}{miko}{}
```

and we set it as the default one.

```

4460   \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.

```

4461   \expandafter\def\csname Parent2\endcsname{}
4462 }
4463 \bool_if:NT \c__mikoslides_notes_bool {
4464   \renewenvironment{columns}[1][]{%
4465     \par\noindent%
4466     \begin{minipage}%
4467       \slidewidth\centering\leavevmode%
4468   }{%
4469     \end{minipage}\par\noindent%
4470   }%
4471   \newsavebox\columnbox%
4472   \renewenvironment<>{column}[2][]{%
4473     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
4474   }{%
4475     \end{minipage}\end{lrbox}\usebox\columnbox%
4476   }%
4477 }
4478 \bool_if:NTF \c__mikoslides_noproblems_bool {
4479   \newenvironment{problems}{}{}
4480 }{
4481   \excludecomment{problems}
4482 }
```

## 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.

```

4483 \gdef\printexcursions{}
4484 \newcommand\excursionref[2]{% label, text
4485   \bool_if:NT \c__mikoslides_notes_bool {
4486     \begin{omtext}[title=Excursion]
4487       #2 \sref[fallback=the appendix]{#1}.
4488     \end{omtext}
4489   }
4490 }
4491 \newcommand\activate@excursion[2][{}{
4492   \gappto\printexcursions{\inputref[#1]{#2}}
```

```

4493 }
4494 \newcommand\excursion[4][{}]{% repos, label, path, text
4495   \bool_if:NT \c__mikoslides_notes_bool {
4496     \activate@excursion[#1]{#3}\excursionref{#2}{#4}
4497   }
4498 }

```

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

## \excursiongroup

```

4499 \keys_define:nn{mikoslides / excursiongroup }{
4500   id          .str_set_x:N = \l__mikoslides_excursion_id_str,
4501   intro       .tl_set:N   = \l__mikoslides_excursion_intro_tl,
4502   mhrepos     .str_set_x:N = \l__mikoslides_excursion_mhrepos_str
4503 }
4504 \cs_new_protected:Nn \__mikoslides_excursion_args:n {
4505   \tl_clear:N \l__mikoslides_excursion_intro_tl
4506   \str_clear:N \l__mikoslides_excursion_id_str
4507   \str_clear:N \l__mikoslides_excursion_mhrepos_str
4508   \keys_set:nn {mikoslides / excursiongroup }{ #1 }
4509 }
4510 \newcommand\excursiongroup[1][{}]{
4511   \__mikoslides_excursion_args:n{ #1 }
4512   \ifdefempty\printexcursions{}% only if there are excursions
4513   {\begin{note}
4514     \begin{omgroup}[#1]{Excursions}%
4515     \ifdefempty\l__mikoslides_excursion_intro_tl{{
4516       \inputref[\l__mikoslides_excursion_mhrepos_str]{
4517         \l__mikoslides_excursion_intro_tl
4518       }
4519     }
4520     \printexcursions%
4521     \end{omgroup}
4522     \end{note}}
4523   }
4524 }

```

(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.

```
4525 <*package>
4526 <@@=problems>
4527 \ProvidesExplPackage{problem}{2019/03/20}{1.3}{Semantic Markup for Problems}
4528 \RequirePackage{l3keys2e,expl-keystr-compatible}
4529
4530 \keys_define:nn { problem / pkg }{
4531   notes      .default:n    = { true },
4532   notes      .bool_set:N   = \c__problems_notes_bool,
4533   gnotes     .default:n    = { true },
4534   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
4535   hints      .default:n    = { true },
4536   hints      .bool_set:N   = \c__problems_hints_bool,
4537   solutions  .default:n    = { true },
4538   solutions  .bool_set:N   = \c__problems_solutions_bool,
4539   pts        .default:n    = { true },
4540   pts        .bool_set:N   = \c__problems_pts_bool,
4541   min        .default:n    = { true },
4542   min        .bool_set:N   = \c__problems_min_bool,
4543   boxed      .default:n    = { true },
4544   boxed      .bool_set:N   = \c__problems_boxed_bool,
4545   unknown    .code:n       = {}
4546 }
4547 \def\solutionstrue{
4548   \bool_set_true:N \c__problems_solutions_bool
4549 }
4550 \def\solutionsfalse{
4551   \bool_set_false:N \c__problems_solutions_bool
4552 }
4553
4554 \ProcessKeysOptions{ problem / pkg }
```

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

```

4555 \RequirePackage{stex-compatibility}
4556 \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.

```

4557 \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.

```

4558 \def\prob@problem@kw{Problem}
4559 \def\prob@solution@kw{Solution}
4560 \def\prob@hint@kw{Hint}
4561 \def\prob@note@kw{Note}
4562 \def\prob@gnote@kw{Grading}
4563 \def\prob@pt@kw{pt}
4564 \def\prob@min@kw{min}

```

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

For the other languages, we set up triggers

```

4565 \@ifpackageloaded{babel}{
4566   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4567   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4568     \input{problem-ngerman.ldf}
4569   }
4570   \clist_if_in:NnT \l_tmpa_clist {finnish}{
4571     \input{problem-finnish.ldf}
4572   }
4573   \clist_if_in:NnT \l_tmpa_clist {french}{
4574     \input{problem-french.ldf}
4575   }
4576   \clist_if_in:NnT \l_tmpa_clist {russian}{
4577     \input{problem-russian.ldf}
4578   }
4579 }{}

```

## 33.2 Problems and Solutions

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

```

4580 \keys_define:nn{ problem / problem }{
4581   id      .str_set:x:N = \l__problems_prob_id_str,
4582   pts     .tl_set:N    = \l__problems_prob_pts_tl,
4583   min     .tl_set:N    = \l__problems_prob_min_tl,
4584   title   .tl_set:N    = \l__problems_prob_title_tl,
4585   refnum  .int_set:N    = \l__problems_prob_refnum_int
4586 }
4587 \cs_new_protected:Nn \__problems_prob_args:n {
4588   \str_clear:N \l__problems_prob_id_str
4589   \tl_clear:N \l__problems_prob_pts_tl
4590   \tl_clear:N \l__problems_prob_min_tl
4591   \tl_clear:N \l__problems_prob_title_tl

```

```

4592 \int_zero_new:N \l__problems_prob_refnum_int
4593 \keys_set:nn { problem / problem }{ #1 }
4594 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
4595   \let\l__problems_inclprob_refnum_int\undefined
4596 }
4597 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

4598 \newcounter{problem}
4599 \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.

```

4600 \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

```

4601 \newcommand\prob@number{
4602   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
4603     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
4604   }{
4605     \int_if_exist:NTF \l__problems_prob_refnum_int {
4606       \prob@label{\int_use:N \l__problems_prob_refnum_int }
4607     }{
4608       \prob@label\theproblem
4609     }
4610   }
4611 }

```

(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.

```

4612 \newcommand\prob@title[3]{%
4613   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
4614     #2 \l__problems_inclprob_title_tl #3
4615   }{
4616     \tl_if_exist:NTF \l__problems_prob_title_tl {
4617       #2 \l__problems_prob_title_tl #3
4618     }{
4619       #1
4620     }
4621   }
4622 }

```

(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.

```

4623 \def\prob@heading{
4624   \prob@problem@kw~\prob@number\prob@title{~}{~}{~}\strut}
4625   %\sref@label{id{\prob@problem@kw~\prob@number}}{~}
4626 }

```

(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`

```

4627 \newenvironment{problem}[1][1]{
4628   \__problems_prob_args:n{#1}%\sref@target%
4629   \@in@omtexttrue% we are in a statement (for inline definitions)
4630   \stepcounter{problem}\record@problem
4631   \def\current@section@level{\prob@problem@kw}
4632   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
4633 }%
4634 {\smallskip}
4635 \bool_if:NT \c__problems_boxed_bool {
4636   \surroundwithmdframed{problem}
4637 }

```

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

```

4638 \def\record@problem{
4639   \protected@write\@auxout{}
4640   {
4641     \string\@problem{\prob@number}
4642     {
4643       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
4644         \l__problems_inclprob_pts_tl
4645       }{
4646         \l__problems_prob_pts_tl
4647       }
4648     }%
4649     {
4650       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
4651         \l__problems_inclprob_min_tl
4652       }{
4653         \l__problems_prob_min_tl
4654       }
4655     }
4656   }
4657 }

```

(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).

```

4658 \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.

```

4659 \keys_define:nn { problem / solution }{
4660   id          .str_set_x:N = \l__problems_solution_id_str ,
4661   for         .tl_set:N    = \l__problems_solution_for_tl ,
4662   height      .dim_set:N   = \l__problems_solution_height_dim ,
4663   creators    .clist_set:N = \l__problems_solution_creators_clist ,
4664   contributors .clist_set:N = \l__problems_solution_contributors_clist ,
4665   srccite     .tl_set:N    = \l__problems_solution_srccite_tl
4666 }
4667 \cs_new_protected:Nn \__problems_solution_args:n {
4668   \str_clear:N \l__problems_solution_id_str
4669   \tl_clear:N \l__problems_solution_for_tl
4670   \tl_clear:N \l__problems_solution_srccite_tl
4671   \clist_clear:N \l__problems_solution_creators_clist
4672   \clist_clear:N \l__problems_solution_contributors_clist
4673   \dim_zero:N \l__problems_solution_height_dim
4674   \keys_set:nn { problem / solution }{ #1 }
4675 }

```

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

```

4676 \newcommand\@startsolution[1][ ]{
4677   \__problems_solution_args:n { #1 }
4678   \@in@omtexttrue% we are in a statement.
4679   \bool_if:NF \c__problems_boxed_bool { \hrule }
4680   \smallskip\noindent
4681   {\textbf\prob@solution@kw : \enspace}
4682   \begin{small}
4683   \def\current@section@level{\prob@solution@kw}
4684   \ignorespacesandpars
4685 }

```

**\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.

```

4686 \newcommand\startsolutions{
4687   \specialcomment{solution}{\@startsolution}{
4688     \bool_if:NF \c__problems_boxed_bool {
4689       \hrule\medskip
4690     }
4691     \end{small}%
4692   }
4693   \bool_if:NT \c__problems_boxed_bool {
4694     \surroundwithmdframed{solution}
4695   }
4696 }

```

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

**\stopsolutions**

```

4697 \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.

```

4698 \bool_if:NTF \c__problems_solutions_bool {
4699   \startsolutions
4700 }{
4701   \stopsolutions
4702 }

```

**exnote**

```

4703 \bool_if:NTF \c__problems_notes_bool {
4704   \newenvironment{exnote}[1][]{
4705     \par\smallskip\hrule\smallskip
4706     \noindent\textbf{\prob@note@kw : }\small
4707   }{
4708     \smallskip\hrule
4709   }
4710 }{
4711   \excludecomment{exnote}
4712 }

```

**hint**

```

4713 \bool_if:NTF \c__problems_notes_bool {
4714   \newenvironment{hint}[1][]{
4715     \par\smallskip\hrule\smallskip
4716     \noindent\textbf{\prob@hint@kw :~ }\small
4717   }{
4718     \smallskip\hrule
4719   }
4720   \newenvironment{exhint}[1][]{
4721     \par\smallskip\hrule\smallskip
4722     \noindent\textbf{\prob@hint@kw :~ }\small
4723   }{
4724     \smallskip\hrule
4725   }
4726 }{
4727   \excludecomment{hint}
4728   \excludecomment{exhint}
4729 }

```

**gnote**

```

4730 \bool_if:NTF \c__problems_notes_bool {
4731   \newenvironment{gnote}[1][]{
4732     \par\smallskip\hrule\smallskip
4733     \noindent\textbf{\prob@gnote@kw : }\small
4734   }{
4735     \smallskip\hrule
4736   }
4737 }{
4738   \excludecomment{gnote}
4739 }

```

### 33.3 Multiple Choice Blocks

```

4740 \newenvironment{mcb}{
4741   \begin{enumerate}
4742 }{
4743   \end{enumerate}
4744 }

```

we define the keys for the mcc macro

```

4745 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
4746   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
4747     \bool_set_true:N #1
4748   }{
4749     \bool_set_false:N #1
4750   }
4751 }
4752 \keys_define:nn { problem / mcc }{
4753   id          .str_set:x:N = \l__problems_mcc_id_str ,
4754   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
4755   T           .default:n    = { true } ,
4756   T           .bool_set:N   = \l__problems_mcc_t_bool ,
4757   F           .default:n    = { true } ,
4758   F           .bool_set:N   = \l__problems_mcc_f_bool ,
4759   Ttext       .code:n       = {
4760     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
4761   } ,
4762   Ftext       .code:n       = {
4763     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
4764   }
4765 }
4766 \cs_new_protected:Nn \l__problems_mcc_args:n {
4767   \str_clear:N \l__problems_mcc_id_str
4768   \tl_clear:N \l__problems_mcc_feedback_tl
4769   \bool_set_true:N \l__problems_mcc_t_bool
4770   \bool_set_true:N \l__problems_mcc_f_bool
4771   \bool_set_true:N \l__problems_mcc_Ttext_bool
4772   \bool_set_false:N \l__problems_mcc_Ftext_bool
4773   \keys_set:nn { problem / mcc }{ #1 }
4774 }

```

\mcc

```

4775 \newcommand\mcc[2][]{
4776   \l__problems_mcc_args:n{ #1 }
4777   \item #2
4778   \bool_if:NT \c__problems_solutions_bool {
4779     \\\
4780     \bool_if:NT \l__problems_mcc_t_bool {
4781       % TODO!
4782       % \ifcsstring{mcc@T}{T}{\mcc@Ttext}%
4783     }
4784     \bool_if:NT \l__problems_mcc_f_bool {

```

---

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

```

4785      % TODO!
4786      % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
4787    }
4788    \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
4789      !
4790    }{
4791      \l__problems_mcc_feedback_tl
4792    }
4793  }
4794 } %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.

```

4795
4796 \keys_define:nn{ problem / inclproblem }{
4797   % id      .str_set_x:N = \l__problems_inclprob_id_str,
4798   pts      .tl_set:N    = \l__problems_inclprob_pts_tl,
4799   min      .tl_set:N    = \l__problems_inclprob_min_tl,
4800   title    .tl_set:N    = \l__problems_inclprob_title_tl,
4801   refnum   .int_set:N    = \l__problems_inclprob_refnum_int,
4802   mhrepos  .str_set_x:N = \l__problems_inclprob_mhrepos_str
4803 }
4804 \cs_new_protected:Nn \l__problems_inclprob_args:n {
4805   % \str_clear:N \l__problems_prob_id_str
4806   \tl_clear:N \l__problems_inclprob_pts_tl
4807   \tl_clear:N \l__problems_inclprob_min_tl
4808   \tl_clear:N \l__problems_inclprob_title_tl
4809   \int_zero_new:N \l__problems_inclprob_refnum_int
4810   \str_clear:N \l__problems_inclprob_mhrepos_str
4811   \keys_set:nn { problem / inclproblem }{ #1 }
4812   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
4813     \let\l__problems_inclprob_pts_tl\undefined
4814   }
4815   \tl_if_empty:NT \l__problems_inclprob_min_tl {
4816     \let\l__problems_inclprob_min_tl\undefined
4817   }
4818   \tl_if_empty:NT \l__problems_inclprob_title_tl {
4819     \let\l__problems_inclprob_title_tl\undefined
4820   }
4821   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
4822     \let\l__problems_inclprob_refnum_int\undefined
4823   }
4824 }
4825
4826 \cs_new_protected:Nn \l__problems_inclprob_clear: {
4827   % \str_clear:N \l__problems_prob_id_str
4828   \let\l__problems_inclprob_pts_tl\undefined
4829   \let\l__problems_inclprob_min_tl\undefined

```

```

4830 \let\l__problems_inclprob_title_tl\undefined
4831 \let\l__problems_inclprob_refnum_int\undefined
4832 \let\l__problems_inclprob_mhrepos_str\undefined
4833 }
4834
4835 \newcommand\includeproblem[2][ ]{
4836   \__problems_inclprob_args:n{ #1 }
4837   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
4838     \input{#2}
4839   }{
4840     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
4841       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
4842     }
4843   }
4844   \__problems_inclprob_clear:
4845 }

```

(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.

```

4846 \AddToHook{enddocument}{
4847   \bool_if:NT \c__problems_pts_bool {
4848     \message{Total:~\arabic{pts}~points}
4849   }
4850   \bool_if:NT \c__problems_min_bool {
4851     \message{Total:~\arabic{min}~minutes}
4852   }
4853 }

```

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

```

4854 \def\pts#1{
4855   \bool_if:NT \c__problems_pts_bool {
4856     \marginpar{#1~\prob@pt@kw}
4857   }
4858 }
4859 \def\min#1{
4860   \bool_if:NT \c__problems_min_bool {
4861     \marginpar{#1~\prob@min@kw}
4862   }
4863 }

```

**\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.

```

4864 \newcounter{pts}
4865 \def\show@pts{
4866   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
4867     \bool_if:NT \c__problems_pts_bool {
4868       \marginpar{\l__problems_inclprob_pts_tl;\prob@pt@kw\smallskip}
4869       \addtocounter{pts}{\l__problems_inclprob_pts_tl}

```

```

4870     }
4871   }{
4872     \tl_if_exist:NT \l__problems_prob_pts_tl {
4873       \bool_if:NT \c__problems_pts_bool {
4874         \marginpar{\l__problems_prob_pts_tl;\prob@pt@kw\smallskip}
4875         \addtocounter{pts}{\l__problems_prob_pts_tl}
4876       }
4877     }
4878   }
4879 }

```

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

\show@min

```

4880 \newcounter{min}
4881 \def\show@min{
4882   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
4883     \bool_if:NT \c__problems_min_bool {
4884       \marginpar{\l__problems_inclprob_pts_tl;min}
4885       \addtocounter{min}{\l__problems_inclprob_min_tl}
4886     }
4887   }{
4888     \tl_if_exist:NT \l__problems_prob_min_tl {
4889       \bool_if:NT \c__problems_min_bool {
4890         \marginpar{\l__problems_prob_min_tl;min}
4891         \addtocounter{min}{\l__problems_prob_min_tl}
4892       }
4893     }
4894   }
4895 }
4896 \</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.

```
4897 <@@=hwexam>
4898 <*cls>
4899 \ProvidesExplClass{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
4900 \RequirePackage{l3keys2e,expl-keystr-compatible}
4901 \DeclareOption*{
4902   \PassOptionsToClass{\CurrentOption}{omdoc}
4903   \PassOptionsToPackage{\CurrentOption}{stex}
4904   \PassOptionsToPackage{\CurrentOption}{hwexam}
4905   \PassOptionsToPackage{\CurrentOption}{tikzinput}
4906 }
4907 \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.

```
4908 \LoadClass{omdoc}
4909 \RequirePackage{stex}
4910 \RequirePackage{hwexam}
4911 \RequirePackage{tikzinput}
4912 \RequirePackage{graphicx}
4913 \RequirePackage{a4wide}
4914 \RequirePackage{amssymb}
4915 \RequirePackage{amstext}
4916 \RequirePackage{amsmath}
```

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

```

4917 \newcommand\assig@default@type{\hwexam@assignment@kw}
4918 \def\document@hwexamtype{\assig@default@type}
4919 <@@=document_structure>
4920 \keys_define:nn { document-structure / document }{
4921 id .str_set_x:N = \c_document_structure_document_id_str,
4922 hwexamtype .tl_set:N = \document@hwexamtype
4923 }
4924 <@@=hwexam>
4925 </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.

```
4926 \*package>
4927 \ProvidesExplPackage{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
4928 \RequirePackage{l3keys2e,expl-keystr-compat}
4929
4930 \newif\iftest\testfalse
4931 \DeclareOption{test}{\testtrue}
4932 \newif\ifmultiple\multiplefalse
4933 \DeclareOption{multiple}{\multipletrue}
4934 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
4935 \ProcessOptions
```

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

```
4936 \RequirePackage{keyval}[1997/11/10]
4937 \RequirePackage{problem}
```

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

```
4938 \newcommand\hwexam@assignment@kw{Assignment}
4939 \newcommand\hwexam@given@kw{Given}
4940 \newcommand\hwexam@due@kw{Due}
4941 \newcommand\hwexam@testemptypage@kw{This page was intentionally left blank for extra
4942 space}%
4943 \newcommand\correction@probs@kw{prob.}%
4944 \newcommand\correction@pts@kw{total}%
4945 \newcommand\correction@reached@kw{reached}%
4946 \newcommand\correction@sum@kw{Sum}%
4947 \newcommand\correction@grade@kw{grade}%
4948 \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

```

4949 \ifpackageloaded{babel}{\RequirePackage[base]{babel}}
4950
4951 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4952 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4953   \input{hwexam-ngerman.ldf}
4954 }
4955 \clist_if_in:NnT \l_tmpa_clist {finnish}{
4956   \input{hwexam-finnish.ldf}
4957 }
4958 \clist_if_in:NnT \l_tmpa_clist {french}{
4959   \input{hwexam-french.ldf}
4960 }
4961 \clist_if_in:NnT \l_tmpa_clist {russian}{
4962   \input{hwexam-russian.ldf}
4963 }

```

## 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.

```

4964 \newcounter{assignment}
4965 \numberproblemsin{assignment}
4966 \renewcommand\prob@label[1]{\arabic{assignment}.#1}

```

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

```

4967 \keys_define:nn { hwexam / assignment } {
4968   id .str_set:N = \l__hwexam_assign_id_str,
4969   number .int_set:N = \l__hwexam_assign_number_int,
4970   title .tl_set:N = \l__hwexam_assign_title_tl,
4971   type .tl_set:N = \l__hwexam_assign_type_tl,
4972   given .tl_set:N = \l__hwexam_assign_given_tl,
4973   due .tl_set:N = \l__hwexam_assign_due_tl,
4974   loadmodules .code:n = {
4975     \bool_set_true:N \l__hwexam_assign_loadmodules_bool
4976   }
4977 }
4978 \cs_new_protected:Nn \__hwexam_assignment_args:n {
4979   \str_clear:N \l__hwexam_assign_id_str
4980   \int_set:Nn \l__hwexam_assign_number_int {-1}
4981   \tl_clear:N \l__hwexam_assign_title_tl
4982   \tl_clear:N \l__hwexam_assign_type_tl
4983   \tl_clear:N \l__hwexam_assign_given_tl
4984   \tl_clear:N \l__hwexam_assign_due_tl
4985   \bool_set_false:N \l__hwexam_assign_loadmodules_bool
4986   \keys_set:nn { hwexam / assignment }{ #1 }
4987 }

```

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.

```

4988 \newcommand\given@due[2]{
4989 \bool_lazy_all:nF {
4990 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
4991 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
4992 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
4993 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
4994 }{ #1 }
4995
4996 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
4997 \tl_if_empty:NF \l__hwexam_assign_given_tl {
4998 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
4999 }
5000 }{
5001 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
5002 }
5003
5004 \bool_lazy_or:nnF {
5005 \bool_lazy_and_p:nn {
5006 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
5007 }{
5008 \tl_if_empty_p:V \l__hwexam_assign_due_tl
5009 }
5010 }{
5011 \bool_lazy_and_p:nn {
5012 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
5013 }{
5014 \tl_if_empty_p:V \l__hwexam_assign_due_tl
5015 }
5016 }{ ,~ }
5017
5018 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
5019 \tl_if_empty:NF \l__hwexam_assign_due_tl {
5020 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
5021 }
5022 }{
5023 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
5024 }
5025
5026 \bool_lazy_all:nF {
5027 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
5028 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
5029 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
5030 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
5031 }{ #2 }
5032 }

```

`\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.

```

5033 \newcommand\assignment@title[3]{

```

```

5034 \tl_if_empty:NTF \l__hwexam_inclassassign_title_tl {
5035 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
5036 #1
5037 }{
5038 #2\l__hwexam_assign_title_tl#3
5039 }
5040 }{
5041 #2\l__hwexam_inclassassign_title_tl#3
5042 }
5043 }

```

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

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

```

5044 \newcommand\assignment@number{
5045 \int_compare:nNnTF \l__hwexam_inclassassign_number_int = {-1} {
5046 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
5047 \int_use:N \l__hwexam_assign_number_int
5048 }
5049 }{
5050 \int_use:N \l__hwexam_inclassassign_number_int
5051 }
5052 }

```

(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.

```

5053 \newenvironment{assignment}[1][]{
5054 \__hwexam_assignment_args:n { #1 }
5055 %\sref@target
5056 \let\__hwexamnum\l__hwexam_assign_number_int
5057 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
5058 \stepcounter{assignment}
5059 }{
5060 \setcounter{assignment}{\int_use:N\__hwexamnum}
5061 }
5062 \setcounter{problem}{0}
5063 \def\current@section@level{\document@hwexamtype}
5064 %\sref@label@id{\document@hwexamtype \thesection}
5065 \begin{@assignment}
5066 }{
5067 \end{@assignment}
5068 }

```

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

```

5069 \def\__hwexasstitle{
5070 \protect\document@hwexamtype~\arabic{assignment}
5071 \assignment@title{}\;{} \; -- \given@due{}{}
5072 }

```

```

5073 \ifmultiple
5074 \newenvironment{@assignment}{
5075 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
5076 \begin{omgroup}[loadmodules]{\__hwexasstitle}
5077 }{
5078 \begin{omgroup}{\__hwexasstitle}
5079 }
5080 }{
5081 \end{omgroup}
5082 }

```

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

```

5083 \else
5084 \newenvironment{@assignment}{
5085 \begin{center}\bf
5086 \Large\@title\strut\
5087 \document@hwexamtype~\arabic{assignment}\assignment@title{\;}{:};{\}\}
5088 \large\given@due{--\;}{\;}{--}
5089 \end{center}
5090 }{}
5091 \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.

```

5092 \keys_define:nn { hwexam / inclassignment } {
5093 %id .str_set_x:N = \l__hwexam_assign_id_str,
5094 number .int_set:N = \l__hwexam_inclassign_number_int,
5095 title .tl_set:N = \l__hwexam_inclassign_title_tl,
5096 type .tl_set:N = \l__hwexam_inclassign_type_tl,
5097 given .tl_set:N = \l__hwexam_inclassign_given_tl,
5098 due .tl_set:N = \l__hwexam_inclassign_due_tl,
5099 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
5100 }
5101 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
5102 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
5103 \tl_clear:N \l__hwexam_inclassign_title_tl
5104 \tl_clear:N \l__hwexam_inclassign_type_tl
5105 \tl_clear:N \l__hwexam_inclassign_given_tl
5106 \tl_clear:N \l__hwexam_inclassign_due_tl
5107 \str_clear:N \l__hwexam_inclassign_mhrepos_str
5108 \keys_set:nn { hwexam / inclassignment }{ #1 }
5109 }
5110 \__hwexam_inclassignment_args:n {}
5111
5112 \newcommand\inputassignment[2][ ]{
5113 \__hwexam_inclassignment_args:n { #1 }
5114 \str_if_empty:NTF \l__hwexam_inclassign_mhrepos_str {
5115 \input{#2}
5116 }{
5117 \stex_in_repository:nn{\l__hwexam_inclassign_mhrepos_str}{

```

```

5118 \input{\mhp{path}\l__hwexam_inclasssign_mhrepos_str}{#2}}
5119 }
5120 }
5121 \__hwexam_inclasssignment_args:n {}
5122 }
5123 \newcommand\includeassignment[2][ ]{
5124 \newpage
5125 \inputassignment[#1]{#2}
5126 }

```

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

## 35.4 Typesetting Exams

\quizheading

```

5127 \ExplSyntaxOff
5128 \newcommand\quizheading[1]{%
5129 \def\@tas{#1}%
5130 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
5131 \ifx\@tas\empty\else%
5132 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
5133 \fi%
5134 }
5135 \ExplSyntaxOn

```

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

\testheading

```

5136 \keys_define:nn { hwexam / testheading } {
5137 min .tl_set:N = \l__hwexam_testheading_min_tl,
5138 duration .tl_set:N = \l__hwexam_testheading_duration_tl,
5139 reqpts .tl_set:N = \l__hwexam_testheading_reqpts_tl
5140 }
5141 \cs_new_protected:Nn \__hwexam_testheading_args:n {
5142 \tl_clear:N \l__hwexam_testheading_min_tl
5143 \tl_clear:N \l__hwexam_testheading_duration_tl
5144 \tl_clear:N \l__hwexam_testheading_reqpts_tl
5145 \keys_set:nn { hwexam / testheading }{ #1 }
5146 }
5147 \newenvironment{testheading}[1][ ]{
5148 \__hwexam_testheading_args:n{ #1 }
5149 \noindent\large{Name:~\hfill
5150 Matriculation Number:\hspace*{2cm}\strut}\[1ex]
5151 \begin{center}
5152 \Large\textbf{\@title}\[1ex]
5153 \large\@date\[3ex]
5154 \end{center}
5155 \textbf{You~have~
5156 \tl_if_empty:NTF \l__hwexam_testheading_duration_tl {
5157 \l__hwexam_testheading_min_tl~minutes
5158 }{
5159 \l__hwexam_testheading_duration_tl
5160 }~

```

```

5161 (sharp)~for~the~test
5162 };\
5163 Write~the~solutions~to~the~sheet.
5164 \par\noindent
5165 \newcount\check@time\check@time=\l__hwexam_testheading_min_tl
5166 \advance\check@time by -\theassignment@totalmin
5167 The~estimated~time~for~solving~this~exam~is~
5168 {\theassignment@totalmin}~minutes,~
5169 leaving~you~{\the\check@time}~minutes~for~revising~
5170 your~exam.
5171
5172 \par\noindent
5173 \newcount\bonus@pts\bonus@pts=\theassignment@totalpts
5174 \advance\bonus@pts by -\l__hwexam_testheading_reqpts_tl
5175 You~can~reach~{\theassignment@totalpts}~points~if~you~
5176 solve~all~problems.~You~will~only~need~
5177 {\l__hwexam_testheading_reqpts_tl}~points~for~a~perfect~score,~
5178 i.e.~\ {\the\bonus@pts}~points~are~bonus~points.
5179 \vfill
5180 \begin{center}
5181 {
5182 \Large\em You~have~ample~time,~so~take~it~slow~
5183 and~avoid~rushing~to~mistakes!\}[2ex]
5184 Different~problems~test~different~skills~and~
5185 knowledge,~so~do~not~get~stuck~on~one~problem.
5186 }
5187 \vfill\par\resizebox{\textwidth}{!}{\correction@table}\}[3ex]
5188 \end{center}
5189 }{
5190 \newpage
5191 }

```

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

\testspace

```

5192 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

5193 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

5194 \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.

```

5195 <@=problems>
5196 \renewcommand\@problem[3]{
5197 \stepcounter{assignment@probs}
5198 \def\__problemspts{#2}

```

```

5199 \ifx\__problemspts\@empty\else
5200 \addtocounter{assignment@totalpts}{#2}
5201 \fi
5202 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
5203 \xdef\correction@probs{\correction@probs & #1}%
5204 \xdef\correction@pts{\correction@pts & #2}
5205 \xdef\correction@reached{\correction@reached & }
5206 }
5207 \@@=hwexam

```

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

\correction@table This macro generates the correction table

```

5208 \newcounter{assignment@probs}
5209 \newcounter{assignment@totalpts}
5210 \newcounter{assignment@totalmin}
5211 \def\correction@probs{\correction@probs@kw}%
5212 \def\correction@pts{\correction@pts@kw}%
5213 \def\correction@reached{\correction@reached@kw}%
5214 \def\after@correction@table{}%
5215 \stepcounter{assignment@probs}
5216 \newcommand\correction@table{
5217 \resizebox{\textwidth}{!}{%
5218 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
5219 &\multicolumn{\theassignment@probs}{c|}|%|
5220 {\footnotesize\correction@forgrading@kw} &\\ \hline
5221 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
5222 \correction@pts & \theassignment@totalpts & \\ \hline
5223 \correction@reached & & \[.7cm]\hline
5224 \end{tabular}}
5225 \ifx\after@correction@table\@empty\else\strut\par\noindent\after@correction@table\fi}
5226 \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\wierfont=../assignments/wierfont
\font\denkerfont=../assignments/denker
\font\uhrfont=../assignments/uhr
\font\warnschildfont=../assignments/achtung

```

```

\newcommand\wierfont{\font\wierfont\char65}
\newcommand\denkerfont{\font\denkerfont\char65}
\newcommand\uhrfont{\font\uhrfont\char65}
\newcommand\warnschildfont{\font\warnschildfont\char 65}
\newcommand\hardA{\warnschild}
\newcommand\longA{\uhr}
\newcommand\thinkA{\denker}
\newcommand\discussA{\wierfont}

```