

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

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

2022-01-19

## Abstract

TODO

---

\*Version 3.0 (last revised 2022-01-19)

# Contents

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

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

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

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

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

**Part I**  
**Manual**

# Chapter 1

## Stuff

### 1.1 Modules

---

`\sTeX`  
`\stex`

---

Both print this  $\text{\TeX}$  logo.

#### 1.1.1 Semantic Macros and Notations

Semantic macros invoke a formally declared symbol.

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

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

##### Example 1

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

$ab$

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

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



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

### Example 2

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

$a \cdot b$  and  $a \times b$

.

Not using an explicit option with a semantic macro yields the first declared notation, unless changed<sup>1</sup>.

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

### Example 3

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

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

.

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

### Example 4

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

Multiplying again by  $b$  yields...

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

### Example 5

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

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

<sup>1</sup>EdNOTE: TODO

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

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

### Example 6

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

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

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

### Example 7

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

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

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

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

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

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

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

### 2.1 Macros and Environments

---

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

---

---

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

---

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

---

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

---

---

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

---

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

---

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

---

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

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

---

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

---

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

---

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

---

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

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

---

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

---

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



## Chapter 3

# STEX-MathHub

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

### 3.1 Macros and Environments

---

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

---

#### 3.1.1 Files, Paths, URIs

---

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

---

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

---

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

---

---

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

---

---

<code>\stex_path_if_absolute_p:N</code>	$\star$
<code>\stex_path_if_absolute: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 .sms-file.
<hr/> <hr/> <code>\stex_in_repository:nn</code>	<code>\stex_in_repository:nn{&lt;repository-name&gt;}{&lt;code&gt;}</code> Change the current repository to <code>{&lt;repository-name&gt;}</code> (or not, if <code>{&lt;repository-name&gt;}</code> is empty), and passes its ID on to <code>{&lt;code&gt;}</code> as #1. Switches back to the previous repository after executing <code>{&lt;code&gt;}</code> .
<hr/> <hr/> <code>\mhpath *</code>	<code>\mhpath{&lt;archive-ID&gt;}{&lt;filename&gt;}</code> Expands to the full path of file <code>&lt;filename&gt;</code> in repository <code>&lt;archive-ID&gt;</code> . Does not check whether the file or the repository exist.
<hr/> <hr/> <code>\inputref</code> <hr/> <code>\inputref:nn</code>	<code>\inputref[&lt;archive-ID&gt;]{&lt;filename&gt;}</code> <code>\inputs</code> the file <code>&lt;filename&gt;</code> in repository <code>&lt;archive-ID&gt;</code> .
<hr/> <hr/> <code>\libinput</code>	<code>\libinput{&lt;filename&gt;}</code> Inputs <code>&lt;filename&gt;.tex</code> from the <code>lib</code> folders in the current archive and the <code>meta-inf</code> -archive of the current archive group (if existent). Throws an error if no file by that name exists in either folder, includes both if both exist.

## Test 2

```

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

```

```

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

```

## Chapter 4

# sTeX-References

Code related to links and cross-references

### 4.1 Macros and Environments

# Chapter 5

## sTeX-Modules

Code related to Modules

### 5.1 Macros and Environments

---

`\l_stex_current_module_prop`

---

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

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

Additionally, it stores:

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

---

`\l_stex_all_modules_seq`

---

Stores full URIs for all modules currently in scope.

---

```
\g_stex_module_files_prop
\g_stex_modules_in_file_seq
```

---

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

---

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

---



---

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

---

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

---

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

---

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

---

```
\stex_add_constant_to_current_module:n
```

---

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

---

```
\stex_add_import_to_current_module:n
```

---

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

---

```
\stex_modules_compute_namespace:nN \stex_modules_compute_namespace:nN
{\<namespace>} {\<path>}
```

---

Computes the namespace for file `<path>` in repository with namespace `<namespace>` as follows:

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

---

```
\stex_modules_current_namespace:
```

---

Computes the current namespace

### Test 3

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

```

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

```

.

### 5.1.1 The module-environment

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

---

`\stex_module_setup:nn`    `\stex_module_setup:nn{\langle params \rangle}{\langle name \rangle}`  
 Sets up a new module with name  $\langle name \rangle$  and optional parameters  $\langle params \rangle$ . In particular, sets `\l_stex_current_module_prop` appropriately.

---

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

**@module**            `\begin{@module}[\langle options \rangle]{\langle name \rangle}`  
 Core functionality of the `module-environment` without a header.

### Test 4

```

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

```

```

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

```

.

## Test 5

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

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

---

`\STEXModule` `\STEXModule {⟨fragment⟩}`

---

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

---

`\stex_invoke_module:n`

---

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

## Test 6

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






---

`\stex_activate_module:n`

---

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

## Chapter 6

# STEX-Module Inheritance

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

### 6.1 Macros and Environments

#### 6.1.1 SMS Mode

“SMS Mode” is used when loading modules from external tex files. It deactivates any output and ignores all T<sub>E</sub>X commands not explicitly allowed via the following lists:

---

`\g_stex_smsmode_allowedmacros_tl`

---

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

---

`\g_stex_smsmode_allowedmacros_escape_tl`

---

Macros that are executed with the category codes restored.

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

---

`\g_stex_smsmode_allowedenvs_seq`

---

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

---

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

---

Tests whether SMS mode is currently active.

---

`\stex_smsmode_set_codes:`

---

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

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

---

---

`\stex_in_smsmode:nn`

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

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

### Test 7

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

## 6.1.2 Imports and Inheritance

---

---

`\importmodule`

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

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

### Test 8

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

**Module 6.1.1[Foo]**

Meaning: >macro:->\stex\_invoke\_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo?foo}<

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

**Module 6.1.2[Importtest]**

modulesImporting module: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo Meaning: >macro:->\stex\_invoke\_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo?foo}<

**Module 6.1.3[Importtest2]**

modulesImporting module: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Importtest Meaning: >macro:->\stex\_invoke\_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo?foo}<

---

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

---

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

### Test 9

```

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

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

```

**Module 6.1.4**[UseTest1]

**Module 6.1.5**[UseTest2]  
file://home/jazzpirate/work/Software/ext/sTeX/doc/stextestUseTest1 Meaning: >undefined<

**Module 6.1.6**[UseTest3]  
modulesImporting module: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest2 Meaning: >undefined<  
Meaning: >macro:->\stex\_invoke\_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest2?bar}<  
All modules: http://mathhub.info/sTeX?Metatheory, file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest3, file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest2  
All symbols: http://mathhub.info/sTeX?Metatheory?isa, http://mathhub.info/sTeX?Metatheory?bind, http://mathhub.info/sTeX?Metatheory?collect, http://mathhub.info/sTeX?Metatheory?fromto, http://mathhub.info/sTeX?Metatheory?apply, http://mathhub.info/sTeX?Metatheory?collect, http://mathhub.info/sTeX?Metatheory?seqtype, http://mathhub.info/sTeX?Metatheory?sequence-index, http://mathhub.info/sTeX?Metatheory?aseqfromto, http://mathhub.info/sTeX?Metatheory?aseqfromtovia, http://mathhub.info/sTeX?Metatheory?module-type, http://mathhub.info/sTeX?Metatheory?mathematical-structure, file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest2?bar

### Test 10

```

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

```

Circular dependencies:

**Module 6.1.7**[CircDep1]  
>macro:->\stex\_invoke\_symbol:n {http://mathhub.info/tests/Foo/Bar/circular1?Circular1?fooA}<  
>macro:->\stex\_invoke\_symbol:n {http://mathhub.info/tests/Bar/Foo/circular2?Circular2?fooB}<

---

---

`\stex_import_module_uri:nn`

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

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

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

1. If `⟨archive-ID⟩` is empty:

- (a) If `⟨path⟩` is empty, then `⟨name⟩` must have been declared earlier in the same file and retrievable from `\g_stex_modules_in_file_seq`, or a file with name `⟨name⟩.⟨lang⟩.tex` must exist in the same folder, containing a module `⟨name⟩`. That module should have the same namespace as the current one.

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

2. Otherwise:

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

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

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

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

---

---

`\stex_import_require_module:nnnn`

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

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

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

# Chapter 7

## STEX-Symbols

Code related to symbol declarations and notations

### 7.1 Macros and Environments

---

<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
```

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

---

---

`\symdef`

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

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

### Test 13

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

Module 7.1.3[SymdefTest]  
 $a+b+c$



# Chapter 8

## STEX-Terms

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

### 8.1 Macros and Environments

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

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

### Test 14

```

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

```

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

### Test 15

```

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

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

```

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

---

---

`\stex_term_custom:nn`

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

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

### Test 16

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

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

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

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

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

\bar![bar]

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

\end{module}
```

```
Module 8.1.3[TextTest]
modulesImporting module: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo
some aand some band also some here.
some a and some b and also some c here.
bar
or just some c
bar
or first b, then c, and finally a
```

---

---

`\stex_highlight_term:nn`

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

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

---

`\comp`

`\comp{<args>}`

`\compemph`

`\compemph@uri`

`\defemph`

`\defemph@uri`

`\symrefemph`

`\symrefemph@uri`

---

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

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

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

---

`\STEXinvisible`

---

Exports its argument as OMDoc (invisible), but does not produce PDF output. Useful e.g. for semantic macros that take arguments that are not part of the symbolic notation.

---

`\ellipses`

---

TODO

# Chapter 9

## STEX-Structural Features

Code related to structural features

### 9.1 Macros and Environments

#### 9.1.1 Structures

mathstructure    TODO

Test 17

```

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

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

\present\magma

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

Test: $\mM{op}ab$

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

```

Module 9.1.1[StructureTest1]

a**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

# sTeX-Statements

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

### 10.1 Macros and Environments

`symboldoc`      `\begin{<symboldoc>}{<symbols>} <text> \end{<symboldoc>}`  
Declares *<text>* to be a (natural language, encyclopaedic) description of *{<symbols>}*  
(a comma separated list of symbol identifiers).

## Chapter 11

# **sTeX-Proofs: Structural Markup for Proofs**

The `sproof` package is part of the sTeX collection, a version of T<sub>E</sub>X/L<sup>A</sup>T<sub>E</sub>X that allows to markup T<sub>E</sub>X/L<sup>A</sup>T<sub>E</sub>X documents semantically without leaving the document format, essentially turning T<sub>E</sub>X/L<sup>A</sup>T<sub>E</sub>X into a document format for mathematical knowledge management (MKM).

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

## Contents

## 11.1 Introduction

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

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

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

Example 1: A very explicit proof, marked up semantically

We will go over the general intuition by way of our running example (see Figure 1 for the source and Figure 2 for the formatted result).<sup>4</sup>

<sup>4</sup>EdNOTE: talk a bit more about proofs and their structure,... maybe copy from OMDoc spec.



## 11.2 The User Interface

### 11.2.1 Package Options

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

### 11.2.2 Proofs and Proof steps

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

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

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

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

### 11.2.3 Justifications

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

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

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

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

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

#### 11.2.4 Proof Structure

<b>subproof</b>	The <b>pfcases</b> environment is used to mark up a subproof. This environment takes an optional <b>KeyVal</b> argument for semantic annotations and a second argument that allows to specify an introductory comment (just like in the <b>proof</b> environment). The <b>method</b> key can be used to give the name of the proof method executed to make this subproof.
<b>spfcases</b>	The <b>pfcases</b> environment is used to mark up a proof by cases. Technically it is a variant of the <b>subproof</b> where the <b>method</b> is <b>by-cases</b> . Its contents are <b>spfcases</b> environments that mark up the cases one by one.
<b>spfcases</b>	The content of a <b>pfcases</b> environment are a sequence of case proofs marked up in the <b>pfcases</b> environment, which takes an optional <b>KeyVal</b> argument for semantic annotations. The second argument is used to specify the the description of the case under consideration. The content of a <b>pfcases</b> environment is the same as that of a <b>proof</b> , i.e. <b>steps</b> , <b>proofcomments</b> , and <b>pfcases</b> environments. <b>\spfcasesketch</b> is a variant of the <b>spfcases</b> environment that takes the same arguments, but instead of the <b>spfsteps</b> in the body uses a third argument for a proof sketch.
<b>\spfcasesketch</b>	
<b>sproofcomment</b>	The <b>proofcomment</b> environment is much like a <b>step</b> , only that it does not have an object-level assertion of its own. Rather than asserting some fact that is relevant for the proof, it is used to explain where the proof is going, what we are attempting to to, or what we have achieved so far. As such, it cannot be the target of a <b>\premise</b> .



1. The numbering scheme of proofs cannot be changed. It is more geared for teaching proof structures (the author's main use case) and not for writing papers. reported by Tobias Pfeiffer (fixed)
2. currently proof steps are formatted by the `LATEX description` environment. We would like to configure this, e.g. to use the `inparaenum` environment for more condensed proofs. I am just not sure what the best user interface would be I can imagine redefining an internal environment `spf@proofstep@list` or adding a key `prooflistenv` to the `proof` environment that allows to specify the environment directly. Maybe we should do both.

## Chapter 12

# sTeX-Metatheory

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

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

### 12.1 Symbols

**Part III**  
**Extensions**

## Chapter 13

# Tikzinput

### 13.1 Macros and Environments

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

## Chapter 14

# document-structure.sty: Semantic Markup for Open Mathematical Documents in L<sup>A</sup>T<sub>E</sub>X

The `omdoc` package is part of the  $\text{\LaTeX}$  collection, 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).

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

### 14.1 Introduction

$\text{\LaTeX}$  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). 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  $\text{\LaTeX}$  sources, or after translation. Currently, trans-document referencing provided by this package can only be used in the  $\text{\LaTeX}$  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 $\text{\LaTeX}$ packages

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

### 14.2.2 Document Structure

document

\documentkeys

id

The top-level `document` environment can be given key/value information by the `\documentkeys` macro in the preamble<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  $\text{\LaTeX}$ ML transformation.

omgroup

id

creators

contributors

short

loadmodules

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

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

blindomgroup

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

<sup>6</sup>EdNOTE: integrate with `latexml`'s `XMRef` in the `Math` mode.  
<sup>2</sup>We cannot patch the document environment to accept an optional argument, since other packages we load already do; pity.

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

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

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

Example 3: A typical Document Structure of a Book

`\skipomgroup`

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

`\currentsectionlevel`

`\CurrentSectionLevel`

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

### 14.2.3 Ignoring Inputs

`ignore`  
`showignores`

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

<sup>3</sup>We shied away from redefining the `frontmatter` to induce a `blindomgroup`, but this may be the “right” way to go in the future.

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

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

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

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

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

#### 14.2.4 Structure Sharing

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

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

#### 14.2.5 Global Variables

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

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

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

### 14.2.6 Colors

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

## 14.3 Limitations

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

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

# Chapter 15

## Slides and Course Notes

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

### 15.1 Introduction

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

### 15.2 The User Interface

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

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

#### 15.2.1 Package Options

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

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

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

### 15.2.2 Notes and Slides

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

⚠ Note that it is essential to start and end the `notes` environment at the start of the line – in particular, there may not be leading blanks – else L<sup>A</sup>T<sub>E</sub>X becomes confused and throws error messages that are difficult to decipher.

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

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

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

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

Example 4: A typical Course Notes File

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

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

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

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

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

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

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

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

### 15.2.3 Header and Footer Lines of the Slides

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

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

### 15.2.4 Frame Images

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

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

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

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

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


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

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

<sup>9</sup>EdNOTE: MK: the `hyperref` link does not seem to work yet. I wonder why but do not have the time to fix it.

`\mhframeimage{baz/foobar}`

## 15.2.5 Colors and Highlighting

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

## 15.2.6 Front Matter, Titles, etc.

### 15.2.7 Excursions

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

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

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

```
\activateexcursion where \activateexcursion{<path>} augments the \printexcursions macro by a
\printexcursions call \inputref{<path>}. In this way, the3 \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

<code>T</code>	• <code>T</code> for true answers, <code>F</code> for false ones,
<code>F</code>	
<code>Ttext</code>	• <code>Ttext</code> the verdict for true answers, <code>Ftext</code> for false ones, and
<code>Ftext</code>	
<code>feedback</code>	• <code>feedback</code> for a short feedback text given to the student.

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 — 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` specifies a string that specifies the duration of the test, `min` specifies the equivalent in number of minutes, and `reqpts` the points that are required for a perfect grade.

### 17.2.4 Including Assignments

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

### 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-compat}
22 \RequirePackage{morewrites}
23 \RequirePackage{amsmath}
24
25 Package options:
26 \keys_define:nn { stex } {
27   debug      .clist_set:N = \c_stex_debug_clist ,
```

```

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

```

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

**\sTeX**

```

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

```

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

## 18.3 Messages and logging

```

41 <@@=stex_log>

```

Warnings and error messages

```

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

```

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

```

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

```

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

Redirecting messages:

```

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

```

## 18.4 Persistence

76 `<@=stex_persist>`

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

```

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

```

(End definition for `\c__stex_persist_sms_iow`.)

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

```

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

```

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

## 18.5 HTML Annotations

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

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

```

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

```

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

```

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

```

```

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

```

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

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

```

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

```

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

`\_stex_annotate_checkempty:n`

```

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

```

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

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

```

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

```

(End definition for \l\_stex\_html\_do\_output\_bool and \stex\_if\_do\_html:. These functions are documented on page ??.)

`\stex_suppress_html:n` Whether to (locally) produce HTML output

```

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

```

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

`\stex_annotate:env`

`\stex_annotate_invisible:n`

`\stex_annotate_invisible:nnn`

We define four macros for introducing attributes in the HTML output. The definitions depend on the “backend” used (L<sup>A</sup>T<sub>E</sub>XML, R<sub>U</sub>S<sub>T</sub>E<sub>X</sub>, p<sub>D</sub>F<sub>L</sub>A<sub>T</sub>E<sub>X</sub>).

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

```

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

```

```

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

```

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

## 18.6 Languages

```

232 <@=stex_language>

```

`\c_stex_languages_prop`  
`\c_stex_language_abbrevs_prop`

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

```

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

```

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

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

```

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

```

## 18.7 Activating/Deactivating Macros

`\stex_deactivate_macro:Nn`

```

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

```



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

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

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

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

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

## Chapter 19

# STEX -MathHub Implementation

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

### 19.1 Generic Path Handling

We treat paths as L<sup>A</sup>T<sub>E</sub>X3-sequences (of the individual path segments, i.e. separated by a /-character) unix-style; i.e. a path is absolute if the sequence starts with an empty entry.

```
\stex_path_from_string:Nn
\stex_path_from_string:NV
\stex_path_from_string:cn
\stex_path_from_string:cV
295 \cs_new_protected:Nn \stex_path_from_string:Nn {
296   \str_set:Nx \l_tmpa_str { #2 }
297   \str_if_empty:NTF \l_tmpa_str {
298     \seq_clear:N #1
299   }{
300     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
301     \sys_if_platform_windows:T{
302       \seq_clear:N \l_tmpa_tl
303       \seq_map_inline:Nn #1 {
304         \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
305         \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
306       }
307     }
308   }
```

```

306     }
307     \seq_set_eq:NN #1 \l_tmpa_tl
308   }
309   \stex_path_canonicalize:N #1
310 }
311 }
312 \cs_generate_variant:Nn \stex_path_from_string:Nn
313 { NV, cn, cV }

```

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

```

\stex_path_to_string:NN
\stex_path_to_string:N
314 \cs_new_protected:Nn \stex_path_to_string:NN {
315   \exp_args:NNe \str_set:Nn #2 { \seq_use:Nn #1 / }
316 }
317
318 \cs_new:Nn \stex_path_to_string:N {
319   \seq_use:Nn #1 /
320 }

```

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

```

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

```

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

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

```

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

```

```

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

```

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

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

```

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

```

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

## 19.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

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

```

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

We determine the PWD

`\c_stex_pwd_seq`  
`\c_stex_pwd_str`

```

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

```

(End definition for `\c_stex_pwd_seq` and `\c_stex_pwd_str`. These variables are documented on page 11.)

## 19.3 File Hooks and Tracking

385 `<@@=stex_files>`

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

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

`\g__stex_files_stack` keeps track of file changes

386 `\seq_gclear_new:N\g__stex_files_stack`

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`

`\c_stex_mainfile_str`

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

388 `\stex_path_from_string:Nn \c_stex_mainfile_seq`

389 `\c_stex_mainfile_str`

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

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

```

390 \seq_gclear_new:N\g_stex_currentfile_seq
391 \AddToHook{file/before}{
392   \stex_path_from_string:Nn\g_stex_currentfile_seq{\CurrentFilePath}
393   \stex_path_if_absolute:NTF\g_stex_currentfile_seq{
394     \exp_args:NNe\seq_put_right:Nn\g_stex_currentfile_seq{\CurrentFile}
395   }{
396     \stex_path_from_string:Nn\g_stex_currentfile_seq{
397       \c_stex_pwd_str/\CurrentFilePath/\CurrentFile
398     }
399   }
400   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
401   \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
402 }
403 \AddToHook{file/after}{
404   \seq_if_empty:NF\g__stex_files_stack{
405     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
406   }
407   \seq_if_empty:NTF\g__stex_files_stack{
408     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
409   }{
410     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
411     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
412   }
413 }
```

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

## 19.4 MathHub Repositories

```

414 <@@=stex_mathhub>

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

```

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

```

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

```

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

```

\l__stex_mathhub_manifest_file_seq
451 \str_new:N\l__stex_mathhub_manifest_file_seq

```

(End definition for `\l__stex_mathhub_manifest_file_seq`.)

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

```

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

```

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

`\c_stex_mathhub_manifest_ior` File variable used for MANIFEST-files

```

487 \ior_new:N \c_stex_mathhub_manifest_ior

```

(End definition for `\c_stex_mathhub_manifest_ior`.)

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

```

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

```

```

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

```

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

`\stex_set_current_repository:n`

```

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

```

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

`\stex_require_repository:n`

```

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

```



```

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

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

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

```

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

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

```

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

```

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

```

\inputref
\stex_inputref:nn
\mhinput\stex_mhinput:nn

585 \newif \ifinputref \inputreffalse
586
587 \cs_new_protected:Nn \stex_mhinput:nn {
588   \stex_in_repository:nn {#1} {
589     \ifinputref
590       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
591     \else
592       \inputreftrue
593       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
594       \inputreffalse
595     \fi
596   }
597 }
598 \NewDocumentCommand \mhinput { 0{} m}{
599   \stex_mhinput:nn{ #1 }{ #2 }
600 }
601
602 \cs_new_protected:Nn \stex_inputref:nn {
603   \stex_in_repository:nn {#1} {
604     \bool_lazy_any:nTF {
605       {\rustex_if_p:} {\latexml_if_p:}
606     } {
607       \str_clear:N \l_tmpa_str
608       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
609         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
610       }
611       \stex_annotate_invisible:nnn{inputref}{
612         \l_tmpa_str / #2
613       }{}
614     }{
615       \begingroup
616         \inputreftrue
617         \input{ \c_stex_mathhub_str / ##1 / source / #2 }
618       \endgroup
619     }
620   }
621 }
622
623 \NewDocumentCommand \inputref { 0{} m}{
624   \stex_inputref:nn{ #1 }{ #2 }
625 }
626
627 \cs_new_protected:Nn \stex_mhbibresource:nn {
628   \stex_in_repository:nn {#1} {
629     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
630   }
631 }
632 \newcommand\addmhbibresource[2] []{
633   \stex_mhbibresource:nn{ #1 }{ #2 }
634 }

```

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

**\mhpath**

```
635 \def \mhpath #1 #2 {
636   \exp_args:Ne \str_if_eq:nnTF{#1}{}{
637     \c_stex_mathhub_str /
638     \prop_item:Nn \l_stex_current_repository_prop { id }
639     / source / #2
640   }{
641     \c_stex_mathhub_str / #1 / source / #2
642   }
643 }
```

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

**\libinput**

```
644 \cs_new_protected:Npn \libinput #1 {
645   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
646     \msg_error:nnn{stex}{error/notinarchive}\libinput
647   }
648   \bool_set_false:N \l_tmpa_bool
649   \tl_clear:N \l_tmpa_tl
650   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
651   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
652   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str
653   \seq_pop_left:NNT \l_tmpb_seq \l_tmpb_str {
654     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
655     \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
656       / meta-inf / lib / #1.tex}{
657       \bool_set_true:N \l_tmpa_bool
658       \tl_put_right:Nx \l_tmpa_tl {
659         \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
660           / meta-inf / lib / #1.tex}
661       }
662     }{}
663   }
664   \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
665     / \l_tmpa_str / lib / #1.tex
666   }{
667     \bool_set_true:N \l_tmpa_bool
668     \tl_put_right:Nx \l_tmpa_tl {
669       \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
670         / \l_tmpa_str / lib / #1.tex}
671     }
672   }{}
673   \bool_if:NF \l_tmpa_bool {
674     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
675   }
676   \l_tmpa_tl
677 }
```

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

```
678 \</package>
```

## Chapter 20

# STEX -References Implementation

```
679 <*package>
680
681 %%%%%%%%%% references.dtx %%%%%%%%%%
682
683 %\RequirePackage{hyperref}
684 %\RequirePackage{cleveref}
685 <@@=stex_refs>
686
687 Warnings and error messages
688
689 \iow_new:N \c__stex_refs_refs_iow
690 \AddToHook{begindocument}{
691   \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
692 }
693 \AddToHook{enddocument}{
694   \iow_close:N \c__stex_refs_refs_iow
695 }
696
697 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
698
699 \NewDocumentCommand \STEXreftitle { m } {
700   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
701 }
702
```

### 20.1 Document URIs and URLs

```
700 \seq_new:N \g__stex_refs_all_refs_seq
701
702 \str_new:N \l_stex_current_docns_str
703
704 \cs_new_protected:Nn \stex_get_document_uri: {
705   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
706   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
707   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
708   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
709 }
```

```

709 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
710
711 \str_clear:N \l_tmpa_str
712 \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
713   \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
714 }
715
716 \str_if_empty:NTF \l_tmpa_str {
717   \str_set:Nx \l_stex_current_docns_str {
718     file:/\stex_path_to_string:N \l_tmpa_seq
719   }
720 }{
721   \bool_set_true:N \l_tmpa_bool
722   \bool_while_do:Nn \l_tmpa_bool {
723     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
724     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
725       {source} { \bool_set_false:N \l_tmpa_bool }
726     }{}{
727       \seq_if_empty:NT \l_tmpa_seq {
728         \bool_set_false:N \l_tmpa_bool
729       }
730     }
731   }
732
733   \seq_if_empty:NTF \l_tmpa_seq {
734     \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
735   }{
736     \str_set:Nx \l_stex_current_docns_str {
737       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
738     }
739   }
740 }
741 }
742
743 \str_new:N \l_stex_current_docurl_str
744 \cs_new_protected:Nn \stex_get_document_url: {
745   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
746   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
747   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
748   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
749   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
750
751   \str_clear:N \l_tmpa_str
752   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
753     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
754       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
755     }
756   }
757
758   \str_if_empty:NTF \l_tmpa_str {
759     \str_set:Nx \l_stex_current_docurl_str {
760       file:/\stex_path_to_string:N \l_tmpa_seq
761     }
762   }{
763     \bool_set_true:N \l_tmpa_bool

```

```

763 \bool_while_do:Nn \l_tmpa_bool {
764   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
765   \exp_args:No \str_case:nnTF { \l_tmpb_str } {
766     {source} { \bool_set_false:N \l_tmpa_bool }
767   }{}{
768     \seq_if_empty:NT \l_tmpa_seq {
769       \bool_set_false:N \l_tmpa_bool
770     }
771   }
772 }
773
774 \seq_if_empty:NTF \l_tmpa_seq {
775   \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
776 }{
777   \str_set:Nx \l_stex_current_docurl_str {
778     \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
779   }
780 }
781 }
782 }

```

## 20.2 Setting Reference Targets

```

783 \str_const:Nn \c__stex_refs_url_str{URL}
784 \str_const:Nn \c__stex_refs_ref_str{REF}
785 % @currentlabel -> number
786 % @currentlabelname -> title
787 % @currentHref -> name.number <- id of some kind
788 % \theH# -> \arabic{section}
789 % \the# -> number
790 % \hyper@makecurrent{#}
791 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
792   \stex_get_document_uri:
793   \str_set:Nx \l_tmpa_str { #1 }
794   \str_if_empty:NT \l_tmpa_str {
795     \int_zero:N \l_tmpa_int
796     \bool_set_true:N \l_tmpa_bool
797     \bool_while_do:Nn \l_tmpa_bool {
798       \cs_if_exist:cTF {
799         sref_\l_stex_current_docns_str\c_hash_str REF_\int_use:N \l_tmpa_int _type
800       }{
801         \int_incr:N \l_tmpa_int
802       }{
803         \str_set:Nx \l_tmpa_str { REF_\int_use:N \l_tmpa_int }
804         \bool_set_false:N \l_tmpa_bool
805       }
806     }
807   }
808   \str_set:Nx \l_tmpa_str {
809     \l_stex_current_docns_str\c_hash_str\l_tmpa_str
810   }
811   \seq_gput_right:No \g__stex_refs_all_refs_seq \l_tmpa_str
812   \stex_if_smsmode:TF {
813     \stex_get_document_url:

```

```

814 \str_gset_eq:cN {sref_url_\l_tmpa_str_str}\l_stex_current_docurl_str
815 \str_gset_eq:cN {sref_\l_tmpa_str_type}\c__stex_refs_url_str
816 }{
817 \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter{\@currentlabel\iffalse}}{
818 \exp_after:wN\label\exp_after:wN{sref_\l_tmpa_str}
819 \str_gset:cn {sref_\l_tmpa_str_type}\c__stex_refs_ref_str
820 }
821 }

822 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
823 \str_gset_eq:cN {sref_sym_#1_uri} \l_stex_current_docns_str
824 }

```

## 20.3 Using References

```

825 \str_new:N \l__stex_refs_indocument_str
826 \keys_define:nn { stex / sref } {
827   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
828   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
829   pre           .tl_set:N = \l__stex_refs_pre_tl ,
830   post          .tl_set:N = \l__stex_refs_post_tl ,
831   %indoc        .str_set_x:N = \l__stex_refs_repo_str ,
832 }
833
834 \bool_new:N \c__stex_refs_hyperref_bool
835 \bool_set_false:N \c__stex_refs_hyperref_bool
836 \AddToHook{begindocument}{
837   \@ifpackageloaded{hyperref}{
838     \bool_set_true:N \c__stex_refs_hyperref_bool
839   }{}
840 }
841
842
843 \cs_new_protected:Nn \__stex_refs_args:n {
844   \tl_clear:N \l__stex_refs_linktext_tl
845   \tl_clear:N \l__stex_refs_fallback_tl
846   \tl_clear:N \l__stex_refs_pre_tl
847   \tl_clear:N \l__stex_refs_post_tl
848   \str_clear:N \l__stex_refs_repo_str
849   \keys_set:nn { stex / sref } { #1 }
850 }
851
852 \NewDocumentCommand \sref { 0{} m}{
853   \__stex_refs_args:n { #1 }
854   \str_if_empty:NTF \l__stex_refs_indocument_str {
855     \str_set:Nn \l_tmpa_str { #2 }
856     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
857     \tl_set:Nn \l_tmpa_tl {
858       \l__stex_refs_fallback_tl
859     }
860     \seq_map_inline:Nn \g__stex_refs_all_refs_seq {
861       \str_set:Nn \l_tmpb_str { ##1 }
862       \str_if_eq:eeT { \l_tmpa_str } {
863         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int }{-1 }
864       } {

```

```

865     \seq_map_break:n {
866       \tl_set:Nn \l_tmpa_tl {
867         % doc uri in \l_tmpb_str
868         \str_set:Nx \l_tmpa_str {sref_url_\l_tmpb_str_type}
869         \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
870           % reference
871           \l__stex_refs_pre_tl\ref{sref_\l_tmpb_str}\l__stex_refs_post_tl
872         }{
873           % URL
874           \if_bool:N \c__stex_refs_hyperref_bool {
875             \exp_args:Nx \href{\use:c{sref_url_\l_tmpb_str_str}}{\l__stex_refs_fallback
876           }{
877             \l__stex_refs_fallback_tl
878           }
879         }
880       }
881     }
882   }
883 }
884 \l_tmpa_tl
885 }{
886   % TODO
887 }
888 }
889
890 </package>

```



## Chapter 21

# STEX -Modules Implementation

```
891 <*package>
892
893 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
894
895 <@@=stex_modules>
896
897   Warnings and error messages
898   \msg_new:nnn{stex}{error/unknownmodule}{
899     No~module~#1~found
900   }
901   \msg_new:nnn{stex}{error/syntax}{
902     Syntax~error:~#1
903   }
904   \msg_new:nnn{stex}{error/siglanguage}{
905     Module~#1~declares~signature~#2,~but~does~not~
906     declare~its~language
907   }
```

`\l_stex_current_module_prop` The current module:

```
906 \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

```
907 \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`

```
908 \seq_new:N \g_stex_modules_in_file_seq
909 \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
910 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
911   \prop_if_empty:NTF \l_stex_current_module_prop
912   \prg_return_false: \prg_return_true:
913 }

```

(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
914 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
915   \prop_if_exist:cTF { c_stex_module_#1_prop }
916   \prg_return_true: \prg_return_false:
917 }

```

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

```

\stex_add_to_current_module:n
\STEXexport
918 \cs_new_protected:Nn \stex_add_to_current_module:n {
919   \prop_get:NnN \l_stex_current_module_prop { content } \l_tmpa_tl
920   \tl_put_right:Nn \l_tmpa_tl { #1 }
921   \prop_put:Nno \l_stex_current_module_prop { content } { \l_tmpa_tl }
922 }
923 \cs_new_protected:Npn \STEXexport {
924   \begingroup
925   \newlinechar=-1\relax
926   \endlinechar=-1\relax
927   %\catcode'\ = 9\relax
928   \expandafter\endgroup\STEXexport:n
929 }
930 \cs_new_protected:Nn \STEXexport:n {
931   \ignorespaces #1
932   \stex_add_to_current_module:n { \ignorespaces #1 }
933   \stex_smsmode_set_codes:
934 }
935 \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
936 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
937   \str_set:Nx \l_tmpa_str { #1 }
938   \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
939   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
940   \prop_put:Nno \l_stex_current_module_prop { constants } \l_tmpa_seq
941 }

```

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

```

\stex_add_import_to_current_module:n
942 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
943   \str_set:Nx \l_tmpa_str { #1 }
944   \prop_get:NnN \l_stex_current_module_prop { imports } \l_tmpa_seq
945   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
946   \prop_put:Nno \l_stex_current_module_prop { imports } \l_tmpa_seq
947 }

```

(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).

```

948 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
949   \str_set:Nx \l_tmpa_str { #1 }
950   \seq_set_eq:NN \l_tmpa_seq #2
951   % split off file extension
952   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
953   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
954   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
955   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
956
957   \bool_set_true:N \l_tmpa_bool
958   \bool_while_do:Nn \l_tmpa_bool {
959     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
960     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
961       {source} { \bool_set_false:N \l_tmpa_bool }
962     }{}{
963       \seq_if_empty:NT \l_tmpa_seq {
964         \bool_set_false:N \l_tmpa_bool
965       }
966     }
967   }
968
969   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
970   \str_if_empty:NTF \l_stex_modules_subpath_str {
971     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
972   }{
973     \str_set:Nx \l_stex_modules_ns_str {
974       \l_tmpa_str/\l_stex_modules_subpath_str
975     }
976   }
977 }
```

(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`

```

978 \str_new:N \l_stex_modules_ns_str
979 \str_new:N \l_stex_modules_subpath_str
```

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

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

```

980 \cs_new_protected:Nn \stex_modules_current_namespace: {
981   \str_clear:N \l_stex_modules_subpath_str
982   \prop_get:NnTF \l_stex_current_repository_prop { ns } \l_tmpa_str {
983     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
984   }{
985     % split off file extension
986     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
987     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
```

```

988     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
989     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
990     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
991     \str_set:Nx \l_stex_modules_ns_str {
992       file:/\stex_path_to_string:N \l_tmpa_seq
993     }
994   }
995 }

```

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

## 21.1 The module environment

module arguments:

```

996 \keys_define:nn { stex / module } {
997   title      .str_set_x:N = \l_stex_module_title_str ,
998   ns         .str_set_x:N = \l_stex_module_ns_str ,
999   lang       .str_set_x:N = \l_stex_module_lang_str ,
1000  sig         .str_set_x:N = \l_stex_module_sig_str ,
1001  creators    .str_set_x:N = \l_stex_module_creators_str ,
1002  contributors .str_set_x:N = \l_stex_module_contributors_str ,
1003  meta        .str_set_x:N = \l_stex_module_meta_str ,
1004  srccite     .str_set_x:N = \l_stex_module_srccite_str
1005 }
1006
1007 \cs_new_protected:Nn \__stex_modules_args:n {
1008   \str_clear:N \l_stex_module_title_str
1009   \str_clear:N \l_stex_module_ns_str
1010   \str_clear:N \l_stex_module_lang_str
1011   \str_clear:N \l_stex_module_sig_str
1012   \str_clear:N \l_stex_module_creators_str
1013   \str_clear:N \l_stex_module_contributors_str
1014   \str_clear:N \l_stex_module_meta_str
1015   \str_clear:N \l_stex_module_srccite_str
1016   \keys_set:nn { stex / module } { #1 }
1017 }
1018
1019 % module parameters here? In the body?
1020

```

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

```

1021 \cs_new_protected:Nn \stex_module_setup:nn {
1022   \str_set:Nx \l_stex_module_name_str { #2 }
1023   \__stex_modules_args:n { #1 }

```

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

```

1024 \stex_if_in_module:TF {
1025   % Nested module
1026   \prop_get:NnN \l_stex_current_module_prop
1027     { ns } \l_stex_module_ns_str
1028   \str_set:Nx \l_stex_module_name_str {
1029     \prop_item:Nn \l_stex_current_module_prop

```

```

1030     { name } / \l_stex_module_name_str
1031   }
1032 }{
1033   % not nested:
1034   \str_if_empty:NT \l_stex_module_ns_str {
1035     \stex_modules_current_namespace:
1036     \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1037     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1038       / { \l_stex_module_ns_str }
1039     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1040     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1041       \str_set:Nx \l_stex_module_ns_str {
1042         \stex_path_to_string:N \l_tmpa_seq
1043       }
1044     }
1045   }
1046 }

```

Next, we determine the language of the module:

```

1047   \str_if_empty:NT \l_stex_module_lang_str {
1048     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1049     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1050     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1051     \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1052     \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1053       \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1054         inferred~from~file~name}
1055       \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1056     }
1057   }
1058
1059   \str_if_empty:NF \l_stex_module_lang_str {
1060     \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1061     \l_tmpa_str {
1062       \ltx@ifpackageloaded{babel}{
1063         \exp_args:Nx \selectlanguage { \l_tmpa_str }
1064       }{}
1065     } {
1066       \msg_error:nmx{stex}{error/unknownlanguage}{\l_tmpa_str}
1067     }
1068   }

```

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

```

1069   \str_if_empty:NTF \l_stex_module_sig_str {
1070     \str_clear:N \l_tmpa_str
1071     \seq_clear:N \l_tmpa_seq
1072     \tl_clear:N \l_tmpa_tl
1073     \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
1074       name      = \l_stex_module_name_str ,
1075       ns        = \l_stex_module_ns_str ,
1076       imports   = \exp_not:o { \l_tmpa_seq } ,
1077       constants = \exp_not:o { \l_tmpa_seq } ,
1078       content   = \exp_not:o { \l_tmpa_tl } ,

```

```

1079     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1080     lang      = \l_stex_module_lang_str ,
1081     sig       = \l_stex_module_sig_str ,
1082     meta      = \l_stex_module_meta_str
1083   }
1084 }{
1085   \str_if_empty:NT \l_stex_module_lang_str {
1086     \msg_error:nnxx{stex}{error/siglanguage}{
1087       \l_stex_module_ns_str?\l_stex_module_name_str
1088     }\l_stex_module_sig_str}
1089   }
1090
1091   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1092   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1093   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1094   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1095   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1096   \str_set:Nx \l_tmpa_str {
1097     \stex_path_to_string:N \l_tmpa_seq /
1098     \l_tmpa_str . \l_stex_module_sig_str .tex
1099   }
1100   \IfFileExists \l_tmpa_str {
1101     \exp_args:No \stex_in_smsmode:nn { \l_tmpa_str } {
1102       \seq_clear:N \l_stex_all_modules_seq
1103       \prop_clear:N \l_stex_current_module_prop
1104       \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1105       \input { \l_tmpa_str }
1106     }
1107   }{
1108     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1109   }
1110   \stex_activate_module:n {
1111     \l_stex_module_ns_str ? \l_stex_module_name_str
1112   }
1113   \prop_set_eq:Nc \l_stex_current_module_prop {
1114     c_stex_module_
1115     \l_stex_module_ns_str ?
1116     \l_stex_module_name_str
1117     _prop
1118   }
1119 }

```

We load the metatheory:

```

1120   \str_if_empty:NT \l_stex_module_meta_str {
1121     \str_set:Nx \l_stex_module_meta_str {
1122       \c_stex_metatheory_ns_str ? Metatheory
1123     }
1124   }
1125   \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1126     \exp_args:Nx \stex_add_to_current_module:n {
1127       \stex_activate_module:n {\l_stex_module_meta_str}
1128     }
1129     \stex_activate_module:n {\l_stex_module_meta_str}
1130   }

```

```
1131 }
```

(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}

1132 \cs_new_protected:Nn \__stex_modules_begin_module:nn {
1133   \stex_reactivate_macro:N \STEXexport
1134   \stex_reactivate_macro:N \importmodule
1135   \stex_reactivate_macro:N \symdecl
1136   \stex_reactivate_macro:N \notation
1137   \stex_reactivate_macro:N \symdef
1138   \stex_module_setup:nn{#1}{#2}
1139
1140   \stex_debug:nn{modules}{
1141     New~module:\\
1142     Namespace:~\l_stex_module_ns_str\\
1143     Name:~\l_stex_module_name_str\\
1144     Language:~\l_stex_module_lang_str\\
1145     Signature:~\l_stex_module_sig_str\\
1146     Metatheory:~\l_stex_module_meta_str\\
1147     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1148   }
1149
1150   \seq_put_right:Nx \l_stex_all_modules_seq {
1151     \l_stex_module_ns_str ? \l_stex_module_name_str
1152   }
1153
1154   \seq_gput_right:Nx \g_stex_modules_in_file_seq
1155     { \l_stex_module_ns_str ? \l_stex_module_name_str }
1156
1157   \stex_if_smsmode:TF {
1158     \stex_smsmode_set_codes:
1159   } {
1160     \begin{stex_annotate_env} {theory} {
1161       \l_stex_module_ns_str ? \l_stex_module_name_str
1162     }
1163
1164     \stex_annotate_invisible:nnn{header}{} {
1165       \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1166       \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1167       \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1168         \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1169       }
1170     }
1171   }
1172   % TODO: Inherit metatheory for nested modules?
1173 }
1174 \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}
```

```

1175 \cs_new_protected:Nn \__stex_modules_end_module: {
1176   \str_set:Nx \l_tmpa_str {
1177     c_stex_module_
1178     \prop_item:Nn \l_stex_current_module_prop { ns } ?
1179     \prop_item:Nn \l_stex_current_module_prop { name }
1180     _prop
1181   }
1182   %^^A \prop_new:c { \l_tmpa_str }
1183   \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
1184   \stex_debug:nn{modules}{Closing module~\prop_item:Nn \l_stex_current_module_prop { name }}
1185 }

```

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

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

```

1186 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1187 \NewDocumentEnvironment { @module } { 0{} m } {
1188   \par
1189   \__stex_modules_begin_module:nn{#1}{#2}
1190 } {
1191   \__stex_modules_end_module:
1192   \stex_if_smsmode:TF {
1193     \exp_args:Nx \stex_add_to_sms:n {
1194       \prop_gset_from_keyval:cn {
1195         c_stex_module_
1196         \prop_item:Nn \l_stex_current_module_prop { ns } ?
1197         \prop_item:Nn \l_stex_current_module_prop { name }
1198         _prop
1199       } {
1200         name      = \prop_item:cn { \l_tmpa_str } { name } ,
1201         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
1202         imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
1203         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
1204         content   = \prop_item:cn { \l_tmpa_str } { content } ,
1205         file      = \prop_item:cn { \l_tmpa_str } { file } ,
1206         lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
1207         sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
1208         meta      = \prop_item:cn { \l_tmpa_str } { meta }
1209       }
1210     }
1211   }{
1212     \end{stex_annotate_env}
1213   }
1214 }

```

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

```

1215 \cs_if_exist:NTF \thesection {
1216   \newcounter{module}[section]
1217 }{
1218   \newcounter{module}
1219 }
1220
1221 \bool_if:NT \c_stex_showmods_bool {
1222   \latexml_if:F { \RequirePackage{mdframed} }

```



```

1223 }
1224
1225 \cs_new_protected:Nn \stex_modules_heading: {
1226   \stepcounter{module}
1227   \par
1228   \bool_if:NT \c_stex_showmods_bool {
1229     \noindent{\textbf{Module} ~
1230       \cs_if_exist:NT \thesection {\thesection.}
1231       \themodule ~ [\l_stex_module_name_str]
1232     }
1233     \str_if_empty:NTF \l_stex_module_title_str {
1234       }{
1235         \quad(\l_stex_module_title_str)\hfill
1236       }\par
1237     }
1238     \edef\@currentlabel{Module~\thesection.\themodule~[\l_stex_module_name_str]}
1239     % TODO
1240     \stex_ref_new_doc_target:n \l_stex_module_name_str
1241   }

```

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

Finally:

```

1242 \NewDocumentEnvironment { module } { 0{} m } {
1243   \bool_if:NT \c_stex_showmods_bool {
1244     \begin{mdframed}
1245   }
1246   \begin{@module}[#1]{#2}
1247   \stex_modules_heading:
1248 }{
1249   \end{@module}
1250   \bool_if:NT \c_stex_showmods_bool {
1251     \end{mdframed}
1252   }
1253 }

```

## 21.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n
1254 \NewDocumentCommand \STEXModule { m } {
1255   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1256   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1257   \tl_set:Nn \l_tmpa_tl {
1258     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1259   }
1260   \seq_map_inline:Nn \l_stex_all_modules_seq {
1261     \str_set:Nn \l_tmpb_str { ##1 }
1262     \str_if_eq:eeT { \l_tmpa_str } {
1263       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1264     } {
1265       \seq_map_break:n {
1266         \tl_set:Nn \l_tmpa_tl {
1267           \stex_invoke_module:n { ##1 }
1268         }

```

```

1269     }
1270   }
1271 }
1272 \l_tmpa_tl
1273 }
1274
1275 \cs_new_protected:Nn \stex_invoke_module:n {
1276   \stex_debug:nn{modules}{Invoking~module~#1}
1277   \peek_charcode_remove:NTF ! {
1278     \__stex_modules_invoke_uri:nN { #1 }
1279   } {
1280     \peek_charcode_remove:NTF ? {
1281       \__stex_modules_invoke_symbol:nn { #1 }
1282     } {
1283       \msg_error:nnx{stex}{error/syntax}{
1284         ?~or~!~expected~after~
1285         \c_backslash_str STEXModule{#1}
1286       }
1287     }
1288   }
1289 }
1290
1291 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1292   \str_set:Nn #2 { #1 }
1293 }
1294
1295 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1296   \stex_invoke_symbol:n{#1?#2}
1297 }

```

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

`\stex_activate_module:n`

```

1298 \cs_new_protected:Nn \stex_activate_module:n {
1299   \stex_debug:nn{modules}{Activating~module~#1}
1300   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1301     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1302     \prop_item:cn { c_stex_module_#1_prop } { content }
1303   }
1304 }

```

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

```

1305 </package>

```

## Chapter 22

# STEX -Module Inheritance Implementation

```
1306 <*package>
1307
1308 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1309
```

### 22.1 SMS Mode

```
1310 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1311 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1312 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1313 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1314
1315 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1316   \makeatletter
1317   \makeatother
1318   \ExplSyntaxOn
1319   \ExplSyntaxOff
1320 }
1321
1322 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1323   \symdef
1324   \importmodule
1325   \notation
1326   \symdecl
1327   \STEXexport
1328 }
1329
1330 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1331   \tl_to_str:n {
1332     module,
1333     @module
```

```

1334 }
1335 }

```

(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
1336 \bool_new:N \g__stex_smsmode_bool
1337 \bool_set_false:N \g__stex_smsmode_bool
1338 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1339   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1340 }

```

(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
1341 \bool_new:N \g__stex_smsmode_catcode_bool
1342 \bool_set_false:N \g__stex_smsmode_catcode_bool
1343 \prg_new_conditional:Nnn \_stex_smsmode_if_catcodes: { p, T, F, TF } {
1344   \bool_if:NTF \g__stex_smsmode_catcode_bool
1345   \prg_return_true: \prg_return_false:
1346 }

```

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

`\stex_smsmode_set_codes:`

```

1347 \cs_new_protected:Nn \stex_smsmode_set_codes: {
1348   \stex_if_smsmode:T {
1349     \_stex_smsmode_if_catcodes:F {
1350       \bool_gset_true:N \g__stex_smsmode_catcode_bool
1351       \exp_after:wN \char_gset_active_eq:NN
1352       \c_backslash_str \_stex_smsmode_cs:
1353       \tex_global:D \char_set_catcode_active:N \
1354       \tex_global:D \char_set_catcode_other:N $
1355       \tex_global:D \char_set_catcode_other:N ^
1356       \tex_global:D \char_set_catcode_other:N _
1357       \tex_global:D \char_set_catcode_other:N &
1358       \tex_global:D \char_set_catcode_other:N ##
1359     }
1360   }
1361 } \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.

```

1362 \cs_new_protected:Nn \_stex_smsmode_unset_codes: {
1363   \_stex_smsmode_if_catcodes:T {
1364     \bool_gset_false:N \g__stex_smsmode_catcode_bool
1365     \exp_after:wN \tex_global:D \exp_after:wN
1366     \char_set_catcode_escape:N \c_backslash_str
1367     \tex_global:D \char_set_catcode_math_toggle:N $
1368     \tex_global:D \char_set_catcode_math_superscript:N ^
1369     \tex_global:D \char_set_catcode_math_subscript:N _
1370     \tex_global:D \char_set_catcode_alignment:N &
1371     \tex_global:D \char_set_catcode_parameter:N ##
1372   }
1373 } \iffalse $ \fi % to make syntax highlighting work again

```

(End definition for `\_stex_smsmode_unset_codes:`.)

`\stex_in_smsmode:nn`

```

1374 \cs_new_protected:Nn \stex_in_smsmode:nn {
1375   \vbox_set:Nn \l_tmpa_box {
1376     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1377     \bool_gset_true:N \g__stex_smsmode_bool
1378     \stex_smsmode_set_codes:
1379     #2
1380     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1381     \stex_if_smsmode:F {
1382       \__stex_smsmode_unset_codes:
1383     }
1384   }
1385   \box_clear:N \l_tmpa_box
1386 }

```

(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:

```

1387 \cs_new_protected:Nn \_stex_smsmode_cs: {
1388   \str_clear:N \l_tmpa_str
1389   \peek_analysis_map_inline:n {
1390     % #1: token (one expansion)
1391     % #2: charcode
1392     % #3 catcode
1393     \token_if_eq_charcode:NNTF ##3 B {
1394       % token is a letter
1395       \exp_args:NNNo \str_put_right:Nn \l_tmpa_str { ##1 }
1396     } {
1397       \str_if_empty:NTF \l_tmpa_str {
1398         % we don't allow (or need) single non-letter CSs
1399         % for now
1400         \peek_analysis_map_break:
1401       }{
1402         \str_if_eq:onTF \l_tmpa_str { begin } {
1403           \peek_analysis_map_break:n {
1404             \exp_after:wN \_stex_smsmode_checkbegin:n ##1
1405           }
1406         } {
1407           \str_if_eq:onTF \l_tmpa_str { end } {
1408             \peek_analysis_map_break:n {
1409               \exp_after:wN \_stex_smsmode_checkend:n ##1
1410             }
1411           } {
1412             \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
1413             \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1414             \g_stex_smsmode_allowedmacros_tl
1415             { \use:c{\l_tmpa_str} } {
1416               \stex_debug:nn{modules}{Executing-1:~\l_tmpa_str}
1417               \peek_analysis_map_break:n {
1418                 \exp_after:wN \l_tmpa_tl ##1
1419               }

```

```

1420     } {
1421         \exp_args:NNo \exp_args:NNo \tl_if_in:NnTF
1422         \g_stex_smsmode_allowedmacros_escape_tl
1423         { \use:c{\l_tmpa_str} } {
1424             \__stex_smsmode_unset_codes:
1425             \stex_debug:nn{modules}{Executing~2:~\l_tmpa_str}
1426             % TODO \__stex_smsmode_rescan_cs:
1427             \int_compare:nNnTF {##2} = {92} {
1428                 \peek_analysis_map_break:n {
1429                     \__stex_smsmode_unset_codes:
1430                     \__stex_smsmode_rescan_cs:
1431                 }
1432             } {
1433                 \peek_analysis_map_break:n {
1434                     \exp_after:wN \l_tmpa_tl ##1
1435                 }
1436             }
1437         } {
1438             \int_compare:nNnTF {##2} = {92} {
1439                 \peek_analysis_map_break:n { \__stex_smsmode_cs: }
1440             } {
1441                 \peek_analysis_map_break:n { \exp_after:wN\relax ##1 }
1442             }
1443         }
1444     }
1445 }
1446 }
1447 }
1448 }
1449 }
1450 }

```

(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:

```

1451 \cs_new_protected:Nn \__stex_smsmode_rescan_cs: {
1452     \str_clear:N \l_tmpb_str
1453     \peek_analysis_map_inline:n {
1454         \token_if_eq_charcode:NNTF ##3 B {
1455             % token is a letter
1456             \exp_args:NNo \str_put_right:Nn \l_tmpb_str { ##1 }
1457         } {
1458             \peek_analysis_map_break:n {
1459                 \exp_after:wN \use:c \exp_after:wN {
1460                     \exp_after:wN \l_tmpa_str\exp_after:wN
1461                 } \use:c { \l_tmpb_str \exp_after:wN } ##1
1462             }
1463         }
1464     }
1465 }

```

(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.

```

1466 \cs_new_protected:Nn \__stex_smsmode_checkbegin:n {
1467   \str_set:Nn \l_tmpa_str { #1 }
1468   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1469     \__stex_smsmode_unset_codes:
1470     \begin{#1}
1471   }
1472 }
```

(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.

```

1473 \cs_new_protected:Nn \__stex_smsmode_checkend:n {
1474   \str_set:Nn \l_tmpa_str { #1 }
1475   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1476     \end{#1}
1477   }
1478 }
```

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

## 22.2 Inheritance

1479 `<@@=stex_importmodule>`

`\stex_import_module_uri:nn`

```

1480 \cs_new_protected:Nn \stex_import_module_uri:nn {
1481   \str_set:Nx \l__stex_importmodule_archive_str { #1 }
1482   \str_set:Nn \l__stex_importmodule_path_str { #2 }
1483
1484   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l__stex_importmodule_path_str }
1485   \seq_pop_right:NN \l_tmpb_seq \l__stex_importmodule_name_str
1486   \str_set:Nx \l__stex_importmodule_path_str { \seq_use:Nn \l_tmpb_seq ? }
1487
1488   \stex_modules_current_namespace:
1489   \bool_lazy_all:nTF {
1490     {\str_if_empty_p:N \l__stex_importmodule_archive_str}
1491     {\str_if_empty_p:N \l__stex_importmodule_path_str}
1492     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } }
1493   }{
1494     \str_set_eq:NN \l__stex_importmodule_path_str \l_stex_modules_subpath_str
1495     \str_set_eq:NN \l_stex_module_ns
1496   }{
1497     \str_if_empty:NT \l__stex_importmodule_archive_str {
1498       \prop_if_empty:NF \l_stex_current_repository_prop {
1499         \prop_get:NnN \l_stex_current_repository_prop { id } \l__stex_importmodule_archive_s
1500       }
1501     }
1502     \str_if_empty:NTF \l__stex_importmodule_archive_str {
1503       \str_if_empty:NF \l__stex_importmodule_path_str {
1504         \str_set:Nx \l_stex_module_ns_str {
1505           \l_stex_module_ns_str / \l__stex_importmodule_path_str
1506         }
1507       }
1508     }
```

```

1508   }{
1509     \stex_require_repository:n \l__stex_importmodule_archive_str
1510     \prop_get:cnN { c_stex_mathhub\_l__stex_importmodule_archive_str _manifest_prop } { ns
1511       \l_stex_module_ns_str
1512     \str_if_empty:NF \l__stex_importmodule_path_str {
1513       \str_set:Nx \l_stex_module_ns_str {
1514         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1515       }
1516     }
1517   }
1518 }
1519 }

```

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

<code>\l__stex_importmodule_name_str</code>	Store the return values of <code>\stex_import_module_uri:nn</code> .
<code>\l__stex_importmodule_archive_str</code>	1520 <code>\str_new:N \l__stex_importmodule_name_str</code>
<code>\l__stex_importmodule_path_str</code>	1521 <code>\str_new:N \l__stex_importmodule_archive_str</code>
<code>\l__stex_importmodule_file_str</code>	1522 <code>\str_new:N \l__stex_importmodule_path_str</code>
	1523 <code>\str_new:N \g__stex_importmodule_file_str</code>

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

```

\stex_import_require_module:nnnnn    {<ns>} {<archive-ID>} {<path>} {<name>}
1524 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1525   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1526
1527     % archive
1528     \str_set:Nx \l_tmpa_str { #2 }
1529     \str_if_empty:NTF \l_tmpa_str {
1530       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1531     } {
1532       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1533       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1534       \seq_put_right:Nn \l_tmpa_seq { source }
1535     }
1536
1537     % path
1538     \str_set:Nx \l_tmpb_str { #3 }
1539     \str_if_empty:NTF \l_tmpb_str {
1540       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1541
1542       \ltx@ifpackageloaded{babel} {
1543         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1544           { \language } \l_tmpb_str {
1545           \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1546         }
1547       } {
1548         \str_clear:N \l_tmpb_str
1549       }
1550
1551       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1552       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1553         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }

```



```

1554     }{
1555       \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1556       \IfFileExists{ \l_tmpa_str.tex }{
1557         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1558       }{
1559         % try english as default
1560         \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1561         \IfFileExists{ \l_tmpa_str.en.tex }{
1562           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1563         }{
1564           \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1565         }
1566       }
1567     }
1568
1569   } {
1570     \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1571     \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1572
1573     \ltx@ifpackageloaded{babel} {
1574       \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1575       { \language } \l_tmpb_str {
1576         \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1577       }
1578     } {
1579       \str_clear:N \l_tmpb_str
1580     }
1581
1582     \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1583
1584     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1585     \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1586       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1587     }{
1588       \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1589       \IfFileExists{ \l_tmpa_str/#4.tex }{
1590         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1591       }{
1592         % try english as default
1593         \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1594         \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1595           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1596         }{
1597           \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1598           \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1599             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1600           }{
1601             \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1602             \IfFileExists{ \l_tmpa_str.tex }{
1603               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1604             }{
1605               % try english as default
1606               \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1607               \IfFileExists{ \l_tmpa_str.en.tex }{

```

```

1608         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1609     }{
1610         \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1611     }
1612 }
1613 }
1614 }
1615 }
1616 }
1617 }
1618
1619 \seq_set_eq:NN \l_tmpa_seq \g_stex_modules_in_file_seq
1620 \seq_clear:N \g_stex_modules_in_file_seq
1621 % \exp_args:Nnx \use:nn {
1622     \exp_args:No \stex_in_smsmode:nn { \g__stex_importmodule_file_str } {
1623         \seq_clear:N \l_stex_all_modules_seq
1624         \prop_clear:N \l_stex_current_module_prop
1625         \str_set:Nx \l_tmpb_str { #2 }
1626         \str_if_empty:NF \l_tmpb_str {
1627             \stex_set_current_repository:n { #2 }
1628         }
1629         \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1630         \input { \g__stex_importmodule_file_str }
1631     }
1632 % }{
1633
1634 % }
1635 \prop_gput:Noo \g_stex_module_files_prop
1636 \g__stex_importmodule_file_str \g_stex_modules_in_file_seq
1637 \seq_set_eq:NN \g_stex_modules_in_file_seq \l_tmpa_seq
1638
1639 \stex_if_module_exists:nF { #1 ? #4 } {
1640     \msg_error:nnx{stex}{error/unknownmodule}{
1641         #1?#4~(in~file~\g__stex_importmodule_file_str)
1642     }
1643 }
1644 }
1645 \stex_activate_module:n { #1 ? #4 }
1646 }

```

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

## `\importmodule`

```

1647 \NewDocumentCommand \importmodule { 0{} m } {
1648     \stex_import_module_uri:nn { #1 } { #2 }
1649     \stex_debug:nn{modules}{Importing~module:~
1650         \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1651     }
1652     \stex_if_smsmode:F {
1653         \stex_import_require_module:nnnn
1654         { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1655         { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1656         \stex_annotate_invisible:nnn
1657         {import} { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } {}

```

```

1658 }
1659 \exp_args:Nx \stex_add_to_current_module:n {
1660   \stex_import_require_module:nnnn
1661   { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1662   { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1663 }
1664 \exp_args:Nx \stex_add_import_to_current_module:n {
1665   \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1666 }
1667 \stex_smsmode_set_codes:
1668 }
1669 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

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

### `\usemodule`

```

1670 \NewDocumentCommand \usemodule { 0{} m } {
1671   \stex_if_smsmode:F {
1672     \stex_import_module_uri:nn { #1 } { #2 }
1673     \stex_import_require_module:nnnn
1674     { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1675     { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1676     \stex_annotate_invisible:nnn
1677     {usemodule} { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } {}
1678   }
1679   \stex_smsmode_set_codes:
1680 }

```

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

```

1681 \endpackage

```

## Chapter 23

# STEX -Symbols Implementation

```
1682 <*package>
1683
1684 %%%%%%%%%%%%% symbols.dtx %%%%%%%%%%%%%
1685
1686 Warnings and error messages
```

### 23.1 Symbol Declarations

```
1687 <@@=stex_symdecl>

\l_stex_all_symbols_seq Stores all available symbols
1688 \seq_new:N \l_stex_all_symbols_seq

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

\STEXsymbol

1689 \NewDocumentCommand \STEXsymbol { m } {
1690   \stex_get_symbol:n { #1 }
1691   \exp_args:No
1692   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1693 }

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

symdecl arguments:

1694 \keys_define:nn { stex / symdecl } {
1695   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1696   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1697   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1698   type      .tl_set:N = \l_stex_symdecl_type_tl ,
1699   align     .str_set:N = \l_stex_symdecl_align_str , % TODO(?)
1700   gfc       .str_set:N = \l_stex_symdecl_gfc_str , % TODO(?)
1701   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
1702   def       .tl_set:N = \l_stex_symdecl_definiens_tl
1703 }
```

```

1704
1705 \bool_new:N \l_stex_symdecl_make_macro_bool
1706
1707 \cs_new_protected:Nn \__stex_symdecl_args:n {
1708   \str_clear:N \l_stex_symdecl_name_str
1709   \str_clear:N \l_stex_symdecl_args_str
1710   \bool_set_false:N \l_stex_symdecl_local_bool
1711   \tl_clear:N \l_stex_symdecl_type_tl
1712   \tl_clear:N \l_stex_symdecl_definiens_tl
1713
1714   \keys_set:nn { stex / symdecl } { #1 }
1715 }

```

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

```

1716
1717 \NewDocumentCommand \symdecl { s O{} m } {
1718   \__stex_symdecl_args:n { #2 }
1719   \IfBooleanTF #1 {
1720     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1721   } {
1722     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1723   }
1724   \stex_symdecl_do:n { #3 }
1725   \stex_smsmode_set_codes:
1726 }
1727 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

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

```

1728 \cs_new_protected:Nn \stex_symdecl_do:n {
1729   \stex_if_in_module:F {
1730     % TODO throw error? some default namespace?
1731   }
1732
1733   \str_if_empty:NT \l_stex_symdecl_name_str {
1734     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1735   }
1736
1737   \prop_if_exist:cT { g_stex_symdecl_
1738     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1739     \prop_item:Nn \l_stex_current_module_prop {name} ?
1740     \l_stex_symdecl_name_str
1741     _prop
1742   }{
1743     % TODO throw error (beware of circular dependencies)
1744   }
1745
1746   \prop_clear:N \l_tmpa_prop
1747   \prop_put:Nnx \l_tmpa_prop { module } {
1748     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1749     \prop_item:Nn \l_stex_current_module_prop {name}
1750   }

```

```

1751 \seq_clear:N \l_tmpa_seq
1752 \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
1753 \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1754 \prop_put:Nno \l_tmpa_prop { local } \l_stex_symdecl_local_bool
1755 \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1756
1757 \exp_args:No \stex_add_constant_to_current_module:n {
1758   \l_stex_symdecl_name_str
1759 }
1760
1761 % arity/args
1762 \int_zero:N \l_tmpb_int
1763
1764 \bool_set_true:N \l_tmpa_bool
1765 \str_map_inline:Nn \l_stex_symdecl_args_str {
1766   \token_case_meaning:NnF ##1 {
1767     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1768     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1769     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1770     {\tl_to_str:n a} {
1771       \bool_set_false:N \l_tmpa_bool
1772       \int_incr:N \l_tmpb_int
1773     }
1774     {\tl_to_str:n B} {
1775       \bool_set_false:N \l_tmpa_bool
1776       \int_incr:N \l_tmpb_int
1777     }
1778   }{
1779     \msg_set:nnn{stex}{error/wrongargs}{
1780       args~value~in~symbol~declaration~for~
1781       \prop_item:Nn \l_stex_current_module_prop {ns} ?
1782       \prop_item:Nn \l_stex_current_module_prop {name} ?
1783       \l_stex_symdecl_name_str ~
1784       needs~to~be~
1785       i,~a,~b~or~B,~but~##1~given
1786     }
1787     \msg_error:nn{stex}{error/wrongargs}
1788   }
1789 }
1790 \bool_if:NTF \l_tmpa_bool {
1791   % possibly numeric
1792   \str_if_empty:NTF \l_stex_symdecl_args_str {
1793     \prop_put:Nnn \l_tmpa_prop { args } {}
1794     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1795   }{
1796     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1797     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1798     \str_clear:N \l_tmpa_str
1799     \int_step_inline:nn \l_tmpa_int {
1800       \str_put_right:Nn \l_tmpa_str i
1801     }
1802     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1803   }
1804 } {

```

```

1805     \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1806     \prop_put:Nnx \l_tmpa_prop { arity }
1807       { \str_count:N \l_stex_symdecl_args_str }
1808   }
1809   \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
1810
1811
1812   % semantic macro
1813
1814   \bool_if:NT \l_stex_symdecl_make_macro_bool {
1815     \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1816       \prop_item:Nn \l_tmpa_prop { module } ?
1817       \prop_item:Nn \l_tmpa_prop { name }
1818     } }
1819
1820     \bool_if:NF \l_stex_symdecl_local_bool {
1821       \exp_args:Nx \stex_add_to_current_module:n {
1822         \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1823           \prop_item:Nn \l_tmpa_prop { module } ?
1824           \prop_item:Nn \l_tmpa_prop { name }
1825         } }
1826       }
1827     }
1828   }
1829
1830   % add to all symbols
1831
1832   \bool_if:NF \l_stex_symdecl_local_bool {
1833     \exp_args:Nx \stex_add_to_current_module:n {
1834       \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1835         \prop_item:Nn \l_tmpa_prop { module } ?
1836         \prop_item:Nn \l_tmpa_prop { name }
1837       }
1838     }
1839   }
1840
1841   \stex_debug:nn{symbols}{New~symbol:~
1842     \prop_item:Nn \l_tmpa_prop { module } ?
1843     \prop_item:Nn \l_tmpa_prop { name } ^^J
1844     Type:~\exp_not:o { \l_stex_symdecl_type_tl } ^^J
1845     Args:~\prop_item:Nn \l_tmpa_prop { args }
1846   }
1847
1848   % circular dependencies require this:
1849
1850   \prop_if_exist:cF {
1851     g_stex_symdecl_
1852     \prop_item:Nn \l_tmpa_prop { module } ?
1853     \prop_item:Nn \l_tmpa_prop { name }
1854     _prop
1855   } {
1856     \prop_gset_eq:cN {
1857       g_stex_symdecl_
1858       \prop_item:Nn \l_tmpa_prop { module } ?

```

```

1859     \prop_item:Nn \l_tmpa_prop { name }
1860     _prop
1861   } \l_tmpa_prop
1862 }
1863
1864 \stex_if_smsmode:TF {
1865   \bool_if:NF \l_stex_symdecl_local_bool {
1866     \exp_args:Nx \stex_add_to_sms:n {
1867       \prop_gset_from_keyval:cn {
1868         g_stex_symdecl_
1869         \prop_item:Nn \l_tmpa_prop { module } ?
1870         \prop_item:Nn \l_tmpa_prop { name }
1871         _prop
1872       } {
1873         name      = \prop_item:Nn \l_tmpa_prop { name }      ,
1874         module    = \prop_item:Nn \l_tmpa_prop { module }    ,
1875         notations = \prop_item:Nn \l_tmpa_prop { notations } ,
1876         local     = \prop_item:Nn \l_tmpa_prop { local }     ,
1877         type      = \prop_item:Nn \l_tmpa_prop { type }      ,
1878         args      = \prop_item:Nn \l_tmpa_prop { args }      ,
1879         arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
1880         assocs    = \prop_item:Nn \l_tmpa_prop { assocs }
1881       }
1882       \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1883         \prop_item:Nn \l_tmpa_prop { module } ?
1884         \prop_item:Nn \l_tmpa_prop { name }
1885       }
1886     }
1887   }
1888 }{
1889   \exp_args:NNx \seq_put_right:Nn \l_stex_all_symbols_seq {
1890     \prop_item:Nn \l_tmpa_prop { module } ?
1891     \prop_item:Nn \l_tmpa_prop { name }
1892   }
1893   \stex_if_do_html:T {
1894     \stex_annotate_invisible:nnn {symdecl} {
1895       \prop_item:Nn \l_tmpa_prop { module } ?
1896       \prop_item:Nn \l_tmpa_prop { name }
1897     } {
1898       \stex_annotate_invisible:nnn{type}{}{\l_stex_symdecl_type_tl$}
1899       \stex_annotate_invisible:nnn{args}{}{
1900         \prop_item:Nn \l_tmpa_prop { args }
1901       }
1902       \stex_annotate_invisible:nnn{macroname}{}{#1}
1903       \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
1904         \stex_annotate_invisible:nnn{definiens}{}{
1905           {\l_stex_symdecl_definiens_tl$}
1906         }
1907       }
1908     }
1909   }
1910 }

```

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



`\stex_get_symbol:n`

```
1911 \str_new:N \l_stex_get_symbol_uri_str
1912
1913 \cs_new_protected:Nn \stex_get_symbol:n {
1914   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
1915     \__stex_symdecl_get_symbol_from_cs:n { #1 }
1916   }{
1917     % argument is a string
1918     % is it a command name?
1919     \cs_if_exist:cTF { #1 }{
1920       \cs_set_eq:Nc \l_tmpa_tl { #1 }
1921       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
1922       \str_if_empty:NNTF \l_tmpa_str {
1923         \exp_args:Nx \cs_if_eq:NNTF {
1924           \tl_head:N \l_tmpa_tl
1925         } \stex_invoke_symbol:n {
1926           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
1927         }{
1928           \__stex_symdecl_get_symbol_from_string:n { #1 }
1929         }
1930       } {
1931         \__stex_symdecl_get_symbol_from_string:n { #1 }
1932       }
1933     }{
1934       % argument is not a command name
1935       \__stex_symdecl_get_symbol_from_string:n { #1 }
1936       % \l_stex_all_symbols_seq
1937     }
1938   }
1939 }
1940
1941 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
1942   \str_set:Nn \l_tmpa_str { #1 }
1943   \bool_set_false:N \l_tmpa_bool
1944   \stex_if_in_module:T {
1945     \prop_get:NnN \l_stex_current_module_prop
1946     { constants } \l_tmpa_seq
1947     \exp_args:NNo \seq_if_in:NnT \l_tmpa_seq { \l_tmpa_str } {
1948       \bool_set_true:N \l_tmpa_bool
1949       \str_set:Nx \l_stex_get_symbol_uri_str {
1950         \prop_item:Nn \l_stex_current_module_prop { ns } ?
1951         \prop_item:Nn \l_stex_current_module_prop { name } ? #1
1952       }
1953     }
1954   }
1955   \bool_if:NF \l_tmpa_bool {
1956     \tl_set:Nn \l_tmpa_tl {
1957       \msg_set:nnn{stex}{error/unknownsymbol}{
1958         No~symbol~#1~found!
1959       }
1960     }
1961     \msg_error:nn{stex}{error/unknownsymbol}
1962   }
1963   \str_set:Nn \l_tmpa_str { #1 }
1964   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
```

```

1964 \seq_map_inline:Nn \l_stex_all_symbols_seq {
1965   \str_set:Nn \l_tmpb_str { ##1 }
1966   \str_if_eq:eeT { \l_tmpa_str } {
1967     \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1968   } {
1969     \seq_map_break:n {
1970       \tl_set:Nn \l_tmpa_tl {
1971         \str_set:Nn \l_stex_get_symbol_uri_str {
1972           ##1
1973         }
1974       }
1975     }
1976   }
1977 }
1978 \l_tmpa_tl
1979 }
1980 }
1981
1982 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
1983   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
1984     { \tl_tail:N \l_tmpa_tl }
1985   \tl_if_single:NTF \l_tmpa_tl {
1986     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
1987       \exp_after:wN \str_set:Nn \exp_after:wN
1988         \l_stex_get_symbol_uri_str \l_tmpa_tl
1989     }{
1990       % TODO
1991       % tail is not a single group
1992     }
1993   }{
1994     % TODO
1995     % tail is not a single group
1996   }
1997 }

```

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

## 23.2 Notations

```

1998 <@@=stex_notation>
1999 notation arguments:
2000 \keys_define:nn { stex / notation } {
2001   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2002   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2003   prec .str_set_x:N = \l__stex_notation_prec_str ,
2004   op .tl_set:N = \l__stex_notation_op_tl ,
2005   unknown .code:n = \str_set:Nx
2006     \l__stex_notation_variant_str \l_keys_key_str
2007 }
2008
2009 \cs_new_protected:Nn \__stex_notation_args:n {
2010   \str_clear:N \l__stex_notation_lang_str
2011   \str_clear:N \l__stex_notation_variant_str

```

```

2011 \str_clear:N \l__stex_notation_prec_str
2012 \tl_clear:N \l__stex_notation_op_tl
2013
2014 \keys_set:nn { stex / notation } { #1 }
2015 }

```

## **\notation**

```

2016 \NewDocumentCommand \notation { 0{ } m } {
2017   \__stex_notation_args:n { #1 }
2018   \tl_clear:N \l_stex_symdecl_definiens_tl
2019   \stex_get_symbol:n { #2 }
2020   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
2021 }
2022 \stex_deactivate_macro:Nn \notation {module~environments}

```

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

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

```

2023 \cs_new_protected:Nn \stex_notation_do:nn {
2024   \prop_set_eq:Nc \l_tmpa_prop {
2025     g_stex_symdecl_ #1 _prop
2026   }
2027
2028   \prop_clear:N \l_tmpb_prop
2029   \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
2030   \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
2031   \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
2032
2033   % precedences
2034   \seq_clear:N \l_tmpb_seq
2035   \exp_args:NNno
2036   \str_if_empty:NTF \l__stex_notation_prec_str {
2037     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2038     \int_compare:nNnTF \l_tmpa_str = 0 {
2039       \exp_args:NNnx
2040       \prop_put:Nno \l_tmpb_prop { opprec }
2041       { \neginfprec }
2042     }{
2043       \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
2044     }
2045   } {
2046     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2047       \exp_args:NNnx
2048       \prop_put:Nno \l_tmpb_prop { opprec }
2049       { \neginfprec }
2050       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2051       \int_step_inline:nn { \l_tmpa_str } {
2052         \exp_args:NNx
2053         \seq_put_right:Nn \l_tmpb_seq { \infprec }
2054       }
2055     }{
2056       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2057       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2058         \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
2059         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {

```

```

2060         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2061         \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2062         \seq_map_inline:Nn \l_tmpa_seq {
2063             \seq_put_right:Nn \l_tmpb_seq { ##1 }
2064         }
2065     }
2066     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2067 }{
2068     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2069     \int_compare:nNnTF \l_tmpa_str = 0 {
2070         \exp_args:NNnx
2071         \prop_put:Nno \l_tmpb_prop { opprec }
2072         { \infprec }
2073     }{
2074         \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
2075     }
2076 }
2077 }
2078 }
2079
2080 \seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
2081 \int_step_inline:nn { \l_tmpa_str } {
2082     \seq_pop_left:NnF \l_tmpa_seq \l_tmpb_str {
2083         \exp_args:NNx
2084         \seq_put_right:Nn \l_tmpb_seq {
2085             \prop_item:Nn \l_tmpb_prop { opprec }
2086         }
2087     }
2088 }
2089
2090 \prop_put:Nno \l_tmpb_prop { argprec } \l_tmpb_seq
2091 \tl_clear:N \l_tmpa_tl
2092
2093 \int_compare:nNnTF \l_tmpa_str = 0 {
2094     \exp_args:NNe
2095     \cs_set:Npn \l__stex_notation_macrocode_cs {
2096         \_stex_term_math_oms:nnnn { #1 }
2097         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2098         { \prop_item:Nn \l_tmpb_prop { opprec } }
2099         { \exp_not:n { #2 } }
2100     }
2101     \__stex_notation_final:
2102 }{
2103     \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
2104     \str_if_in:NnTF \l_tmpb_str b {
2105         \exp_args:Nne \use:nn
2106         {
2107             \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2108             \cs_set:Npn \l_tmpa_str { {
2109                 \_stex_term_math_omb:nnnn { #1 }
2110                 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2111                 { \prop_item:Nn \l_tmpb_prop { opprec } }
2112                 { \exp_not:n { #2 } }
2113             }}

```

```

2114   }{
2115     \str_if_in:NnTF \l_tmpb_str B {
2116       \exp_args:Nne \use:nn
2117       {
2118         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2119         \cs_set:Npn \l_tmpa_str } { {
2120           \stex_term_math_omb:nnnn { #1 }
2121           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2122           { \prop_item:Nn \l_tmpb_prop { opprec } }
2123           { \exp_not:n { #2 } }
2124         } }
2125       }{
2126         \exp_args:Nne \use:nn
2127         {
2128           \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2129           \cs_set:Npn \l_tmpa_str } { {
2130             \stex_term_math_oma:nnnn { #1 }
2131             { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2132             { \prop_item:Nn \l_tmpb_prop { opprec } }
2133             { \exp_not:n { #2 } }
2134           } }
2135         }
2136       }
2137
2138     \int_zero:N \l_tmpa_int
2139     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2140     \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2141     \__stex_notation_arguments:
2142   }
2143 }

```

(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

```

2144 \cs_new_protected:Nn \__stex_notation_arguments: {
2145   \int_incr:N \l_tmpa_int
2146   \str_if_empty:NNTF \l_tmpa_str {
2147     \__stex_notation_final:
2148   }{
2149     \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2150     \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2151     \str_if_eq:VnTF \l_tmpb_str a {
2152       \__stex_notation_argument_assoc:n
2153     }{
2154       \str_if_eq:VnTF \l_tmpb_str B {
2155         \__stex_notation_argument_assoc:n
2156       }{
2157         \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2158         \tl_put_right:Nx \l_tmpa_tl {
2159           { \stex_term_math_arg:nnn
2160             { \int_use:N \l_tmpa_int }
2161             { \l_tmpb_str }
2162             { ####\int_use:N \l_tmpa_int }
2163           }
2164         }
2165       }
2166     }
2167   }

```

```

2164     }
2165     \__stex_notation_arguments:
2166   }
2167 }
2168 }
2169 }

```

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

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

```

2170 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
2171   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2172   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
2173   \tl_put_right:Nx \l_tmpa_tl {
2174     { \stex_term_math_assoc_arg:nnnn
2175       { \int_use:N \l_tmpa_int }
2176       { \l_tmpb_str }
2177       \exp_args:No \exp_not:n
2178       {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
2179       { ####\int_use:N \l_tmpa_int }
2180     }
2181   }
2182   \__stex_notation_arguments:
2183 }

```

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

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

```

2184 \cs_new_protected:Nn \__stex_notation_final: {
2185   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
2186   \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
2187   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2188   \exp_args:Nne \use:nn
2189   {
2190     \cs_generate_from_arg_count:cNnn {
2191       stex_notation_ \l_tmpa_str \c_hash_str
2192       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2193       _cs
2194     }
2195     \cs_gset:Npn \l_tmpb_str { { {
2196       \exp_after:wN \exp_after:wN \exp_after:wN
2197       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2198       { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2199     } } }
2200
2201     \tl_if_empty:NF \l__stex_notation_op_tl {
2202       \cs_gset:cpx {
2203         stex_op_notation_ \l_tmpa_str \c_hash_str
2204         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2205         _cs
2206       } {
2207         \stex_term_oms:nnn {
2208           \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2209           \l__stex_notation_lang_str

```

```

2210     }{
2211         \l_tmpa_str
2212     }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2213 }
2214 }
2215
2216
2217
2218 \stex_debug:nn{symbols}{
2219     Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2220     ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
2221     Operator~precedence:~
2222     \prop_item:Nn \l_tmpb_prop { opprec }^^J
2223     Argument~precedences:~
2224     \seq_use:Nn \l_tmpa_seq {,~}^^J
2225     Notation: \cs_meaning:c {
2226         stex_notation_ \l_tmpa_str \c_hash_str
2227         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2228         _cs
2229     }
2230 }
2231
2232 \prop_gset_eq:cN {
2233     g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2234     \c_hash_str \l__stex_notation_lang_str _prop
2235 } \l_tmpb_prop
2236
2237 \exp_args:Nx
2238 \stex_add_to_current_module:n {
2239     \prop_get:cnN {
2240         g_stex_symdecl_
2241         \prop_item:Nn \l_tmpb_prop { symbol }
2242         _prop
2243     } { notations } \exp_not:N \l_tmpa_seq
2244     \seq_put_right:Nn \exp_not:N \l_tmpa_seq {
2245         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2246     }
2247     \prop_put:cno {
2248         g_stex_symdecl_
2249         \prop_item:Nn \l_tmpb_prop { symbol }
2250         _prop
2251     } { notations } \exp_not:N \l_tmpa_seq
2252 }
2253
2254 \stex_if_smsmode:TF {
2255     \stex_smsmode_set_codes:
2256     \exp_args:Nx \stex_add_to_sms:n {
2257         \prop_gset_from_keyval:cn {
2258             g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2259             \c_hash_str \l__stex_notation_lang_str _prop
2260         } {
2261             symbol = \prop_item:Nn \l_tmpb_prop { symbol } ,
2262             language = \prop_item:Nn \l_tmpb_prop { language } ,
2263             variant = \prop_item:Nn \l_tmpb_prop { variant } ,

```

```

2264         opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2265         argprec     = \prop_item:Nn \l_tmpb_prop { argprec }     ,
2266     }
2267 }
2268 }{
2269   \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
2270   \seq_put_right:Nx \l_tmpa_seq {
2271     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2272   }
2273   \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
2274   \prop_set_eq:cN {
2275     g_stex_symdecl_ \l_tmpa_str _prop
2276   } \l_tmpa_prop
2277
2278   % HTML annotations
2279   \stex_if_do_html:T {
2280     \stex_annotate_invisible:nnn { notation }
2281     { \prop_item:Nn \l_tmpb_prop { symbol } } {
2282       \stex_annotate_invisible:nnn { notationfragment }
2283       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
2284       \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2285       \stex_annotate_invisible:nnn { precedence }
2286       { \prop_item:Nn \l_tmpb_prop { opprec } ;
2287         \seq_use:Nn \l_tmpa_seq { x }
2288       }{}
2289
2290       \int_zero:N \l_tmpa_int
2291       \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2292       \tl_clear:N \l_tmpa_tl
2293       \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
2294         \int_incr:N \l_tmpa_int
2295         \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2296         \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2297         \str_if_eq:VnTF \l_tmpb_str a {
2298           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2299             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2300             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2301           } }
2302         }{
2303           \str_if_eq:VnTF \l_tmpb_str B {
2304             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2305               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2306               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2307             } }
2308           }{
2309             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2310               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2311             } }
2312           }
2313         }
2314       }
2315       \stex_annotate_invisible:nnn { notationcomp }{}{
2316         $ \exp_args:Nno \use:nn { \use:c {
2317           stex_notation_ \prop_item:Nn \l_tmpb_prop { symbol }

```



```

2318         \c_hash_str \l__stex_notation_variant_str
2319         \c_hash_str \l__stex_notation_lang_str_cs
2320     } } { \l_tmpa_tl } $
2321   }
2322 }
2323 }
2324 }
2325 }

```

(End definition for `\__stex_notation_final:`)

**\symdef**

```

2326 \keys_define:nn { stex / symdef } {
2327   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2328   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2329   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2330   type      .tl_set:N   = \l_stex_symdecl_type_tl ,
2331   def       .tl_set:N   = \l_stex_symdecl_definiens_tl ,
2332   op        .tl_set:N   = \l__stex_notation_op_tl ,
2333   lang      .str_set_x:N = \l__stex_notation_lang_str ,
2334   variant   .str_set_x:N = \l__stex_notation_variant_str ,
2335   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2336   unknown   .code:n     = \str_set:Nx
2337             \l__stex_notation_variant_str \l_keys_key_str
2338 }
2339
2340 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2341   \str_clear:N \l_stex_symdecl_name_str
2342   \str_clear:N \l_stex_symdecl_args_str
2343   \bool_set_false:N \l_stex_symdecl_local_bool
2344   \tl_clear:N \l_stex_symdecl_type_tl
2345   \tl_clear:N \l_stex_symdecl_definiens_tl
2346   \str_clear:N \l__stex_notation_lang_str
2347   \str_clear:N \l__stex_notation_variant_str
2348   \str_clear:N \l__stex_notation_prec_str
2349   \tl_clear:N \l__stex_notation_op_tl
2350
2351   \keys_set:nn { stex / symdef } { #1 }
2352 }
2353
2354 \NewDocumentCommand \symdef { 0{} m } {
2355   \__stex_notation_symdef_args:n { #1 }
2356   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2357   \stex_symdecl_do:n { #2 }
2358   \exp_args:Nx \stex_notation_do:nn {
2359     \prop_item:Nn \l_tmpa_prop { module } ?
2360     \prop_item:Nn \l_tmpa_prop { name }
2361   }
2362 }
2363 \stex_deactivate_macro:Nn \symdef {module~environments}

```

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

```

2364 \</package>

```

## Chapter 24

# STEX -Terms Implementation

```
2365 <*package>
2366
2367 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2368
2369 <@@=stex_terms>
2370
2371 Warnings and error messages
2372 \msg_new:nnn{stex}{error/nonotation}{
2373   Symbol~#1~invoked,~but~has~no~notation~#2!
2374 }
2375 \msg_new:nnn{stex}{error/notationarg}{
2376   Error~in~parsing~notation~#1
2377 }
2378 \msg_new:nnn{stex}{error/noop}{
2379   Symbol~#1~has~no~operator~notation~for~notation~#2
2380 }
```

### 24.1 Symbol Invocations

Arguments:

```
2380 \keys_define:nn { stex / terms } {
2381   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2382   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2383   unknown .code:n = \str_set:Nx
2384     \l__stex_terms_variant_str \l_keys_key_str
2385 }
2386
2387 \cs_new_protected:Nn \__stex_terms_args:n {
2388   \str_clear:N \l__stex_terms_lang_str
2389   \str_clear:N \l__stex_terms_variant_str
2390   \str_clear:N \l__stex_terms_prec_str
2391   \tl_clear:N \l__stex_terms_op_tl
2392 }
2393 \keys_set:nn { stex / terms } { #1 }
```

2394 }

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

```
2395 \cs_new_protected:Nn \stex_invoke_symbol:n {
2396   \if_mode_math:
2397     \exp_after:wN \__stex_terms_invoke_math:n
2398   \else:
2399     \exp_after:wN \__stex_terms_invoke_text:n
2400   \fi: { #1 }
2401 }
```

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

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

```
2402 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2403   \peek_charcode_remove:NTF ! {
2404     \peek_charcode:NTF [ {
2405       \__stex_terms_invoke_op:nw { #1 }
2406     }{
2407       \peek_charcode_remove:NTF ! {
2408         \peek_charcode:NTF [ {
2409           \__stex_terms_invoke_op_custom:nw
2410         }{
2411           % TODO throw error
2412         }
2413       }{
2414         \__stex_terms_invoke_op:nw { #1 } []
2415       }
2416     }
2417   }{
2418     \peek_charcode_remove:NTF * {
2419       \__stex_terms_invoke_text:n { #1 }
2420     }{
2421       \peek_charcode:NTF [ {
2422         \__stex_terms_invoke_math:nw { #1 }
2423       }{
2424         \__stex_terms_invoke_math:nw { #1 } []
2425       }
2426     }
2427   }
2428 }
```

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

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

```
2429 \cs_new_protected:Npn \__stex_terms_invoke_op_custom:nw #1 [#2] {
2430   \stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
2431     \stex_highlight_term:nn{#1}{#2}
2432   }
2433 }
```

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

\\_stex\_terms\_invoke\_op:nw

```

2434 \cs_new_protected:Npn \_stex_terms_invoke_op:nw #1 [#2] {
2435   \_stex_terms_args:n { #2 }
2436   \cs_if_exist:cTF {
2437     stex_op_notation_ #1 \c_hash_str
2438     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2439   }{
2440     \csname stex_op_notation_ #1 \c_hash_str
2441       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2442     \endcsname
2443   }{
2444     \msg_error:nnxx{stex}{error/noop}{#1}{\l__stex_terms_variant_str \c_hash_str \l__stex_te
2445   }
2446 }

```

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

\\_stex\_terms\_invoke\_math:nw

```

2447 \cs_new_protected:Npn \_stex_terms_invoke_math:nw #1 [#2] {
2448   \_stex_terms_args:n { #2 }
2449   \prop_set_eq:Nc \l_tmpa_prop {
2450     g_stex_symdecl_ #1 _prop
2451   }
2452   \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
2453   \seq_if_empty:NTF \l_tmpa_seq {
2454     \msg_error:nnxn{stex}{error/nonotation}{#1}{s}
2455   } {
2456     \seq_if_in:NxTF \l_tmpa_seq
2457     { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
2458       \use:c{
2459         stex_notation_ #1 \c_hash_str
2460         \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2461         _cs
2462       }
2463     }{
2464       \str_if_empty:NTF \l__stex_terms_variant_str {
2465         \str_if_empty:NTF \l__stex_terms_lang_str {
2466           \seq_get_left:NN \l_tmpa_seq \l_tmpa_str
2467           \use:c{
2468             stex_notation_ #1 \c_hash_str \l_tmpa_str
2469             _cs
2470           }
2471         }{
2472           \msg_error:nnxx{stex}{error/nonotation}{#1}{
2473             ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2474           }
2475         }
2476       }{
2477         \msg_error:nnxx{stex}{error/nonotation}{#1}{
2478           ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2479         }
2480       }
2481     }
2482 }

```

```
2483 }
(End definition for \_stex_terms_invoke_math:nw.)
```

```
\_stex_terms_invoke_text:n
2484 \cs_new_protected:Nn \_stex_terms_invoke_text:n {
2485   \peek_charcode_remove:NTF ! {
2486     \stex_term_custom:nn { #1 } { }
2487   }{
2488     \prop_set_eq:Nc \l_tmpa_prop {
2489       g_stex_symdecl_ #1 _prop
2490     }
2491     \prop_get:Nn \l_tmpa_prop { args } \l_tmpa_str
2492     \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2493   }
2494 }
(End definition for \_stex_terms_invoke_text:n.)
```

## 24.2 Terms

Precedences:

```
\infprec
\neginfprec
\l__stex_terms_downprec
2495 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2496 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2497 \int_new:N \l__stex_terms_downprec
2498 \int_set_eq:NN \l__stex_terms_downprec \infprec
```

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

Bracketing:

```
\l_stex_terms_left_bracket_str
\l_stex_terms_right_bracket_str
2499 \tl_set:Nn \l__stex_terms_left_bracket_str (
2500 \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

```
2501 \cs_new_protected:Nn \_stex_terms_maybe_brackets:nn {
2502   \bool_if:NTF \l__stex_terms_brackets_done_bool {
2503     \bool_set_false:N \l__stex_terms_brackets_done_bool
2504     #2
2505   } {
2506     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
2507       \bool_if:NTF \l_stex_inarray_bool { #2 }{
2508         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
2509         \dobrackets { #2 }
2510       }
2511     }{ #2 }
2512   }
2513 }
```

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

### **\dobrackets**

```
2514 \bool_new:N \l__stex_terms_brackets_done_bool
2515 %\RequirePackage{scalerel}
2516 \cs_new_protected:Npn \dobrackets #1 {
2517   %\ThisStyle{\if D\m@switch
2518   %   \exp_args:Nnx \use:nn
2519   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
2520   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
2521   % \else
2522   \exp_args:Nnx \use:nn
2523   {
2524     \bool_set_true:N \l__stex_terms_brackets_done_bool
2525     \int_set:Nn \l__stex_terms_downprec \infpref
2526     \l__stex_terms_left_bracket_str
2527     #1
2528   }
2529   {
2530     \bool_set_false:N \l__stex_terms_brackets_done_bool
2531     \l__stex_terms_right_bracket_str
2532     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2533   }
2534   %\fi}
2535 }
```

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

### **\withbrackets**

```
2536 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2537   \exp_args:Nnx \use:nn
2538   {
2539     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
2540     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
2541     #3
2542   }
2543   {
2544     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
2545     {\l__stex_terms_left_bracket_str}
2546     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
2547     {\l__stex_terms_right_bracket_str}
2548   }
2549 }
```

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

### **\STEXinvisible**

```
2550 \cs_new_protected:Npn \STEXinvisible #1 {
2551   \stex_annotate_invisible:n { #1 }
2552 }
```

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

OMDoc terms:

`\_stex_term_math_oms:nnnn`

```
2553 \cs_new_protected:Nn \_stex_term_oms:nnn {
2554   \stex_annotate:nnn{ OMID }{ #2 }{
2555     \stex_highlight_term:nn { #1 } { #3 }
2556   }
2557 }
2558
2559 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2560   \__stex_terms_maybe_brackets:nn { #3 }{
2561     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2562   }
2563 }
```

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

`\_stex_term_math_oma:nnnn`

```
2564 \cs_new_protected:Nn \_stex_term_oma:nnn {
2565   \stex_annotate:nnn{ OMA }{ #2 }{
2566     \stex_highlight_term:nn { #1 } { #3 }
2567   }
2568 }
2569
2570 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2571   \__stex_terms_maybe_brackets:nn { #3 }{
2572     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2573   }
2574 }
```

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

`\_stex_term_math_omb:nnnn`

```
2575 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2576   \stex_annotate:nnn{ OMBIND }{ #2 }{
2577     \stex_highlight_term:nn { #1 } { #3 }
2578   }
2579 }
2580
2581 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2582   \__stex_terms_maybe_brackets:nn { #3 }{
2583     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2584   }
2585 }
```

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

`\_stex_term_math_arg:nnn`

```
2586 \cs_new_protected:Nn \_stex_term_arg:nn {
2587   \stex_unhighlight_term:n {
2588     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2589   }
2590 }
2591 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
2592   \exp_args:Nnx \use:nn
2593     { \int_set:Nn \l__stex_terms_downprec { #2 }

```

```

2594     \stex_term_arg:nn { #1 }{ #3 }
2595   }
2596   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2597 }

```

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

`\stex_term_math_assoc_arg:nnnn`

```

2598 \cs_new_protected:Nn \stex_term_math_assoc_arg:nnnn {
2599   \clist_set:Nn \l_tmpa_clist{ #4 }
2600   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
2601     \tl_set:Nn \l_tmpa_tl { #4 }
2602   }{
2603     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
2604     \clist_reverse:N \l_tmpa_clist
2605     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
2606
2607     \clist_map_inline:Nn \l_tmpa_clist {
2608       \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
2609         \exp_args:Nno
2610         \l_tmpa_cs { ##1 } \l_tmpa_tl
2611       }
2612     }
2613
2614   }
2615   \exp_args:Nnno
2616   \stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2617 }

```

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

`\stex_term_custom:nn`

```

2618 \cs_new_protected:Nn \stex_term_custom:nn {
2619   \str_set:Nn \l__stex_terms_custom_uri { #1 }
2620   \str_set:Nn \l_tmpa_str { #2 }
2621   \tl_clear:N \l_tmpa_tl
2622   \int_zero:N \l_tmpa_int
2623   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2624   \__stex_terms_custom_loop:
2625 }

```

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

`\__stex_terms_custom_loop:`

```

2626 \cs_new_protected:Nn \__stex_terms_custom_loop: {
2627   \bool_set_false:N \l_tmpa_bool
2628   \bool_while_do:nn {
2629     \str_if_eq_p:ee X {
2630       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2631     }
2632   }{
2633     \int_incr:N \l_tmpa_int
2634   }
2635
2636   \peek_charcode:NNTF [ {

```



```

2637 % notation/text component
2638 \__stex_terms_custom_component:w
2639 } {
2640 \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2641 % all arguments read => finish
2642 \__stex_terms_custom_final:
2643 } {
2644 % arguments missing
2645 \peek_charcode_remove:NTF * {
2646 % invisible, specific argument position or both
2647 \peek_charcode:NTF [ {
2648 % visible specific argument position
2649 \__stex_terms_custom_arg:wn
2650 } {
2651 % invisible
2652 \peek_charcode_remove:NTF * {
2653 % invisible specific argument position
2654 \__stex_terms_custom_arg_inv:wn
2655 } {
2656 % invisible next argument
2657 \__stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2658 }
2659 }
2660 } {
2661 % next normal argument
2662 \__stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
2663 }
2664 }
2665 }
2666 }

```

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

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

```

2667 \cs_new_protected:Npn \__stex_terms_custom_arg_inv:wn [ #1 ] #2 {
2668 \bool_set_true:N \l_tmpa_bool
2669 \__stex_terms_custom_arg:wn [ #1 ] { #2 }
2670 }

```

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

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

```

2671 \cs_new_protected:Npn \__stex_terms_custom_arg:wn [ #1 ] #2 {
2672 \str_set:Nx \l_tmpb_str {
2673 \str_item:Nn \l_tmpa_str { #1 }
2674 }
2675 \str_case:VnTF \l_tmpb_str {
2676 { X } {
2677 \msg_error:nnx{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2678 }
2679 { i } { \__stex_terms_custom_set_X:n { #1 } }
2680 { b } { \__stex_terms_custom_set_X:n { #1 } }
2681 { a } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
2682 { B } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
2683 }{}{

```

```

2684 \msg_error:nnx{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2685 }
2686
2687 \bool_if:nTF \l_tmpa_bool {
2688   \tl_put_right:Nx \l_tmpa_tl {
2689     \stex_annotate_invisible:n {
2690       \stex_term_arg:nn { \int_eval:n { #1 } }
2691       \exp_not:n { { #2 } }
2692     }
2693   }
2694 } {
2695   \tl_put_right:Nx \l_tmpa_tl {
2696     \stex_term_arg:nn { \int_eval:n { #1 } }
2697     \exp_not:n { { #2 } }
2698   }
2699 }
2700
2701 \__stex_terms_custom_loop:
2702 }

```

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

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

```

2703 \cs_new_protected:Nn \__stex_terms_custom_set_X:n {
2704   \str_set:Nx \l_tmpa_str {
2705     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
2706     X
2707     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
2708   }
2709 }

```

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

\\_stex\_terms\_custom\_component:

```

2710 \cs_new_protected:Npn \_stex_terms_custom_component:w [ #1 ] {
2711   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
2712   \__stex_terms_custom_loop:
2713 }

```

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

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

```

2714 \cs_new_protected:Nn \__stex_terms_custom_final: {
2715   \int_compare:nNnTF \l_tmpb_int = 0 {
2716     \exp_args:Nnno \stex_term_oms:nnn
2717   } {
2718     \str_if_in:NnTF \l_tmpa_str {b} {
2719       \exp_args:Nnno \stex_term_ombind:nnn
2720     } {
2721       \exp_args:Nnno \stex_term_oma:nnn
2722     }
2723   }
2724   { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
2725 }

```

(End definition for `\_stex_terms_custom_final:`.)

**`\symref`**

**`\symname`**

```

2726 \NewDocumentCommand \symref { m m }{
2727   \let\compemph_uri_prev:\compemph@uri
2728   \let\compemph@uri\symrefemph@uri
2729   \STEXsymbol{#1}! [#2]
2730   \let\compemph@uri\compemph_uri_prev:
2731 }
2732
2733 \keys_define:nn { stex / symname } {
2734   post      .str_set_x:N   = \l_stex_symname_post_str
2735 }
2736
2737 \cs_new_protected:Nn \stex_symname_args:n {
2738   \str_clear:N \l_stex_symname_post_str
2739   \keys_set:nn { stex / symname } { #1 }
2740 }
2741
2742 \NewDocumentCommand \symname { 0{} m }{
2743   \stex_symname_args:n { #1 }
2744   \stex_get_symbol:n { #2 }
2745   \str_set:Nx \l_tmpa_str {
2746     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
2747   }
2748   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
2749
2750   \let\compemph_uri_prev:\compemph@uri
2751   \let\compemph@uri\symrefemph@uri
2752   \exp_args:NNx \use:nn
2753   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }! [
2754     \l_tmpa_str \l_stex_symname_post_str
2755   ] }
2756   \let\compemph@uri\compemph_uri_prev:
2757 }

```

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

## 24.3 Notation Components

2758 `<@@=stex_notationcomps>`

**`\stex_highlight_term:nn`**

```

2759
2760 \str_new:N \l__stex_notationcomps_highlight_uri_str
2761 \cs_new_protected:Nn \stex_highlight_term:nn {
2762   \exp_args:Nnx
2763   \use:nn {
2764     \str_set:Nx \l__stex_notationcomps_highlight_uri_str { #1 }
2765     #2
2766   } {
2767     \str_set:Nx \exp_not:N \l__stex_notationcomps_highlight_uri_str
2768     { \l__stex_notationcomps_highlight_uri_str }
2769   }

```

```

2770 }
2771
2772 \cs_new_protected:Nn \stex_unhighlight_term:n {
2773 % \latexml_if:TF {
2774 % #1
2775 % } {
2776 % \rustex_if:TF {
2777 % #1
2778 % } {
2779 #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
2780 % }
2781 % }
2782 }

```

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

```

\comp
\compemph@uri 2783 \cs_new_protected:Npn \comp #1 {
\compemph 2784 \str_if_empty:NF \l__stex_notationcomps_highlight_uri_str {
\defemph 2785 \rustex_if:TF {
\defemph@uri 2786 \stex_annotate:nnn { comp }{ \l__stex_notationcomps_highlight_uri_str }{ #1 }
\symrefemph 2787 }{
\symrefemph@uri 2788 \exp_args:Nnx \compemph@uri { #1 } { \l__stex_notationcomps_highlight_uri_str }
2789 }
2790 }
2791 }
2792
2793 \cs_new_protected:Npn \compemph@uri #1 #2 {
2794 \compemph{ #1 }
2795 }
2796
2797
2798 \cs_new_protected:Npn \compemph #1 {
2799 \textcolor{blue}{#1}
2800 }
2801
2802 \cs_new_protected:Npn \defemph@uri #1 #2 {
2803 \defemph{#1}
2804 }
2805
2806 \cs_new_protected:Npn \defemph #1 {
2807 \textbf{#1}
2808 }
2809
2810 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
2811 \symrefemph{#1}
2812 }
2813
2814 \cs_new_protected:Npn \symrefemph #1 {
2815 \textbf{#1}
2816 }

```

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

**\ellipses**

```
2817 \NewDocumentCommand \ellipses {} { \ldots }
```

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

```

\parray
\prmatrix 2818 \bool_new:N \l_stex_inarray_bool
\parrayline 2819 \bool_set_false:N \l_stex_inarray_bool
\parraylineh 2820 \NewDocumentCommand \parray { m m } {
\parraycell 2821 \begingroup
2822 \bool_set_true:N \l_stex_inarray_bool
2823 \begin{array}{#1}
2824 #2
2825 \end{array}
2826 \endgroup
2827 }
2828
2829 \NewDocumentCommand \prmatrix { m } {
2830 \begingroup
2831 \bool_set_true:N \l_stex_inarray_bool
2832 \begin{matrix}
2833 #1
2834 \end{matrix}
2835 \endgroup
2836 }
2837
2838 \def \maybepline {
2839 \bool_if:NT \l_stex_inarray_bool {\hline}
2840 }
2841
2842 \def \parrayline #1 #2 {
2843 #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
2844 }
2845
2846 \def \pmrow #1 { \parrayline{}{ #1 } }
2847
2848 \def \parraylineh #1 #2 {
2849 #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
2850 }
2851
2852 \def \parraycell #1 {
2853 #1 \bool_if:NT \l_stex_inarray_bool {&}
2854 }

```

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

```
2855 \endpackage
```

## Chapter 25

# STEX -Structural Features Implementation

```
2856 <*package>
2857
2858 %%%%%%%%%%% features.dtx %%%%%%%%%%%
2859
2860 <@@=stex_features>
      Warnings and error messages
2861
```

### 25.1 The feature environment

**structural@feature**

```
2862
2863 \NewDocumentEnvironment{structural@feature}{ m m m }{
2864   \stex_if_in_module:F {
2865     \msg_set:nnn{stex}{error/nomodule}{
2866       Structural~Feature~has~to~occur~in~a~module:\\
2867       Feature~#2~of~type~#1\\
2868       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
2869     }
2870     \msg_error:nn{stex}{error/nomodule}
2871   }
2872
2873   \str_set:Nx \l_stex_module_name_str {
2874     \prop_item:Nn \l_stex_current_module_prop
2875       { name } / #2 - feature
2876   }
2877
2878   \str_set:Nx \l_stex_module_ns_str {
2879     \prop_item:Nn \l_stex_current_module_prop
2880       { ns }
2881   }
2882
```

```

2883
2884 \str_clear:N \l_tmpa_str
2885 \seq_clear:N \l_tmpa_seq
2886 \tl_clear:N \l_tmpa_tl
2887 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
2888   origname = #2,
2889   name     = \l_stex_module_name_str ,
2890   ns       = \l_stex_module_ns_str ,
2891   imports  = \exp_not:o { \l_tmpa_seq } ,
2892   constants = \exp_not:o { \l_tmpa_seq } ,
2893   content  = \exp_not:o { \l_tmpa_tl } ,
2894   file     = \exp_not:o { \g_stex_currentfile_seq } ,
2895   lang     = \l_stex_module_lang_str ,
2896   sig      = \l_tmpa_str ,
2897   meta     = \l_tmpa_str ,
2898   feature  = #1 ,
2899 }
2900
2901 \stex_if_smsmode:TF {
2902   \stex_smsmode_set_codes:
2903 } {
2904   \begin{stex_annotate_env}{ feature:#1 }{}
2905   \stex_annotate_invisible:nnn{header}{}{ #3 }
2906 }
2907 }{
2908   \str_set:Nx \l_tmpa_str {
2909     c_stex_feature_
2910     \prop_item:Nn \l_stex_current_module_prop { ns } ?
2911     \prop_item:Nn \l_stex_current_module_prop { name }
2912     _prop
2913   }
2914   \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
2915   \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
2916   \stex_if_smsmode:TF {
2917     \exp_args:Nx \stex_add_to_sms:n {
2918       \prop_gset_from_keyval:cn {
2919         c_stex_feature_
2920         \prop_item:Nn \l_stex_current_module_prop { ns } ?
2921         \prop_item:Nn \l_stex_current_module_prop { name }
2922         _prop
2923       } {
2924         origname = #2,
2925         name     = \prop_item:cn { \l_tmpa_str } { name } ,
2926         ns       = \prop_item:cn { \l_tmpa_str } { ns } ,
2927         imports  = \prop_item:cn { \l_tmpa_str } { imports } ,
2928         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
2929         content  = \prop_item:cn { \l_tmpa_str } { content } ,
2930         file     = \prop_item:cn { \l_tmpa_str } { file } ,
2931         lang     = \prop_item:cn { \l_tmpa_str } { lang } ,
2932         sig      = \prop_item:cn { \l_tmpa_str } { sig } ,
2933         meta     = \prop_item:cn { \l_tmpa_str } { meta } ,
2934         feature  = \prop_item:cn { \l_tmpa_str } { feature }
2935       }
2936     }

```

```

2937 } {
2938     \end{stex_annotate_env}
2939 }
2940 }
2941

```

## 25.2 Features

structure

```

2942
2943 \prop_new:N \l_stex_all_structures_prop
2944
2945 \keys_define:nn { stex / features / structure } {
2946     name .str_set_x:N = \l__stex_features_structure_name_str ,
2947 }
2948
2949 \cs_new_protected:Nn \__stex_features_structure_args:n {
2950     \str_clear:N \l__stex_features_structure_name_str
2951     \keys_set:nn { stex / features / structure } { #1 }
2952 }
2953
2954 %\stex_new_feature:nnnn { structure } { 0{} m } {
2955 % \__stex_features_structure_args:n { ##1 }
2956 % \str_if_empty:NT \l__stex_features_structure_name_str {
2957 %     \str_set:Nx \l__stex_features_structure_name_str { ##2 }
2958 % }
2959 %} {
2960 %
2961 %}
2962
2963 \NewDocumentEnvironment{mathstructure}{ 0{} m }{
2964     \__stex_features_structure_args:n { #1 }
2965     \str_if_empty:NT \l__stex_features_structure_name_str {
2966         \str_set:Nx \l__stex_features_structure_name_str { #2 }
2967     }
2968     \exp_args:Nnnx
2969     \begin{structural@feature}{ structure }
2970         { \l__stex_features_structure_name_str }{}
2971         \seq_clear:N \l_tmpa_seq
2972         \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
2973     }{
2974         \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
2975         \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
2976         \str_set:Nx \l_tmpa_str {
2977             \prop_item:Nn \l_stex_current_module_prop { ns } ?
2978             \prop_item:Nn \l_stex_current_module_prop { name }
2979         }
2980     }
2981     \seq_map_inline:Nn \l_tmpa_seq {
2982         \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
2983     }
2984     \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
2985     \exp_args:Nnx

```



```

2986 \AddToHookNext { env / mathstructure / after }{
2987 \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}{
2988 \_stex_term_math_oms:nnnn { \l_tmpa_str }{}{0}{}}
2989 }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
2990 \STEXexport {
2991 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2992 {\prop_item:Nn \l_stex_current_module_prop { origname }}
2993 {\l_tmpa_str}
2994 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2995 {#2}{\l_tmpa_str}
2996 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
2997 % \prop_item:Nn \l_stex_current_module_prop { origname },
2998 % \l_tmpa_str
2999 % }
3000 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3001 % #2,\l_tmpa_str
3002 % }
3003 % \tl_set:cx { #2 } {
3004 % \stex_invoke_structure:n { \l_tmpa_str }
3005 }
3006 }
3007
3008 \end{structural@feature}
3009 % \g_stex_last_feature_prop
3010 }

```

\instantiate

```

3011 \seq_new:N \l__stex_features_structure_field_seq
3012 \str_new:N \l__stex_features_structure_field_str
3013 \str_new:N \l__stex_features_structure_def_tl
3014 \prop_new:N \l__stex_features_structure_prop
3015 \NewDocumentCommand \instantiate { m O{} m }{
3016 \stex_smsmode_set_codes:
3017 \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
3018 \prop_set_eq:Nc \l__stex_features_structure_prop {
3019 c_stex_feature_\l_tmpa_str _prop
3020 }
3021 \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
3022 \seq_map_inline:Nn \l__stex_features_structure_field_seq {
3023 \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
3024 \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3025 \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
3026 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
3027 {!} \l_tmpa_tl
3028 \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3029 \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
3030 \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3031 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3032 }{
3033 \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
3034 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3035 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
3036 \l_tmpa_tl
3037 \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {

```

```

3038         \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
3039         \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3040     }{
3041         \tl_clear:N \l_tmpb_tl
3042     }
3043 }
3044 }{
3045     \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
3046     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3047         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
3048         \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
3049         \tl_clear:N \l_tmpa_tl
3050     }{
3051         % TODO throw error
3052     }
3053 }
3054 % \l_tmpa_str: name
3055 % \l_tmpa_tl: definiens
3056 % \l_tmpb_tl: notation
3057 \tl_if_empty:NT \l__stex_features_structure_field_str {
3058     % TODO throw error
3059 }
3060 \str_clear:N \l_tmpb_str
3061
3062 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3063 \seq_map_inline:Nn \l_tmpa_seq {
3064     \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
3065     \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
3066     \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
3067         \seq_map_break:n {
3068             \str_set:Nn \l_tmpb_str { ####1 }
3069         }
3070     }
3071 }
3072 \prop_get:cnN { g_stex_symdecl_ \l_tmpb_str _prop } {args}
3073     \l_tmpb_str
3074
3075 \tl_if_empty:NTF \l_tmpb_tl {
3076     \tl_if_empty:NF \l_tmpa_tl {
3077         \exp_args:Nx \use:n {
3078             \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
3079         }
3080     }
3081 }{
3082     \tl_if_empty:NTF \l_tmpa_tl {
3083         \exp_args:Nx \use:n {
3084             \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\
3085         }
3086     }
3087 }{
3088     \exp_args:Nx \use:n {
3089         \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
3090         \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
3091     }

```

```

3092     }
3093   }
3094   % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
3095   % \prop_item:Nn \l_stex_current_module_prop {name} ?
3096   % #3/\l_stex_features_structure_field_str
3097   % \par
3098   % \expandafter\present\csname
3099   %   g_stex_symdecl_
3100   %   \prop_item:Nn \l_stex_current_module_prop {ns} ?
3101   %   \prop_item:Nn \l_stex_current_module_prop {name} ?
3102   %   #3/\l_stex_features_structure_field_str
3103   %   _prop
3104   % \endcsname
3105 }
3106
3107 \tl_clear:N \l__stex_features_structure_def_tl
3108
3109 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3110 \seq_map_inline:Nn \l_tmpa_seq {
3111   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3112   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3113   \exp_args:Nx \use:n {
3114     \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
3115
3116       }
3117   }
3118
3119   \prop_if_exist:cF {
3120     g_stex_symdecl_
3121     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3122     \prop_item:Nn \l_stex_current_module_prop {name} ?
3123     #3/\l_tmpa_str
3124     _prop
3125   }{
3126     \prop_get:cnN { g_stex_symdecl_ ##1 _prop } {args}
3127     \l_tmpb_str
3128     \exp_args:Nx \use:n {
3129       \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
3130     }
3131   }
3132 }
3133
3134 \symdecl*[type={\STEXsymbol{module-type}}{
3135   \_stex_term_math_oms:nnnn {
3136     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3137     \prop_item:Nn \l__stex_features_structure_prop {name}
3138     }{}{0}{}
3139   }{}{#3}
3140
3141 % TODO: -> sms file
3142
3143 \tl_set:cx{ #3 }{
3144   \stex_invoke_structure:nnn {
3145     \prop_item:Nn \l_stex_current_module_prop {ns} ?

```

```

3146     \prop_item:Nn \l_stex_current_module_prop {name} ? #3
3147   } {
3148     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3149     \prop_item:Nn \l__stex_features_structure_prop {name}
3150   }
3151 }
3152
3153 }

```

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

\stex\_invoke\_structure:nnn

```

3154 % #1: URI of the instance
3155 % #2: URI of the instantiated module
3156 \cs_new_protected:Nn \stex_invoke_structure:nnn {
3157   \tl_if_empty:nTF{ #3 }{
3158     \prop_set_eq:Nc \l__stex_features_structure_prop {
3159       c_stex_feature_ #2 _prop
3160     }
3161     \tl_clear:N \l_tmpa_tl
3162     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3163     \seq_map_inline:Nn \l_tmpa_seq {
3164       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3165       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3166       \cs_if_exist:cT {
3167         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
3168       }{
3169         \tl_if_empty:NF \l_tmpa_tl {
3170           \tl_put_right:Nn \l_tmpa_tl {,}
3171         }
3172         \tl_put_right:Nx \l_tmpa_tl {
3173           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
3174         }
3175       }
3176     }
3177     \exp_args:No \mathstrut \l_tmpa_tl
3178   }{
3179     \stex_invoke_symbol:n{#1/#3}
3180   }
3181 }

```

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

```

3182 </package>

```

## Chapter 26

# STEX -Statements Implementation

```
3183 <*package>
3184
3185 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3186
3187 \protected\def\ignorespacesandpars{
3188   \begingroup\catcode13=10\relax
3189   \@ifnextchar\par{
3190     \endgroup\expandafter\ignorespacesandpars\@gobble
3191   }{
3192     \endgroup
3193   }
3194 }
3195
3196 <@@=stex_statements>
3197
3198   Warnings and error messages
3199
3197 \def\titleemph#1{\textbf{#1}}
3198
symboldoc
3199 \NewDocumentEnvironment{symboldoc}{m}{
3200   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3201   \seq_clear:N \l_tmpb_seq
3202   \seq_map_inline:Nn \l_tmpa_seq {
3203     \str_if_eq:nnF{ ##1 }{}{
3204       \stex_get_symbol:n { ##1 }
3205       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3206         \l_stex_get_symbol_uri_str
3207       }
3208     }
3209   }
3210   \par
3211   \exp_args:Nnnx
3212   \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
3213 }
```

```

3214 \end{stex_annotate_env}
3215 }

3216 \seq_new:N \g_stex_statements_patched_seq
3217
3218 \cs_new_protected:Nn \stex_statements_set_patched:n {
3219   \seq_put_right:Nn \g_stex_statements_patched_seq {#1}
3220 }
3221
3222 \cs_new_protected:Nn \stex_statements_patch:nn {
3223   \seq_if_in:NnF \g_stex_statements_patched_seq {#1} {
3224     \AddToHook{begindocument}{
3225       \cs_if_exist:cTF{end#1}{
3226         \AddToHook{env/#1/before}[stex]{\use:c{__stex_statements_#2_begin:n}{}}
3227         \AddToHook{env/#1/after}[stex]{\use:c{__stex_statements_#2_end:}}
3228       }{
3229         \NewDocumentEnvironment{#1}{0{}}{
3230           \titleemph{\stex_capitalize:n #1}~
3231           \use:c{__stex_statements_#2_begin:n}{ }
3232         }{
3233           \use:c{__stex_statements_#2_end:}
3234         }
3235       }
3236     }
3237   }
3238 }

```

## 26.1 Definitions

definition

```

3239 \keys_define:nn {stex / definiendum }{
3240   post      .tl_set:N      = \l__stex_statements_definiendum_post_tl,
3241   root      .str_set_x:N   = \l__stex_statements_definiendum_root_str,
3242   gfa       .str_set_x:N   = \l__stex_statements_definiendum_gfa_str
3243 }
3244 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
3245   \str_clear:N \l__stex_statements_definiendum_root_str
3246   \tl_clear:N \l__stex_statements_definiendum_post_tl
3247   \str_clear:N \l__stex_statements_definiendum_gfa_str
3248   \keys_set:nn { stex / definiendum }{ #1 }
3249 }
3250 \NewDocumentCommand \definiendum { O{} m m } {
3251   \__stex_statements_definiendum_args:n { #1 }
3252   \stex_get_symbol:n { #2 }
3253   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3254   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
3255     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
3256       \tl_set:Nn \l_tmpa_tl { #3 }
3257     } {
3258       \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
3259       \tl_set:Nn \l_tmpa_tl {
3260         \l__stex_statements_definiendum_root_str\l__stex_statements_definiendum_post_tl
3261       }

```

```

3262     }
3263   } {
3264     \tl_set:Nn \l_tmpa_tl { #3 }
3265   }
3266
3267   % TODO root
3268   \rustex_if:TF {
3269     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
3270   } {
3271     \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
3272   }
3273 }
3274 \stex_deactivate_macro:Nn \definiendum {definition-environments}
3275
3276 \NewDocumentCommand \definame { 0{ } m } {
3277   \__stex_statements_definiendum_args:n { #1 }
3278   % TODO: root
3279   \stex_get_symbol:n { #2 }
3280   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3281   \str_set:Nx \l_tmpa_str {
3282     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3283   }
3284   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {-}
3285   \rustex_if:TF {
3286     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3287       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3288     }
3289   } {
3290     \defemph@uri {
3291       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3292     } { \l_stex_get_symbol_uri_str }
3293   }
3294 }
3295 \stex_deactivate_macro:Nn \definame {definition-environments}
3296
3297 \cs_new_protected:Nn \__stex_statements_defi_begin:n {
3298   \stex_reactivate_macro:N \definiendum
3299   \stex_reactivate_macro:N \definame
3300   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3301   \seq_clear:N \l_tmpb_seq
3302   \seq_map_inline:Nn \l_tmpa_seq {
3303     \str_if_eq:nnF{ ##1 }{}{
3304       \stex_get_symbol:n { ##1 }
3305       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3306         \l_stex_get_symbol_uri_str
3307       }
3308     }
3309   }
3310   \stex_smsmode_set_codes:
3311   \exp_args:Nnnx
3312   \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
3313 }
3314
3315 \cs_new_protected:Nn \__stex_statements_defi_end: {

```

```

3316 \end{stex_annotate_env}
3317 }

Hook:

3318 \stex_statements_patch:nn{definition}{defi}

inline:

3319 \NewDocumentCommand \inlinedef { m } {
3320 \begingroup
3321 \stex_reactivate_macro:N \definiendum
3322 \stex_reactivate_macro:N \definame
3323 \stex_ref_new_doc_target:n{
3324 #1
3325 \endgroup
3326 }

```

## 26.2 Assertions

assertion

```

3327 \cs_new_protected:Nn \__stex_statements_assertion_begin:n {
3328 \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3329 \seq_clear:N \l_tmpb_seq
3330 \seq_map_inline:Nn \l_tmpa_seq {
3331 \str_if_eq:nnF{ ##1 }{}{
3332 \stex_get_symbol:n { ##1 }
3333 \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3334 \l_stex_get_symbol_uri_str
3335 }
3336 }
3337 }
3338 \stex_smsmode_set_codes:
3339 \exp_args:Nnnx
3340 \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
3341 }
3342
3343 \cs_new_protected:Nn \__stex_statements_assertion_end: {
3344 \end{stex_annotate_env}
3345 }

Hook:

3346 \stex_statements_patch:nn{assertion}{assertion}

inline:

3347 \NewDocumentCommand \inlineass { m } {
3348 \begingroup
3349 \stex_ref_new_doc_target:n{
3350 #1
3351 \endgroup
3352 }

```



theorem

```

3353 \cs_new_protected:Nn \__stex_statements_theorem_begin:n {
3354   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3355   \seq_clear:N \l_tmpb_seq
3356   \seq_map_inline:Nn \l_tmpa_seq {
3357     \str_if_eq:nnF{ ##1 }{}{
3358       \stex_get_symbol:n { ##1 }
3359       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3360         \l_stex_get_symbol_uri_str
3361       }
3362     }
3363   }
3364   \stex_smsmode_set_codes:
3365   \exp_args:Nnnx
3366   \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
3367 }
3368
3369 \cs_new_protected:Nn \__stex_statements_theorem_end: {
3370   \end{stex_annotate_env}
3371 }

```

Hook:

```

3372 \stex_statements_patch:nn{theorem}{theorem}

```

lemma

```

3373 \cs_new_protected:Nn \__stex_statements_lemma_begin:n {
3374   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3375   \seq_clear:N \l_tmpb_seq
3376   \seq_map_inline:Nn \l_tmpa_seq {
3377     \str_if_eq:nnF{ ##1 }{}{
3378       \stex_get_symbol:n { ##1 }
3379       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3380         \l_stex_get_symbol_uri_str
3381       }
3382     }
3383   }
3384   \stex_smsmode_set_codes:
3385   \exp_args:Nnnx
3386   \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
3387 }
3388
3389 \cs_new_protected:Nn \__stex_statements_lemma_end: {
3390   \end{stex_annotate_env}
3391 }

```

Hook:

```

3392 \stex_statements_patch:nn{lemma}{lemma}

```

axiom

```

3393 \cs_new_protected:Nn \__stex_statements_axiom_begin:n {
3394   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3395   \seq_clear:N \l_tmpb_seq
3396   \seq_map_inline:Nn \l_tmpa_seq {

```

```

3397 \str_if_eq:nnF{ ##1 }{}{
3398 \stex_get_symbol:n { ##1 }
3399 \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3400 \l_stex_get_symbol_uri_str
3401 }
3402 }
3403 }
3404 \stex_smsmode_set_codes:
3405 \exp_args:Nnnx
3406 \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
3407 }
3408
3409 \cs_new_protected:Nn \__stex_statements_axiom_end: {
3410 \end{stex_annotate_env}
3411 }

Hook:

3412 \stex_statements_patch:nn{axiom}{axiom}

```

## 26.3 Examples

example

```

3413 \cs_new_protected:Nn \__stex_statements_example_begin:n {
3414 \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3415 \seq_clear:N \l_tmpb_seq
3416 \seq_map_inline:Nn \l_tmpa_seq {
3417 \str_if_eq:nnF{ ##1 }{}{
3418 \stex_get_symbol:n { ##1 }
3419 \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3420 \l_stex_get_symbol_uri_str
3421 }
3422 }
3423 }
3424 \stex_smsmode_set_codes:
3425 \exp_args:Nnnx
3426 \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpb_seq {,}}
3427 }
3428
3429 \cs_new_protected:Nn \__stex_statements_example_end: {
3430 \end{stex_annotate_env}
3431 }

Hook:

3432 \stex_statements_patch:nn{example}{example}

inline:

3433 \NewDocumentCommand \inlineex { m } {
3434 \begingroup
3435 \stex_ref_new_doc_target:n{
3436 #1
3437 \endgroup
3438 }

```

## 26.4 OMText

```

3439 \keys_define:nn { stex / omtext} {
3440   id      .str_set_x:N    = \l_stex_omtext_id_str ,
3441   title   .tl_set:N       = \l_stex_omtext_title_tl ,
3442   type    .tl_set_x:N     = \l_stex_omtext_type_tl ,
3443   for     .tl_set_x:N     = \l_stex_omtext_for_tl ,
3444   from    .tl_set_x:N     = \l_stex_omtext_from_tl ,
3445   start   .tl_set:N       = \l_stex_omtext_start_tl ,
3446 }
3447 \cs_new_protected:Nn \stex_omtext_args:n {
3448   \tl_clear:N \l_stex_omtext_title_tl
3449   \tl_clear:N \l_stex_omtext_start_tl
3450   \keys_set:nn { stex / omtext }{ #1 }
3451 }
3452 \newif\if@in@omtext\@in@omtextfalse
3453 \NewDocumentEnvironment {omtext} { 0{} } {
3454   \stex_omtext_args:n { #1 }
3455   \tl_if_empty:NTF \l_stex_omtext_start_tl {
3456     \tl_if_empty:NF \l_stex_omtext_title_tl {
3457       \titleemph{\l_stex_omtext_title_tl}:~
3458     }
3459   }{
3460     \titleemph{\l_stex_omtext_start_tl}~
3461   }
3462   \@in@omtexttrue
3463
3464   \stex_ref_new_doc_target:n \l_stex_omtext_id_str
3465   \stex_smsmode_set_codes:
3466   \ignorespacesandpars
3467 }{}
3468 \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>

```
3469 <*package>
3470 <@@=stex_sproof>
3471
3472 %%%%%%%%%%% sproof.dtx %%%%%%%%%%%
3473
```

### 27.2 Proofs

We first define some keys for the proof environment.

```
3474 \keys_define:nn { stex / spf } {
3475   id          .str_set:N = \l__stex_sproof_spf_id_str,
3476   display     .tl_set:N  = \l__stex_sproof_spf_display_tl,
3477   for         .tl_set:N  = \l__stex_sproof_spf_for_tl ,
3478   from        .tl_set:N  = \l__stex_sproof_spf_from_tl ,
3479   proofend    .tl_set:N  = \l__stex_sproof_spf_proofend_tl,
3480   type        .tl_set:N  = \l__stex_sproof_spf_type_tl,
3481   title       .tl_set:N  = \l__stex_sproof_spf_title_tl,
3482   continues   .tl_set:N  = \l__stex_sproof_spf_continues_tl,
3483   functions   .tl_set:N  = \l__stex_sproof_spf_functions_tl,
3484   method      .tl_set:N  = \l__stex_sproof_spf_method_tl
3485 }
3486 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
3487   \str_clear:N \l__stex_sproof_spf_id_str
3488   \tl_clear:N \l__stex_sproof_spf_display_tl
3489   \tl_clear:N \l__stex_sproof_spf_for_tl
3490   \tl_clear:N \l__stex_sproof_spf_from_tl
3491   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
3492   \tl_clear:N \l__stex_sproof_spf_type_tl
3493   \tl_clear:N \l__stex_sproof_spf_title_tl

```

---

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

```

3494 \tl_clear:N \l__stex_sproof_spf_continues_tl
3495 \tl_clear:N \l__stex_sproof_spf_functions_tl
3496 \tl_clear:N \l__stex_sproof_spf_method_tl
3497 \keys_set:nn { stex / spf }{ #1 }
3498 }

```

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

```

3499 \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.

```

3500 \newcount\count_ten
3501 \newenvironment{pst@with@label}[1]{
3502   \edef\pst@label{#1}
3503   \advance\count_ten by 1\relax
3504   \count_ten=1
3505 }{
3506   \advance\count_ten by -1\relax
3507 }

```

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

```

3508 \def\the@pst@label{
3509   \pst@make@label\pst@label{\number\count_ten}\l__stex_sproof_pstlabel_postfix_tl
3510 }

```

(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.

```

3511 \keys_define:nn { stex / pstlabel }{
3512   prefix      .tl_set:N   = \l__stex_sproof_pstlabel_prefix_tl,
3513   delimiter    .tl_set:N   = \l__stex_sproof_pstlabel_delimiter_tl,
3514   postfix     .tl_set:N   = \l__stex_sproof_pstlabel_postfix_tl
3515 }
3516 \cs_new_protected:Nn \__stex_sproof_pstlabel_args:n {

```

---

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

```

3517 \tl_set:Nn \l__stex_sproof_pstlabel_prefix_tl {P}
3518 \tl_set:Nn \l__stex_sproof_pstlabel_delimiter_tl {.}
3519 \tl_clear:N \l__stex_sproof_pstlabel_postfix_tl
3520 }
3521 \__stex_sproof_pstlabel_args:n {}
3522 \newcommand\setpstlabelstyle[1]{
3523   \__stex_sproof_pstlabel_args:n {#1}
3524 }
3525 \newcommand\setpstlabelstyledefault{%
3526   \__stex_sproof_pstlabel_args:n{prefix=P,delimiter=.,postfix={}}
3527 }

```

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

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

```

3528 \ExplSyntaxOff
3529 \def\pst@make@label@long#1#2{\@for\@I:=#1\do{\expandafter\expandafter\expandafter\@I\csname
3530 \def\pst@make@label@angles#1#2{\ensuremath{\@for\@I:=#1\do{\rangle}}#2}
3531 \def\pst@make@label@short#1#2{#2}
3532 \def\pst@make@label@empty#1#2{}
3533 \ExplSyntaxOn
3534 \def\pstlabelstyle#1{%
3535   \def\pst@make@label{\use:c{pst@make@label@#1}}%
3536 }%
3537 \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.

```

3538 \def\next@pst@label{%
3539   \global\advance\count\count10 by 1%
3540 }%

```

(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

```

3541 \def\sproof@box{
3542   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
3543 }
3544 \def\spf@proofend{\sproof@box}
3545 \def\sproofend{
3546   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
3547     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
3548   }
3549 }
3550 \def\sProofEndSymbol#1{\def\sproof@box{#1}}

```

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

**spf@\*@kw**

```

3551 \def\spf@proofsketch@kw{Proof Sketch}
3552 \def\spf@proof@kw{Proof}
3553 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

3554 \cs_if_exist:NT \bbl@loaded {
3555   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
3556   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
3557     \input{proof-ngerman.lda}
3558   }
3559   \clist_if_in:NnT \l_tmpa_clist {finnish}{
3560     \input{proof-finnish.lda}
3561   }
3562   \clist_if_in:NnT \l_tmpa_clist {french}{
3563     \input{proof-french.lda}
3564   }
3565   \clist_if_in:NnT \l_tmpa_clist {russian}{
3566     \input{proof-russian.lda}
3567   }
3568 }
3569

```

`spfsketch`

```

3570 \newcommand\spfsketch[2] [] {
3571   \__stex_sproof_spf_args:n{#1}
3572   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
3573     \titleemph{
3574       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
3575         \spf@proofsketch@kw
3576       }{
3577         \l__stex_sproof_spf_type_tl
3578       }
3579     }:
3580   }
3581   {-#2}
3582   %\sref@label@id{this \ifx\spf@type\@empty\spf@proofsketch@kw\else\spf@type\fi}
3583   \sproofend
3584 }

```

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

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

```

3585 \newenvironment{spfeq}[2] [] {
3586   \__stex_sproof_spf_args:n{#1}
3587   %\sref@target
3588   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
3589     \titleemph{
3590       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
3591         \spf@proof@kw
3592       }{
3593         \l__stex_sproof_spf_type_tl
3594       }
3595     }:

```

<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

```

3596 }
3597 {~#2}
3598 \begin{displaymath}\begin{array}{rcll}
3599 }{
3600 \end{array}\end{displaymath}
3601 }

```

(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.

```

3602 \newenvironment{spf@proof}[2][]{
3603   \__stex_sproof_spf_args:n{#1}
3604   %\sref@target
3605   \count_ten=10
3606   \par\noindent
3607   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3608     \titleemph{
3609       \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {
3610         \spf@proof@kw
3611       }{
3612         \l__stex_sproof_spf_type_tl
3613       }
3614     }:
3615   }
3616   {~#2}
3617   %\sref@label@id{this \ifx\spf@type\empty\spf@proof@kw\else\spf@type\fi}
3618   \def\pst@label{}
3619   \newcount\pst@count% initialize the labeling mechanism
3620   \begin{description}\begin{pst@with@label}{\l__stex_sproof_pstlabel_prefix_tl}
3621   }{
3622     \end{pst@with@label}\end{description}
3623   }
3624   \newenvironment{sproof}[2][{\begin{spf@proof}[#1]{#2}}{\sproofend\end{spf@proof}}
3625   \newenvironment{sProof}[2][{\begin{spf@proof}[#1]{#2}}{\end{spf@proof}}

```

**\spfidea**

```

3626 \newcommand\spfidea[2][]{
3627   \__stex_sproof_spf_args:n{#1}
3628   \titleemph{
3629     \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {Proof~Idea}{
3630       \l__stex_sproof_spf_type_tl
3631     }:
3632   }~#2
3633   \sproofend
3634 }

```

(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

```

3635 \newenvironment{spfstep}[1][]{
3636   \_stex_sproof_spf_args:n{#1}
3637   \@in@omtexttrue
3638   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3639     \item[\the@pst@label]
3640   }
3641   \tl_if_empty:NF \l__stex_sproof_spf_title_tl {
3642     {(\titleemph{\l__stex_sproof_spf_title_tl})\enspace}
3643   }
3644   %\sref@label@id{\pst@label}
3645   \ignorespacesandpars
3646 }{
3647   \next@pst@label\ignorespacesandpars
3648 }

```

sproofcomment

```

3649 \newenvironment{sproofcomment}[1][]{
3650   \_stex_sproof_spf_args:n{#1}
3651   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3652     \item[\the@pst@label]
3653   }
3654 }{
3655   \next@pst@label
3656 }

```

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.

```

3657 \newenvironment{subproof}[2][]{
3658   \_stex_sproof_spf_args:n{#1}
3659   \def\@test{#2}
3660   \ifx\@test\empty\else
3661     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3662       \item[\the@pst@label]
3663     }{#2}
3664   \fi
3665   \begin{pst@with@label}{\pst@label,\number\count_ten}
3666 }{
3667   \end{pst@with@label}\next@pst@label
3668 }

```

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

```

3669 \newenvironment{spfcases}[2][]{
3670   \def\@test{#1}
3671   \ifx\@test\empty
3672     \begin{subproof}[method=by-cases]{#2}
3673   \else
3674     \begin{subproof}[#1,method=by-cases]{#2}
3675   \fi
3676 }{

```

---

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

```

3677 \end{subproof}
3678 }

```

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

```

3679 \newenvironment{spfcase}[2] [] {
3680   \__stex_sproof_spf_args:n{#1}
3681   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3682     \item[\the@pst@label]
3683   }
3684   \def\@test{#2}
3685   \ifx\@test\@empty
3686   \else
3687     {\titleemph{#2}:~}
3688   \fi
3689   \begin{pst@with@label}{\pst@label,\number\count_ten}
3690   }{
3691     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3692       \sproofend
3693     }
3694     \end{pst@with@label}
3695     \next@pst@label
3696   }

```

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

```

3697 \newcommand\spfcasesketch[3] [] {
3698   \__stex_sproof_spf_args:n{#1}
3699   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3700     \item[\the@pst@label]
3701   }
3702   \def\@test{#2}
3703   \ifx\@test\@empty
3704   \else
3705     {\titleemph{#2}:~}
3706   \fi#3
3707   \next@pst@label
3708 }%

```

## 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.

```

3709 \keys_define:nn { stex / just }{
3710   id          .str_set:x:N = \l__stex_sproof_just_id_str,
3711   method      .tl_set:N   = \l__stex_sproof_just_method_tl,
3712   premises    .tl_set:N   = \l__stex_sproof_just_premises_tl,
3713   args        .tl_set:N   = \l__stex_sproof_just_args_tl
3714 }

```

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**

```
3715 \newenvironment{justification}[1] [] {}{}
```

**\premise**

```
3716 \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.

```
3717 \newcommand\justarg[2] [] {#2}
```

```
3718 \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

```
3719 <*package>
3720
3721 %%%%%%%%%% others.dtx %%%%%%%%%%
3722
3723 <@@=stex_others>
    Warnings and error messages
3724 % None

\MSC Math subject classifier

3725 \NewDocumentCommand \MSC {m} {
3726 % TODO
3727 }

(End definition for \MSC. This function is documented on page 10.)
    Patching tikzinput, if loaded
3728 \@ifpackageloaded{tikzinput}{
3729 \RequirePackage{stex-tikzinput}
3730 }{}
3731 </package>
```

## Chapter 29

# STEX -Metatheory Implementation

```
3732 <*package>
3733 <@@=stex_modules>
3734
3735 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
3736
3737 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
3738 \begingroup
3739 \stex_module_setup:nn{
3740   ns=\c_stex_metatheory_ns_str,
3741   meta=NONE
3742 }{Metatheory}
3743 \stex_reactivate_macro:N \symdecl
3744 \stex_reactivate_macro:N \notation
3745 \stex_reactivate_macro:N \symdef
3746 \ExplSyntaxOff
3747 \csname stex_suppress_html:n\endcsname{
3748   % is-a (a:A, a \in A, a is an A, etc.)
3749   \symdecl[args=ai]{isa}
3750   \notation[typed]{isa}{#1 \comp{:} #2}{#1 \comp, #2}
3751   \notation[in]{isa}{#1 \comp\in #2}{#1 \comp, #2}
3752   \notation[pred]{isa}{#2\comp(#1 \comp)}{#1 \comp, #2}
3753
3754   % bind (\forall, \Pi, \lambda etc.)
3755   \symdecl[args=Bi]{bind}
3756   \notation[forall]{bind}{\comp\forall #1.\;#2}{#1 \comp, #2}
3757   \notation[\Pi]{bind}{\comp\prod_{#1}#2}{#1 \comp, #2}
3758   \notation[deffun]{bind}{\comp( #1 \comp)\; \to\;}{#1 \comp, #2}
3759
3760   % dummy variable
3761   \symdecl{dummyvar}
3762   \notation[underscore]{dummyvar}{\comp\_}
3763   \notation[dot]{dummyvar}{\comp\cdot}
3764   \notation[dash]{dummyvar}{\comp{\rm --}}
3765
3766   %fromto (function space, Hom-set, implication etc.)
```

```

3767 \symdecl[args=ai]{fromto}
3768 \notation[xarrow]{fromto}{#1 \comp\to #2}{#1 \comp\times #2}
3769 \notation[arrow]{fromto}{#1 \comp\to #2}{#1 \comp\to #2}
3770
3771 % mapto (lambda etc.)
3772 %\symdecl[args=Bi]{mapto}
3773 %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
3774 %\notation[lambda]{mapto}{\comp\lambda #1 \comp. \; #2}{#1 \comp, #2}
3775 %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp. \; #2}{#1 \comp, #2}
3776
3777 % function/operator application
3778 \symdecl[args=ia]{apply}
3779 \notation[prec=0;0x\infpres,parens]{apply}{#1 \comp( #2 \comp)}{#1 \comp, #2}
3780 \notation[prec=0;0x\infpres,lambda]{apply}{#1 \; #2 }{#1 \; #2}
3781
3782 % ‘‘type’’ of all collections (sets, classes, types, kinds)
3783 \symdecl{collection}
3784 \notation[U]{collection}{\comp{\mathcal{U}}}
3785 \notation[set]{collection}{\comp{\textsf{Set}}}
3786
3787 % sequences
3788 \symdecl[args=1]{seqtype}
3789 \notation[kleene]{seqtype}{#1^{\comp\ast}}
3790
3791 \symdef[args=2,li,prec=nobrackets]{sequence-index}{#1_{#2}}
3792 \notation[ui,prec=nobrackets]{sequence-index}{#1^{#2}}
3793
3794 %\symdef[args=3,li]{sequence-from-to}{#1_{#2}\comp{\,\ellipses\,}#1_{#3}}
3795 %\notation[ui]{sequence-from-to}{#1^{#2}\comp{\,\ellipses\,}#1^{#3}}
3796 % ^ superceded by \aseqfromto and \livar/\uivar
3797
3798 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{\,\ellipses\,}}{#1\comp,#2}
3799 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{\,\ellipses\,}#2}{#1\comp,#2}
3800 \symdef[args=aui,prec=nobrackets]{aseqfromtovia}{#1\comp{\,\ellipses\,}#2\comp{\,\ellipses\,}#3}{#1\comp,#2}
3801
3802 % letin (‘‘let’’, local definitions, variable substitution)
3803 \symdecl[args=bii]{letin}
3804 \notation[let]{letin}{\comp{\rm let}}{\;#1\comp{=}\;#2\; \comp{\rm in}}{\;#3}
3805 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
3806 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
3807
3808 % structures
3809 \symdecl*[args=1]{module-type}
3810 \notation{module-type}{\mathtt{MOD} #1}
3811 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
3812 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{#1 \comp, #2}
3813
3814 }
3815 \ExplSyntaxOn
3816 \stex_add_to_current_module:n{
3817   \let\nappa\apply
3818   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
3819   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
3820   \def\livar{\csname sequence-index\endcsname[li]}

```

```

3821 \def\uivar{\csname sequence-index\endcsname[ui]}
3822 \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
3823 \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
3824 \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
3825 }
3826 \__stex_modules_end_module:
3827 \endgroup
3828 \endpackage

```

## Chapter 30

# Tikzinput Implementation

```
3829 <*package>
3830
3831 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
3832
3833 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
3834 \RequirePackage{l3keys2e}
3835
3836 \keys_define:nn { tikzinput } {
3837   image .bool_set:N = \c_tikzinput_image_bool,
3838   image .default:n = false ,
3839   unknown .code:n = {}
3840 }
3841
3842 \ProcessKeysOptions { tikzinput }
3843
3844 \bool_if:NTF \c_tikzinput_image_bool {
3845   \RequirePackage{graphicx}
3846
3847   \providecommand\usetikzlibrary[]{}
3848   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
3849 }{
3850   \RequirePackage{tikz}
3851   \RequirePackage{standalone}
3852
3853   \newcommand \tikzinput [2] [] {
3854     \setkeys{Gin}{#1}
3855     \ifx \Gin@ewidth \Gin@exclamation
3856       \ifx \Gin@eheight \Gin@exclamation
3857         \input { #2 }
3858       \else
3859         \resizebox{!}{ \Gin@eheight }{
3860           \input { #2 }
3861         }
3862       \fi
3863     \else
3864       \ifx \Gin@eheight \Gin@exclamation
3865         \resizebox{ \Gin@ewidth }{!}{
3866           \input { #2 }
```



```

3867     }
3868     \else
3869         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
3870             \input { #2 }
3871         }
3872     \fi
3873 \fi
3874 }
3875 }
3876
3877 \newcommand \ctikzinput [2] [] {
3878     \begin{center}
3879         \tikzinput [ #1 ] { #2 }
3880     \end{center}
3881 }
3882
3883 \@ifpackageloaded{stex}{
3884     \RequirePackage{stex-tikzinput}
3885 }{}
3886
3887 </package>
3888 <*stex>
3889 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
3890 \RequirePackage{stex}
3891 \RequirePackage{tikzinput}
3892
3893 \newcommand\mhtikzinput [2] [] {%
3894     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
3895     \stex_in_repository:nn\Gin@mhrepos{
3896         \tikzinput [ #1 ] {\mhp{##1}{#2}}
3897     }
3898 }
3899 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [ #1 ] { #2 }\end{center}}
3900 </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.

```
3901 \*cls)
3902 \<@@=document_structure>
3903 \ProvidesExplClass{omdoc}{2020/10/19}{1.4}{OMDoc Documents}
3904 \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`

```
3905 \keys_define:nn{ document-structure / pkg }{
3906   class      .str_set_x:N = \c_document_structure_class_str,
3907   minimal    .bool_set:N = \c_document_structure_minimal_bool,
3908   report     .code:n      = {
3909     \ClassWarning{omdoc}{the option 'report' is deprecated, use 'class=report', instead}
3910     \str_set:Nn \c_document_structure_class_str {report}
3911   },
3912   book       .code:n      = {
3913     \ClassWarning{omdoc}{the option 'book' is deprecated, use 'class=book', instead}
3914     \str_set:Nn \c_document_structure_class_str {book}
3915   },
3916   bookpart   .code:n      = {
3917     \ClassWarning{omdoc}{the option 'bookpart' is deprecated, use 'class=book,topsect=chapter}
3918     \str_set:Nn \c_document_structure_class_str {book}
3919     \str_set:Nn \c_document_structure_topsect_str {chapter}
3920   },
```

```

3921 docopt      .str_set_x:N = \c_document_structure_docopt_str,
3922 unknown     .code:n      = {
3923   \PassOptionsToPackage{ \CurrentOption }{ omdoc }
3924 }
3925 }
3926 \ProcessKeysOptions{ document-structure / pkg }
3927 \str_if_empty:NT \c_document_structure_class_str {
3928   \str_set:Nn \c_document_structure_class_str {article}
3929 }
3930 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
3931   {\c_document_structure_class_str}
3932

```

### 31.3 Beefing up the document environment

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

```

3933 \RequirePackage{omdoc}
3934 \bool_if:NF \c_document_structure_minimal_bool {
3935   \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>

```

3936 \keys_define:nn { document-structure / document }{
3937   id .str_set_x:N = \c_document_structure_document_id_str
3938 }
3939 \let\__document_structure_orig_document=\document
3940 \renewcommand{\document}[1][]{
3941   \keys_set:nn{ document-structure / document }{ #1 }
3942   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
3943   \__document_structure_orig_document
3944 }

```

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

```

3945 }
3946 \</cls>

```

### 31.4 Implementation: OMDoc Package

```

3947 \*package>
3948 \ProvidesExplPackage{omdoc}{2020/10/19}{1.4}{OMDoc document Structure}
3949 \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

```

3950
3951 \keys_define:nn{ document-structure / pkg }{
3952   class      .str_set_x:N = \c_document_structure_class_str,
3953   topsect    .str_set_x:N = \c_document_structure_topsect_str,
3954   % showignores .bool_set:N = \c_document_structure_showignores_bool,
3955 }
3956 \ProcessKeysOptions{ document-structure / pkg }
3957 \str_if_empty:NT \c_document_structure_class_str {
3958   \str_set:Nn \c_document_structure_class_str {article}
3959 }
3960 \str_if_empty:NT \c_document_structure_topsect_str {
3961   \str_set:Nn \c_document_structure_topsect_str {section}
3962 }

```

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

```

3963 \RequirePackage{xspace}
3964 \RequirePackage{comment}
3965 \@ifpackageloaded{babel}{\RequirePackage[base]{babel}}

```

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

```

3966 \@ifpackageloaded{babel}{
3967   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
3968   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
3969     \input{omdoc-ngerman.ldf}
3970   }
3971 }{}
3972 %\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.

```

3973 \int_new:N \l_document_structure_section_level_int
3974 \str_case:VnF \c_document_structure_topsect_str {
3975   {part}{
3976     \int_set:Nn \l_document_structure_section_level_int {0}
3977   }
3978   {chapter}{
3979     \int_set:Nn \l_document_structure_section_level_int {1}
3980   }
3981 }{
3982   \str_case:VnF \c_document_structure_class_str {
3983     {book}{
3984       \int_set:Nn \l_document_structure_section_level_int {0}
3985     }
3986     {report}{
3987       \int_set:Nn \l_document_structure_section_level_int {0}
3988     }
3989   }{
3990     \int_set:Nn \l_document_structure_section_level_int {2}
3991   }
3992 }

```

## 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

```
3993 \def\current@section@level{document}%
3994 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
3995 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
3996 \cs_new_protected:Npn \skipomgroup {
3997   \ifcase\l_document_structure_section_level_int
3998   \or\stepcounter{part}
3999   \or\stepcounter{chapter}
4000   \or\stepcounter{section}
4001   \or\stepcounter{subsection}
4002   \or\stepcounter{subsubsection}
4003   \or\stepcounter{paragraph}
4004   \or\stepcounter{subparagraph}
4005   \fi
4006 }
```

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

`blindomgroup`

```
4007 \newcommand\at@begin@blindomgroup[1]{%
4008 \newenvironment{blindomgroup}
4009 {
4010   \int_incr:N\l_document_structure_section_level_int
4011   \at@begin@blindomgroup\l_document_structure_section_level_int
4012 }{}}
```

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

```
4013 \newcommand\omgroup@nonum[2]{%
4014   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
4015   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
4016 }
```

*(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.

```
4017 \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.

```

4018 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
4019   \@nameuse{#1}{#2}
4020 }{
4021   \cs_if_exist:NTF\rdfmata@sectioning{
4022     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
4023   }{
4024     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
4025   }
4026 }
4027 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
4028 }

```

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

omgroup

```

4029 \keys_define:nn { document-structure / omgroupp }{
4030   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
4031   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
4032   creators     .clist_set:N = \l__document_structure_omgroup_creators_clist,
4033   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
4034   srccite      .tl_set:N   = \l__document_structure_omgroup_srccite_tl,
4035   type         .tl_set:N   = \l__document_structure_omgroup_type_tl,
4036   short        .tl_set:N   = \l__document_structure_omgroup_short_tl,
4037   display      .tl_set:N   = \l__document_structure_omgroup_display_tl,
4038   intro        .tl_set:N   = \l__document_structure_omgroup_intro_tl,
4039   loadmodules  .bool_set:N = \l__document_structure_omgroup_loadmodules_bool
4040 }
4041 \cs_new_protected:Nn \l__document_structure_omgroup_args:n {
4042   \str_clear:N \l__document_structure_omgroup_id_str
4043   \str_clear:N \l__document_structure_omgroup_date_str
4044   \clist_clear:N \l__document_structure_omgroup_creators_clist
4045   \clist_clear:N \l__document_structure_omgroup_contributors_clist
4046   \tl_clear:N \l__document_structure_omgroup_srccite_tl
4047   \tl_clear:N \l__document_structure_omgroup_type_tl
4048   \tl_clear:N \l__document_structure_omgroup_short_tl
4049   \tl_clear:N \l__document_structure_omgroup_display_tl
4050   \tl_clear:N \l__document_structure_omgroup_intro_tl
4051   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
4052   \keys_set:nn { document-structure / omgroupp } { #1 }
4053 }

```

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 omgroupp, i.e. after the section heading.

```

4054 \newif\if@mainmatter\@mainmattertrue
4055 \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.

```

4056 \keys_define:nn { document-structure / sectioning }{
4057   name .str_set_x:N = \l__document_structure_sect_name_str ,
4058   ref  .str_set_x:N = \l__document_structure_sect_ref_str  ,
4059   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
4060   num   .bool_set:N = \l__document_structure_sect_num_bool  ,
4061 }

```

```

4062 \cs_new_protected:Nn \__document_structure_sect_args:n {
4063   \str_clear:N \l__document_structure_sect_name_str
4064   \str_clear:N \l__document_structure_sect_ref_str
4065   \bool_set_false:N \l__document_structure_sect_clear_bool
4066   \bool_set_false:N \l__document_structure_sect_num_bool
4067   \keys_set:nn { document-structure / sectioning } { #1 }
4068 }
4069 \newcommand\omdoc@sectioning[3][]{
4070   \__document_structure_sect_args:n {#1}
4071   \let\omdoc@sect@name\l__document_structure_sect_name_str
4072   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
4073   \if@mainmatter% numbering not overridden by frontmatter, etc.
4074     \bool_if:NTF \l__document_structure_sect_num_bool {
4075       \omgroup@num{#2}{#3}
4076     }{
4077       \omgroup@nonum{#2}{#3}
4078     }
4079     \def\current@section@level{\omdoc@sect@name}
4080   \else
4081     \omgroup@nonum{#2}{#3}
4082   \fi
4083 }% 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.

```

4084 \newcommand\omgroup@redefine@addtocontents[1]{%
4085   %\edef\__document_structureimport{#1}%
4086   %\@for\@I:=\__document_structureimport\do{%
4087     %\edef\@path{\csname module@\@I @path\endcsname}%
4088     %\@ifundefined{tf@toc}\relax%
4089     % {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
4090   %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
4091   %\def\addcontentsline##1##2##3{%
4092     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
4093   %\else% hyperref.sty not loaded
4094   %\def\addcontentsline##1##2##3{%
4095     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
4096   %\fi
4097 }% 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.

```

4098 \int_new:N \l_document_structure_omgroup_level_int
4099 \newenvironment{omgroup}[2][]{% keys, title
4100 {
4101   \__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.

```

4102 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
4103   \omgroup@redefine@addtocontents{
4104     %\@ifundefined{module@id}\used@modules%
4105     %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}

```

```

4106     }
4107 }

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

4108 \int_incr:N \l_document_structure_omgroup_level_int
4109 \int_incr:N \l_document_structure_section_level_int
4110 \ifcase\l_document_structure_section_level_int
4111   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
4112   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
4113   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
4114   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
4115   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
4116   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
4117   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
4118 \fi
4119 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
4120 \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str
4121 }% for customization
4122 {}

```

and finally, we localize the sections

```

4123 \newcommand\omdoc@part@kw{Part}
4124 \newcommand\omdoc@chapter@kw{Chapter}
4125 \newcommand\omdoc@section@kw{Section}
4126 \newcommand\omdoc@subsection@kw{Subsection}
4127 \newcommand\omdoc@subsubsection@kw{Subsubsection}
4128 \newcommand\omdoc@paragraph@kw{paragraph}
4129 \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`

```

4130 \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).

```

4131 \cs_if_exist:NTF\frontmatter{
4132   \let\__document_structure_orig_frontmatter\frontmatter
4133   \let\frontmatter\relax
4134 }{
4135   \tl_set:Nn\__document_structure_orig_frontmatter{
4136     \clearpage
4137     \@mainmatterfalse
4138     \pagenumbering{roman}
4139   }
4140 }
4141 \cs_if_exist:NTF\backmatter{

```



```

4142 \let\__document_structure_orig_backmatter\backmatter
4143 \let\backmatter\relax
4144 }{
4145 \tl_set:Nn\__document_structure_orig_backmatter{
4146 \clearpage
4147 \@mainmatterfalse
4148 \pagenumbering{roman}
4149 }
4150 }

```

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.

```

4151 \newenvironment{frontmatter}{
4152 \__document_structure_orig_frontmatter
4153 }{
4154 \cs_if_exist:NTF\mainmatter{
4155 \mainmatter
4156 }{
4157 \clearpage
4158 \@mainmattertrue
4159 \pagenumbering{arabic}
4160 }
4161 }

```

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

```

4162 \newenvironment{backmatter}{
4163 \__document_structure_orig_backmatter
4164 }{
4165 \cs_if_exist:NTF\mainmatter{
4166 \mainmatter
4167 }{
4168 \clearpage
4169 \@mainmattertrue
4170 \pagenumbering{arabic}
4171 }
4172 }

```

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

```

4173 \@mainmattertrue\pagenumbering{arabic}

```

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

```

4174 \newcommand\afterprematurestop{}
4175 \def\prematurestop@endomgroup{
4176 \int_compare:nNf \l_document_structure_omgroup_level_int = 0 {
4177 \end{omgroup}
4178 \int_decr:N \l_document_structure_omgroup_level_int
4179 \prematurestop@endomgroup
4180 }
4181 }
4182 \providecommand\prematurestop{

```

```

4183 \message{Stopping sTeX processing prematurely}
4184 \prematurestop@endomgroup
4185 \afterprematurestop
4186 \end{document}
4187 }

```

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

## 31.8 Global Variables

**\setSGvar** set a global variable

```

4188 \RequirePackage{etoolbox}
4189 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

**\useSGvar** use a global variable

```

4190 \newrobustcmd\useSGvar[1]{%
4191   \@ifundefined{sTeX@Gvar@#1}
4192   {\PackageError{omdoc}
4193    {The sTeX Global variable #1 is undefined}
4194    {set it with \protect\setSGvar}}
4195   \@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.

```

4196 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
4197   \@ifundefined{sTeX@Gvar@#1}
4198   {\PackageError{omdoc}
4199    {The sTeX Global variable #1 is undefined}
4200    {set it with \protect\setSGvar}}
4201   {\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.

```
4202 \*cls)
4203 \@@=mikoslides)
4204 \ProvidesExplClass{mikoslides}{2020/12/06}{1.3}{MiKo slides Class}
4205 \RequirePackage{l3keys2e,expl-keystr-compatible}
4206
4207 \keys_define:nn{mikoslides / cls}{
4208   class .code:n = {
4209     \PassOptionsToClass{\CurrentOption}{omdoc}
4210     \str_if_eq:nnT{#1}{book}{
4211       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
4212     }
4213     \str_if_eq:nnT{#1}{report}{
4214       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
4215     }
4216   },
4217   notes .bool_set:N = \c__mikoslides_notes_bool ,
4218   slides .code:n = { \bool_set_false:N \c__mikoslides_notes_bool },
4219   unknown .code:n = {
4220     \PassOptionsToClass{\CurrentOption}{omdoc}
4221     \PassOptionsToClass{\CurrentOption}{beamer}
4222     \PassOptionsToPackage{\CurrentOption}{mikoslides}
4223   }
4224 }
4225 \ProcessKeysOptions{ mikoslides / cls }
4226 \bool_if:NTF \c__mikoslides_notes_bool {
4227   \PassOptionsToPackage{notes=true}{mikoslides}
4228 }{
4229   \PassOptionsToPackage{notes=false}{mikoslides}
4230 }
4231 \</cls)
```

now we do the same for the mikoslides package.

```

4232 <*package>
4233 \ProvidesExplPackage{mikoslides}{2020/12/06}{1.3}{MiKo slides Package}
4234 \RequirePackage{l3keys2e,expl-keystr-compat}
4235
4236 \keys_define:nn{mikoslides / pkg}{
4237   topsect      .str_set_x:N = \c__mikoslides_topsect_str,
4238   defaulttopsect .str_set_x:N = \c__mikoslides_defaulttopsec_str,
4239   notes        .bool_set:N = \c__mikoslides_notes_bool ,
4240   slides       .code:n      = { \bool_set_false:N \c__mikoslides_notes_bool },
4241   sectocframes .bool_set:N = \c__mikoslides_sectocframes_bool ,
4242   frameimages  .bool_set:N = \c__mikoslides_frameimages_bool ,
4243   fiboxed      .bool_set:N = \c__mikoslides_fiboxed_bool ,
4244   nopproblems  .bool_set:N = \c__mikoslides_nopproblems_bool,
4245   unknown      .code:n      = {
4246     \PassOptionsToClass{\CurrentOption}{stex}
4247     \PassOptionsToClass{\CurrentOption}{tikzinput}
4248   }
4249 }
4250 \ProcessKeysOptions{ mikoslides / pkg }
4251 \newif\ifnotes
4252 \bool_if:NTF \c__mikoslides_notes_bool {
4253   \notesttrue
4254 }{
4255   \notesfalse
4256 }
4257

```

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

```

4258 \str_if_empty:NTF \c__mikoslides_topsect_str {
4259   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_defaulttopsec_str
4260 }{
4261   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_topsect_str
4262 }
4263 </package>

```

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

```

4264 <*cls>
4265 \bool_if:NTF \c__mikoslides_notes_bool {
4266   \LoadClass{omdoc}
4267 }{
4268   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
4269   \newcounter{Item}
4270   \newcounter{paragraph}
4271   \newcounter{subparagraph}
4272   \newcounter{Hfootnote}
4273   \RequirePackage{omdoc}
4274 }

```

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

```

4275 \RequirePackage{mikoslides}
4276 </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.

```

4277 \*package>
4278 \bool_if:NT \c__mikoslides_notes_bool {
4279   \RequirePackage{a4wide}
4280   \RequirePackage{marginnote}
4281   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
4282   \RequirePackage{mdframed}
4283   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
4284   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
4285 }
4286 \RequirePackage{stex-compatibility}
4287 \RequirePackage{stex-tikzinput}
4288 \RequirePackage{etoolbox}
4289 \RequirePackage{amssymb}
4290 \RequirePackage{amsmath}
4291 \RequirePackage{comment}
4292 \RequirePackage{textcomp}
4293 \RequirePackage{url}
4294 \RequirePackage{graphicx}
4295 \RequirePackage{pgf}

```

## 32.2 Notes and Slides

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

```

4296 \bool_if:NT \c__mikoslides_notes_bool {
4297   \renewcommand\usetheme[2] [] {\usepackage[#1]{beamernotestheme#2}}
4298 }

```

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

```

4299 \newcounter{slide}
4300 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
4301 \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.

```

4302 \bool_if:NTF \c__mikoslides_notes_bool {
4303   \renewenvironment{note}{\ignorespaces}{\}
4304 }{
4305   \excludecomment{note}
4306 }

```

---

<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.

```
4307 \bool_if:NT \c__mikoslides_notes_bool {
4308   \newlength{\slideframewidth}
4309   \setlength{\slideframewidth}{1.5pt}
```

**frame** We first define the keys.

```
4310 \cs_new_protected:Nn \__mikoslides_do_yes_param:Nn {
4311   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
4312     \bool_set_true:N #1
4313   }{
4314     \bool_set_false:N #1
4315   }
4316 }
4317 \keys_define:nn{mikoslides / frame}{
4318   label .str_set_x:N = \l__mikoslides_frame_label_str,
4319   allowframebreaks .code:n = {
4320     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowframebreaks_bool { #1 }
4321   },
4322   allowdisplaybreaks .code:n = {
4323     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowdisplaybreaks_bool { #1 }
4324   },
4325   fragile .code:n = {
4326     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_fragile_bool { #1 }
4327   },
4328   shrink .code:n = {
4329     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_shrink_bool { #1 }
4330   },
4331   squeeze .code:n = {
4332     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_squeeze_bool { #1 }
4333   },
4334   t .code:n = {
4335     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_t_bool { #1 }
4336   },
4337 }
4338 \cs_new_protected:Nn \__mikoslides_frame_args:n {
4339   \str_clear:N \l__mikoslides_frame_label_str
4340   \bool_set_true:N \l__mikoslides_frame_allowframebreaks_bool
4341   \bool_set_true:N \l__mikoslides_frame_allowdisplaybreaks_bool
4342   \bool_set_true:N \l__mikoslides_frame_fragile_bool
4343   \bool_set_true:N \l__mikoslides_frame_shrink_bool
4344   \bool_set_true:N \l__mikoslides_frame_squeeze_bool
4345   \bool_set_true:N \l__mikoslides_frame_t_bool
4346   \keys_set:nn { mikoslides / frame }{ #1 }
4347 }
```

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

```
4348 \renewenvironment{frame}[1][]{
4349   \__mikoslides_frame_args:n{#1}
4350   \sffamily
4351   \stepcounter{slide}
4352   \def\@currentlabel{\theslide}
4353   \str_if_empty:NF \l__mikoslides_frame_label_str {
4354     \label{\l__mikoslides_frame_label_str}
```

4355 }

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

```

4356 \def\itemize@level{outer}
4357 \def\itemize@outer{outer}
4358 \def\itemize@inner{inner}
4359 \renewcommand\newpage{\addtocounter{framenum}{1}}
4360 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
4361 \renewenvironment{itemize}{
4362   \ifx\itemize@level\itemize@outer
4363     \def\itemize@label{\rhd$}
4364   \fi
4365   \ifx\itemize@level\itemize@inner
4366     \def\itemize@label{$\scriptstyle\rhd$}
4367   \fi
4368   \begin{list}
4369   {\itemize@label}
4370   {\setlength{\labelsep}{.3em}
4371    \setlength{\labelwidth}{.5em}
4372    \setlength{\leftmargin}{1.5em}
4373   }
4374   \edef\itemize@level{\itemize@inner}
4375 }{
4376   \end{list}
4377 }

```

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

```

4378 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth]
4379 }{
4380   \medskip\miko@slidelabel\end{mdframed}
4381 }

```

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

`\frametitle`

```

4382 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
4383 }

```

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

EdN:18

`\pause` 18

```

4384 \bool_if:NT \c__mikoslides_notes_bool {
4385   \newcommand\pause{}
4386 }

```

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

`nomtext`

```

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

```

---

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

nomgroup

```
4392 \bool_if:NTF \c__mikoslides_notes_bool {  
4393   \newenvironment{nomgroup}[2] [] {\begin{omgroup}[#1]{#2}}{\end{omgroup}}  
4394 }{  
4395   \excludecomment{nomgroup}  
4396 }
```

ndefinition

```
4397 \bool_if:NTF \c__mikoslides_notes_bool {  
4398   \newenvironment{ndefinition}[1] [] {\begin{definition}[#1]}{\end{definition}}  
4399 }{  
4400   \excludecomment{ndefinition}  
4401 }
```

nassertion

```
4402 \bool_if:NTF \c__mikoslides_notes_bool {  
4403   \newenvironment{nassertion}[1] [] {\begin{assertion}[#1]}{\end{assertion}}  
4404 }{  
4405   \excludecomment{nassertion}  
4406 }
```

nsproof

```
4407 \bool_if:NTF \c__mikoslides_notes_bool {  
4408   \newenvironment{nsproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}  
4409 }{  
4410   \excludecomment{nsproof}  
4411 }
```

nexample

```
4412 \bool_if:NTF \c__mikoslides_notes_bool {  
4413   \newenvironment{nexample}[1] [] {\begin{example}[#1]}{\end{example}}  
4414 }{  
4415   \excludecomment{nexample}  
4416 }
```

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

```
4417 \def\inputref@preskip{\smallskip}  
4418 \def\inputref@postskip{\medskip}
```

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

\inputref\*

```
4419 \let\orig@inputref\inputref  
4420 \def\inputref{\@ifstar\ninputref\orig@inputref}  
4421 \newcommand\ninputref[2] [] {  
4422   \bool_if:NT \c__mikoslides_notes_bool {  
4423     \orig@inputref[#1]{#2}  
4424   }  
4425 }
```

(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>}`.

```

4426 \newlength{\slidelogoheight}
4427
4428 \bool_if:NTF \c__mikoslides_notes_bool {
4429   \setlength{\slidelogoheight}{.4cm}
4430 }{
4431   \setlength{\slidelogoheight}{1cm}
4432 }
4433 \newsavebox{\slidelogo}
4434 \sbox{\slidelogo}{\text{\TeX}}
4435 \newrobustcmd{\setslidelogo}[1]{
4436   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
4437 }
```

(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.

```

4438 \def\source{Michael Kohlhase}% customize locally
4439 \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.

```

4440 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
4441 \newsavebox{\cclogo}
4442 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
4443 \newif\ifcchref\cchreffalse
4444 \AtBeginDocument{
4445   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
4446 }
4447 \def\licensing{
4448   \ifcchref
4449     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
4450   \else
4451     {\usebox{\cclogo}}
4452   \fi
4453 }
4454 \newrobustcmd{\setlicensing}[2][]{
4455   \def@url{#1}
4456   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
4457   \ifx@url@empty
4458     \def\licensing{{\usebox{\cclogo}}}
4459   \else
4460     \def\licensing{
```

```

4461     \ifcchref
4462     \href{#1}{\usebox{\cclogo}}
4463   \else
4464     {\usebox{\cclogo}}
4465   \fi
4466 }
4467 \fi
4468 }

```

(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>

```

4469 \newrobustcmd\miko@slidelabel{
4470   \vbox to \slidelogoheight{
4471     \vss\hbox to \slidewidth
4472     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
4473   }
4474 }

```

(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.

```

4475 \def\Gin@mhrepos{}
4476 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
4477 \define@key{Gin}{label}{\def\currentlabel{\arabic{slide}}\label{#1}}
4478 \newrobustcmd\frameimage[2][]{
4479   \stepcounter{slide}
4480   \bool_if:NT \c__mikoslides_frameimages_bool {
4481     \def\Gin@ewidth{}\setkeys{Gin}{#1}
4482     \bool_if:NF \c__mikoslides_notes_bool { \vfill }
4483     \begin{center}
4484       \bool_if:NTF \c__mikoslides_fiboxed_bool {
4485         \fbox{
4486           \ifx\Gin@ewidth\@empty
4487             \ifx\Gin@mhrepos\@empty
4488               \mhgraphics[width=\slidewidth,#1]{#2}
4489             \else
4490               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
4491             \fi
4492           \else% Gin@ewidth empty
4493             \ifx\Gin@mhrepos\@empty
4494               \mhgraphics[#1]{#2}
4495             \else
4496               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
4497             \fi
4498           \fi% Gin@ewidth empty
4499         }
4500       }{
4501         \ifx\Gin@ewidth\@empty

```

---

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

```

4502         \ifx\Gin@mhrepos\empty
4503             \mhgraphics[width=\slidewidth,#1]{#2}
4504         \else
4505             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
4506         \fi
4507         \ifx\Gin@mhrepos\empty
4508             \mhgraphics[#1]{#2}
4509         \else
4510             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
4511         \fi
4512     \fi% Gin@ewidth empty
4513 }
4514 \end{center}
4515 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
4516 \bool_if:NF \c__mikoslides_notes_bool { \vfill }
4517 }
4518 } % 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.

```

4519 \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

```

4520 \AddToHook{begindocument}{
4521     \definecolor{green}{rgb}{0,.5,0}
4522     \definecolor{purple}{cmyk}{.3,1,0,.17}
4523 }

```

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`.

```

4524 % \def\STpresent#1{\textcolor{blue}{#1}}
4525 \def\defemph#1{\textcolor{magenta}{#1}}
4526 \def\symrefemph#1{\textcolor{cyan}{#1}}
4527 \def\compemph#1{\textcolor{blue}{#1}}
4528 \def\titleemph#1{\textcolor{blue}{#1}}
4529 \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.

```

4530 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
4531 \def\smalltextwarning{
4532     \pgfuseimage{miko@small@dbend}
4533     \xspace
4534 }
4535 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}

```

```

4536 \newrobustcmd\textwarning{
4537   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
4538   \xspace
4539 }
4540 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}
4541 \newrobustcmd\bigtextwarning{
4542   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
4543   \xspace
4544 }

```

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

```

4545 \newrobustcmd\putgraphicsat[3]{
4546   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
4547 }
4548 \newrobustcmd\putat[2]{
4549   \begin{picture}(0,0)\put(#1){#2}\end{picture}
4550 }

```

## 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`.

```

4551 \bool_if:NT \c__mikoslides_sectocframes_bool {
4552   \str_if_eq:VnTF \__mikoslidestopsect{part}{
4553     \newcounter{chapter}\counterwithin*{section}{chapter}
4554   }{
4555     \str_if_eq:VnT\__mikoslidestopsect{chapter}{
4556       \newcounter{chapter}\counterwithin*{section}{chapter}
4557     }
4558   }
4559 }

```

`\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
4560 \def\part@prefix{}
4561 \@ifpackageloaded{omdoc}{}{
4562   \str_case:VnF \__mikoslidestopsect {
4563     {part}{
4564       \int_set:Nn \l_document_structure_section_level_int {0}
4565       \def\thesection{\arabic{chapter}.\arabic{section}}
4566       \def\part@prefix{\arabic{chapter}.}
4567     }
4568     {chapter}{
4569       \int_set:Nn \l_document_structure_section_level_int {1}
4570       \def\thesection{\arabic{chapter}.\arabic{section}}
4571       \def\part@prefix{\arabic{chapter}.}
4572     }
4573   }{
4574     \int_set:Nn \l_document_structure_section_level_int {2}
4575     \def\part@prefix{}

```

```

4576 }
4577 }
4578
4579 \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

```

4580 \renewenvironment{omgroup}[2][]{
4581   \__document_structure_omgroup_args:n { #1 }
4582   \int_incr:N \l_document_structure_omgroup_level_int
4583   \int_incr:N \l_document_structure_section_level_int
4584   \bool_if:NT \c__mikoslides_sectocframes_bool {
4585     \stepcounter{slide}
4586     \begin{frame}[noframenumbering]
4587     \vfill\Large\centering
4588     \red{
4589       \ifcase\l_document_structure_section_level_int\or
4590         \stepcounter{part}
4591         \def\__mikoslideslabel{\omdoc@part@kw~\Roman{part}}
4592         \def\currentsectionlevel{\omdoc@part@kw}
4593       \or
4594         \stepcounter{chapter}
4595         \def\__mikoslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
4596         \def\currentsectionlevel{\omdoc@chapter@kw}
4597       \or
4598         \stepcounter{section}
4599         \def\__mikoslideslabel{\part@prefix\arabic{section}}
4600         \def\currentsectionlevel{\omdoc@section@kw}
4601       \or
4602         \stepcounter{subsection}
4603         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
4604         \def\currentsectionlevel{\omdoc@subsection@kw}
4605       \or
4606         \stepcounter{subsubsection}
4607         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
4608         \def\currentsectionlevel{\omdoc@subsubsection@kw}
4609       \or
4610         \stepcounter{mparagraph}
4611         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{msubsection}.\arabic{mparagraph}}
4612         \def\currentsectionlevel{\omdoc@paragraph@kw}
4613       \fi% end ifcase
4614       \__mikoslideslabel%\sref@label@id\__mikoslideslabel
4615       \quad #2%
4616     }%
4617     \vfill%
4618     \end{frame}%
4619   }
4620   \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str%
4621 }{}
4622 }

```

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

```

4623 \def\inserttheorembodyfont{\normalfont}
4624 \bool_if:NF \c__mikoslices_notes_bool {
4625   \defbeamertemplate{theorem begin}{miko}
4626   {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
4627     \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
4628     \inserttheorempunctuation\inserttheorembodyfont\space}
4629   \defbeamertemplate{theorem end}{miko}{}
```

and we set it as the default one.

```

4630   \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.

```

4631   \expandafter\def\csname Parent2\endcsname{}
4632 }
4633 \bool_if:NT \c__mikoslices_notes_bool {
4634   \renewenvironment{columns}[1][]{%
4635     \par\noindent%
4636     \begin{minipage}%
4637       \slidewidth\centering\leavevmode%
4638   }{%
4639     \end{minipage}\par\noindent%
4640   }%
4641   \newsavebox\columnbox%
4642   \renewenvironment<>{column}[2][]{%
4643     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
4644   }{%
4645     \end{minipage}\end{lrbox}\usebox\columnbox%
4646   }%
4647 }
4648 \bool_if:NTF \c__mikoslices_noproblems_bool {
4649   \newenvironment{problems}{}{}
4650 }{
4651   \excludecomment{problems}
4652 }
```

## 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.

```

4653 \gdef\printexcursions{}
4654 \newcommand\excursionref[2]{% label, text
4655   \bool_if:NT \c__mikoslices_notes_bool {
4656     \begin{omtext}[title=Excursion]
4657       #2 \sref[fallback=the appendix]{#1}.
4658     \end{omtext}
4659   }
4660 }
4661 \newcommand\activate@excursion[2][{}{
4662   \gappto\printexcursions{\inputref[#1]{#2}}
```

```

4663 }
4664 \newcommand\excursion[4][{}]{% repos, label, path, text
4665   \bool_if:NT \c__mikoslides_notes_bool {
4666     \activate@excursion[#1]{#3}\excursionref{#2}{#4}
4667   }
4668 }

```

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

## \excursiongroup

```

4669 \keys_define:nn{mikoslides / excursiongroup }{
4670   id          .str_set_x:N = \l__mikoslides_excursion_id_str,
4671   intro       .tl_set:N   = \l__mikoslides_excursion_intro_tl,
4672   mhrepos     .str_set_x:N = \l__mikoslides_excursion_mhrepos_str
4673 }
4674 \cs_new_protected:Nn \__mikoslides_excursion_args:n {
4675   \tl_clear:N \l__mikoslides_excursion_intro_tl
4676   \str_clear:N \l__mikoslides_excursion_id_str
4677   \str_clear:N \l__mikoslides_excursion_mhrepos_str
4678   \keys_set:nn {mikoslides / excursiongroup }{ #1 }
4679 }
4680 \newcommand\excursiongroup[1][{}]{
4681   \__mikoslides_excursion_args:n{ #1 }
4682   \ifdefempty\printexcursions{}% only if there are excursions
4683   {\begin{note}
4684     \begin{omgroup}[#1]{Excursions}%
4685       \ifdefempty\l__mikoslides_excursion_intro_tl{{
4686         \inputref[\l__mikoslides_excursion_mhrepos_str]{
4687           \l__mikoslides_excursion_intro_tl
4688         }
4689       }
4690       \printexcursions%
4691     \end{omgroup}
4692   \end{note}}
4693 }
4694 \end{package}

```

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

## Chapter 33

# The Implementation

### 33.1 Package Options

The first step is to declare (a few) package options that handle whether certain information is printed or not. They all come with their own conditionals that are set by the options.

```
4695 <*package>
4696 <@@=problems>
4697 \ProvidesExplPackage{problem}{2019/03/20}{1.3}{Semantic Markup for Problems}
4698 \RequirePackage{l3keys2e,expl-keystr-compat}
4699
4700 \keys_define:nn { problem / pkg }{
4701   notes      .default:n    = { true },
4702   notes      .bool_set:N   = \c__problems_notes_bool,
4703   gnotes     .default:n    = { true },
4704   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
4705   hints      .default:n    = { true },
4706   hints      .bool_set:N   = \c__problems_hints_bool,
4707   solutions  .default:n    = { true },
4708   solutions  .bool_set:N   = \c__problems_solutions_bool,
4709   pts        .default:n    = { true },
4710   pts        .bool_set:N   = \c__problems_pts_bool,
4711   min        .default:n    = { true },
4712   min        .bool_set:N   = \c__problems_min_bool,
4713   boxed      .default:n    = { true },
4714   boxed      .bool_set:N   = \c__problems_boxed_bool,
4715   unknown    .code:n       = {}
4716 }
4717 \def\solutionstrue{
4718   \bool_set_true:N \c__problems_solutions_bool
4719 }
4720 \def\solutionsfalse{
4721   \bool_set_false:N \c__problems_solutions_bool
4722 }
4723
4724 \ProcessKeysOptions{ problem / pkg }
```

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



```

4725 \RequirePackage{stex-compatibility}
4726 \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.

```

4727 \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.

```

4728 \def\prob@problem@kw{Problem}
4729 \def\prob@solution@kw{Solution}
4730 \def\prob@hint@kw{Hint}
4731 \def\prob@note@kw{Note}
4732 \def\prob@gnote@kw{Grading}
4733 \def\prob@pt@kw{pt}
4734 \def\prob@min@kw{min}

```

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

For the other languages, we set up triggers

```

4735 \@ifpackageloaded{babel}{
4736   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4737   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4738     \input{problem-ngerman.ldf}
4739   }
4740   \clist_if_in:NnT \l_tmpa_clist {finnish}{
4741     \input{problem-finnish.ldf}
4742   }
4743   \clist_if_in:NnT \l_tmpa_clist {french}{
4744     \input{problem-french.ldf}
4745   }
4746   \clist_if_in:NnT \l_tmpa_clist {russian}{
4747     \input{problem-russian.ldf}
4748   }
4749 }{}

```

## 33.2 Problems and Solutions

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

```

4750 \keys_define:nn{ problem / problem }{
4751   id      .str_set:x:N = \l__problems_prob_id_str,
4752   pts     .tl_set:N    = \l__problems_prob_pts_tl,
4753   min     .tl_set:N    = \l__problems_prob_min_tl,
4754   title   .tl_set:N    = \l__problems_prob_title_tl,
4755   refnum  .int_set:N    = \l__problems_prob_refnum_int
4756 }
4757 \cs_new_protected:Nn \__problems_prob_args:n {
4758   \str_clear:N \l__problems_prob_id_str
4759   \tl_clear:N \l__problems_prob_pts_tl
4760   \tl_clear:N \l__problems_prob_min_tl
4761   \tl_clear:N \l__problems_prob_title_tl

```

```

4762 \int_zero_new:N \l__problems_prob_refnum_int
4763 \keys_set:nn { problem / problem }{ #1 }
4764 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
4765   \let\l__problems_inclprob_refnum_int\undefined
4766 }
4767 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

4768 \newcounter{problem}
4769 \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.

```

4770 \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

```

4771 \newcommand\prob@number{
4772   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
4773     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
4774   }{
4775     \int_if_exist:NTF \l__problems_prob_refnum_int {
4776       \prob@label{\int_use:N \l__problems_prob_refnum_int }
4777     }{
4778       \prob@label\theproblem
4779     }
4780   }
4781 }

```

(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.

```

4782 \newcommand\prob@title[3]{%
4783   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
4784     #2 \l__problems_inclprob_title_tl #3
4785   }{
4786     \tl_if_exist:NTF \l__problems_prob_title_tl {
4787       #2 \l__problems_prob_title_tl #3
4788     }{
4789       #1
4790     }
4791   }
4792 }

```

(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.

```

4793 \def\prob@heading{
4794   \prob@problem@kw~\prob@number\prob@title{~}{~}{~}\strut}
4795   %\sref@label{id{\prob@problem@kw~\prob@number}}{~}
4796 }

```

(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`

```

4797 \newenvironment{problem}[1][1]{
4798   \_problems_prob_args:n{#1}%\sref@target%
4799   \@in@omtexttrue% we are in a statement (for inline definitions)
4800   \stepcounter{problem}\record@problem
4801   \def\current@section@level{\prob@problem@kw}
4802   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
4803 }%
4804 {\smallskip}
4805 \bool_if:NT \c__problems_boxed_bool {
4806   \surroundwithmdframed{problem}
4807 }

```

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

```

4808 \def\record@problem{
4809   \protected@write\@auxout{}
4810   {
4811     \string\@problem{\prob@number}
4812     {
4813       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
4814         \l__problems_inclprob_pts_tl
4815       }{
4816         \l__problems_prob_pts_tl
4817       }
4818     }%
4819     {
4820       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
4821         \l__problems_inclprob_min_tl
4822       }{
4823         \l__problems_prob_min_tl
4824       }
4825     }
4826   }
4827 }

```

(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).

```

4828 \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.

```

4829 \keys_define:nn { problem / solution }{
4830   id          .str_set_x:N = \l__problems_solution_id_str ,
4831   for         .tl_set:N    = \l__problems_solution_for_tl ,
4832   height      .dim_set:N   = \l__problems_solution_height_dim ,
4833   creators    .clist_set:N = \l__problems_solution_creators_clist ,
4834   contributors .clist_set:N = \l__problems_solution_contributors_clist ,
4835   srccite     .tl_set:N    = \l__problems_solution_srccite_tl
4836 }
4837 \cs_new_protected:Nn \__problems_solution_args:n {
4838   \str_clear:N \l__problems_solution_id_str
4839   \tl_clear:N \l__problems_solution_for_tl
4840   \tl_clear:N \l__problems_solution_srccite_tl
4841   \clist_clear:N \l__problems_solution_creators_clist
4842   \clist_clear:N \l__problems_solution_contributors_clist
4843   \dim_zero:N \l__problems_solution_height_dim
4844   \keys_set:nn { problem / solution }{ #1 }
4845 }

```

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

```

4846 \newcommand\@startsolution[1][ ]{
4847   \__problems_solution_args:n { #1 }
4848   \@in@omtexttrue% we are in a statement.
4849   \bool_if:NF \c__problems_boxed_bool { \hrule }
4850   \smallskip\noindent
4851   {\textbf\prob@solution@kw : \enspace}
4852   \begin{small}
4853   \def\current@section@level{\prob@solution@kw}
4854   \ignorespacesandpars
4855 }

```

**\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.

```

4856 \newcommand\startsolutions{
4857   \specialcomment{solution}{\@startsolution}{
4858     \bool_if:NF \c__problems_boxed_bool {
4859       \hrule\medskip
4860     }
4861     \end{small}%
4862   }
4863   \bool_if:NT \c__problems_boxed_bool {
4864     \surroundwithmdframed{solution}
4865   }
4866 }

```

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

**\stopsolutions**

```

4867 \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.

```

4868 \bool_if:NTF \c_problems_solutions_bool {
4869   \startsolutions
4870 }{
4871   \stopsolutions
4872 }

```

**exnote**

```

4873 \bool_if:NTF \c_problems_notes_bool {
4874   \newenvironment{exnote}[1][]{
4875     \par\smallskip\hrule\smallskip
4876     \noindent\textbf{\prob@note@kw : }\small
4877   }{
4878     \smallskip\hrule
4879   }
4880 }{
4881   \excludecomment{exnote}
4882 }

```

**hint**

```

4883 \bool_if:NTF \c_problems_notes_bool {
4884   \newenvironment{hint}[1][]{
4885     \par\smallskip\hrule\smallskip
4886     \noindent\textbf{\prob@hint@kw :~ }\small
4887   }{
4888     \smallskip\hrule
4889   }
4890   \newenvironment{exhint}[1][]{
4891     \par\smallskip\hrule\smallskip
4892     \noindent\textbf{\prob@hint@kw :~ }\small
4893   }{
4894     \smallskip\hrule
4895   }
4896 }{
4897   \excludecomment{hint}
4898   \excludecomment{exhint}
4899 }

```

**gnote**

```

4900 \bool_if:NTF \c_problems_notes_bool {
4901   \newenvironment{gnote}[1][]{
4902     \par\smallskip\hrule\smallskip
4903     \noindent\textbf{\prob@gnote@kw : }\small
4904   }{
4905     \smallskip\hrule
4906   }
4907 }{
4908   \excludecomment{gnote}
4909 }

```

### 33.3 Multiple Choice Blocks

```

4910 \newenvironment{mcb}{
4911   \begin{enumerate}
4912 }{
4913   \end{enumerate}
4914 }

```

we define the keys for the mcc macro

```

4915 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
4916   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
4917     \bool_set_true:N #1
4918   }{
4919     \bool_set_false:N #1
4920   }
4921 }
4922 \keys_define:nn { problem / mcc }{
4923   id          .str_set_x:N = \l__problems_mcc_id_str ,
4924   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
4925   T           .default:n   = { true } ,
4926   T           .bool_set:N  = \l__problems_mcc_t_bool ,
4927   F           .default:n   = { true } ,
4928   F           .bool_set:N  = \l__problems_mcc_f_bool ,
4929   Ttext       .code:n      = {
4930     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
4931   } ,
4932   Ftext       .code:n      = {
4933     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
4934   }
4935 }
4936 \cs_new_protected:Nn \l__problems_mcc_args:n {
4937   \str_clear:N \l__problems_mcc_id_str
4938   \tl_clear:N \l__problems_mcc_feedback_tl
4939   \bool_set_true:N \l__problems_mcc_t_bool
4940   \bool_set_true:N \l__problems_mcc_f_bool
4941   \bool_set_true:N \l__problems_mcc_Ttext_bool
4942   \bool_set_false:N \l__problems_mcc_Ftext_bool
4943   \keys_set:nn { problem / mcc }{ #1 }
4944 }

```

\mcc

```

4945 \newcommand\mcc[2][] {
4946   \l__problems_mcc_args:n{ #1 }
4947   \item #2
4948   \bool_if:NT \c__problems_solutions_bool {
4949     \
4950     \bool_if:NT \l__problems_mcc_t_bool {
4951       % TODO!
4952       % \ifcsstring{mcc@T}{T}{ }\{ \mcc@Ttext }%
4953     }
4954     \bool_if:NT \l__problems_mcc_f_bool {

```

---

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

```

4955     % TODO!
4956     % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
4957 }
4958 \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
4959   !
4960 }{
4961   \l__problems_mcc_feedback_tl
4962 }
4963 }
4964 } %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.

```

4965
4966 \keys_define:nn{ problem / inclproblem }{
4967   % id      .str_set_x:N = \l__problems_inclprob_id_str,
4968   pts      .tl_set:N    = \l__problems_inclprob_pts_tl,
4969   min      .tl_set:N    = \l__problems_inclprob_min_tl,
4970   title    .tl_set:N    = \l__problems_inclprob_title_tl,
4971   refnum   .int_set:N    = \l__problems_inclprob_refnum_int,
4972   mhrepos  .str_set_x:N = \l__problems_inclprob_mhrepos_str
4973 }
4974 \cs_new_protected:Nn \l__problems_inclprob_args:n {
4975   % \str_clear:N \l__problems_prob_id_str
4976   \tl_clear:N \l__problems_inclprob_pts_tl
4977   \tl_clear:N \l__problems_inclprob_min_tl
4978   \tl_clear:N \l__problems_inclprob_title_tl
4979   \int_zero_new:N \l__problems_inclprob_refnum_int
4980   \str_clear:N \l__problems_inclprob_mhrepos_str
4981   \keys_set:nn { problem / inclproblem }{ #1 }
4982   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
4983     \let\l__problems_inclprob_pts_tl\undefined
4984   }
4985   \tl_if_empty:NT \l__problems_inclprob_min_tl {
4986     \let\l__problems_inclprob_min_tl\undefined
4987   }
4988   \tl_if_empty:NT \l__problems_inclprob_title_tl {
4989     \let\l__problems_inclprob_title_tl\undefined
4990   }
4991   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
4992     \let\l__problems_inclprob_refnum_int\undefined
4993   }
4994 }
4995
4996 \cs_new_protected:Nn \l__problems_inclprob_clear: {
4997   % \str_clear:N \l__problems_prob_id_str
4998   \let\l__problems_inclprob_pts_tl\undefined
4999   \let\l__problems_inclprob_min_tl\undefined

```

```

5000 \let\l__problems_inclprob_title_tl\undefined
5001 \let\l__problems_inclprob_refnum_int\undefined
5002 \let\l__problems_inclprob_mhrepos_str\undefined
5003 }
5004
5005 \newcommand\includeproblem[2][ ]{
5006   \__problems_inclprob_args:n{ #1 }
5007   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
5008     \input{#2}
5009   }{
5010     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
5011       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
5012     }
5013   }
5014   \__problems_inclprob_clear:
5015 }

```

(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.

```

5016 \AddToHook{enddocument}{
5017   \bool_if:NT \c__problems_pts_bool {
5018     \message{Total:~\arabic{pts}~points}
5019   }
5020   \bool_if:NT \c__problems_min_bool {
5021     \message{Total:~\arabic{min}~minutes}
5022   }
5023 }

```

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

```

5024 \def\pts#1{
5025   \bool_if:NT \c__problems_pts_bool {
5026     \marginpar{#1~\prob@pt@kw}
5027   }
5028 }
5029 \def\min#1{
5030   \bool_if:NT \c__problems_min_bool {
5031     \marginpar{#1~\prob@min@kw}
5032   }
5033 }

```

**\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.

```

5034 \newcounter{pts}
5035 \def\show@pts{
5036   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
5037     \bool_if:NT \c__problems_pts_bool {
5038       \marginpar{\l__problems_inclprob_pts_tl;\prob@pt@kw\smallskip}
5039       \addtocounter{pts}{\l__problems_inclprob_pts_tl}

```



```

5040     }
5041   }{
5042     \tl_if_exist:NT \l__problems_prob_pts_tl {
5043       \bool_if:NT \c__problems_pts_bool {
5044         \marginpar{\l__problems_prob_pts_tl;\prob@pt@kw\smallskip}
5045         \addtocounter{pts}{\l__problems_prob_pts_tl}
5046       }
5047     }
5048   }
5049 }

```

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

\show@min

```

5050 \newcounter{min}
5051 \def\show@min{
5052   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
5053     \bool_if:NT \c__problems_min_bool {
5054       \marginpar{\l__problems_inclprob_pts_tl;min}
5055       \addtocounter{min}{\l__problems_inclprob_min_tl}
5056     }
5057   }{
5058     \tl_if_exist:NT \l__problems_prob_min_tl {
5059       \bool_if:NT \c__problems_min_bool {
5060         \marginpar{\l__problems_prob_min_tl;min}
5061         \addtocounter{min}{\l__problems_prob_min_tl}
5062       }
5063     }
5064   }
5065 }
5066 \</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.

```
5067 <@@=hwexam>
5068 <*cls>
5069 \ProvidesExplClass{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
5070 \RequirePackage{l3keys2e,expl-keystr-compatible}
5071 \DeclareOption*{
5072   \PassOptionsToClass{\CurrentOption}{omdoc}
5073   \PassOptionsToPackage{\CurrentOption}{stex}
5074   \PassOptionsToPackage{\CurrentOption}{hwexam}
5075   \PassOptionsToPackage{\CurrentOption}{tikzinput}
5076 }
5077 \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.

```
5078 \LoadClass{omdoc}
5079 \RequirePackage{stex}
5080 \RequirePackage{hwexam}
5081 \RequirePackage{tikzinput}
5082 \RequirePackage{graphicx}
5083 \RequirePackage{a4wide}
5084 \RequirePackage{amssymb}
5085 \RequirePackage{amstext}
5086 \RequirePackage{amsmath}
```

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

```

5087 \newcommand\assig@default@type{\hwexam@assignment@kw}
5088 \def\document@hwexamtype{\assig@default@type}
5089 <@@=document_structure>
5090 \keys_define:nn { document-structure / document }{
5091 id .str_set_x:N = \c_document_structure_document_id_str,
5092 hwexamtype .tl_set:N = \document@hwexamtype
5093 }
5094 <@@=hwexam>
5095 </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.

```
5096 \*package>
5097 \ProvidesExplPackage{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
5098 \RequirePackage{l3keys2e,expl-keystr-compat}
5099
5100 \newif\iftest\testfalse
5101 \DeclareOption{test}{\testtrue}
5102 \newif\ifmultiple\multiplefalse
5103 \DeclareOption{multiple}{\multipletrue}
5104 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
5105 \ProcessOptions
```

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

```
5106 \RequirePackage{keyval}[1997/11/10]
5107 \RequirePackage{problem}
```

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

```
5108 \newcommand\hwexam@assignment@kw{Assignment}
5109 \newcommand\hwexam@given@kw{Given}
5110 \newcommand\hwexam@due@kw{Due}
5111 \newcommand\hwexam@testemptypage@kw{This page was intentionally left blank for extra
5112 space}%
5113 \newcommand\correction@probs@kw{prob.}%
5114 \newcommand\correction@pts@kw{total}%
5115 \newcommand\correction@reached@kw{reached}%
5116 \newcommand\correction@sum@kw{Sum}%
5117 \newcommand\correction@grade@kw{grade}%
5118 \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

```

5119 \ifpackageloaded{babel}{\RequirePackage[base]{babel}}
5120
5121 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5122 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5123   \input{hwexam-ngerman.ldf}
5124 }
5125 \clist_if_in:NnT \l_tmpa_clist {finnish}{
5126   \input{hwexam-finnish.ldf}
5127 }
5128 \clist_if_in:NnT \l_tmpa_clist {french}{
5129   \input{hwexam-french.ldf}
5130 }
5131 \clist_if_in:NnT \l_tmpa_clist {russian}{
5132   \input{hwexam-russian.ldf}
5133 }

```

## 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.

```

5134 \newcounter{assignment}
5135 \numberproblemsin{assignment}
5136 \renewcommand\prob@label[1]{\arabic{assignment}.#1}

```

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

```

5137 \keys_define:nn { hwexam / assignment } {
5138   id .str_set:N = \l__hwexam_assign_id_str,
5139   number .int_set:N = \l__hwexam_assign_number_int,
5140   title .tl_set:N = \l__hwexam_assign_title_tl,
5141   type .tl_set:N = \l__hwexam_assign_type_tl,
5142   given .tl_set:N = \l__hwexam_assign_given_tl,
5143   due .tl_set:N = \l__hwexam_assign_due_tl,
5144   loadmodules .code:n = {
5145     \bool_set_true:N \l__hwexam_assign_loadmodules_bool
5146   }
5147 }
5148 \cs_new_protected:Nn \__hwexam_assignment_args:n {
5149   \str_clear:N \l__hwexam_assign_id_str
5150   \int_set:Nn \l__hwexam_assign_number_int {-1}
5151   \tl_clear:N \l__hwexam_assign_title_tl
5152   \tl_clear:N \l__hwexam_assign_type_tl
5153   \tl_clear:N \l__hwexam_assign_given_tl
5154   \tl_clear:N \l__hwexam_assign_due_tl
5155   \bool_set_false:N \l__hwexam_assign_loadmodules_bool
5156   \keys_set:nn { hwexam / assignment }{ #1 }
5157 }

```

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.

```

5158 \newcommand\given@due[2]{
5159 \bool_lazy_all:nF {
5160 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
5161 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
5162 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
5163 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
5164 }{ #1 }
5165
5166 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
5167 \tl_if_empty:NF \l__hwexam_assign_given_tl {
5168 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
5169 }
5170 }{
5171 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
5172 }
5173
5174 \bool_lazy_or:nnF {
5175 \bool_lazy_and_p:nn {
5176 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
5177 }{
5178 \tl_if_empty_p:V \l__hwexam_assign_due_tl
5179 }
5180 }{
5181 \bool_lazy_and_p:nn {
5182 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
5183 }{
5184 \tl_if_empty_p:V \l__hwexam_assign_due_tl
5185 }
5186 }{ ,~ }
5187
5188 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
5189 \tl_if_empty:NF \l__hwexam_assign_due_tl {
5190 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
5191 }
5192 }{
5193 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
5194 }
5195
5196 \bool_lazy_all:nF {
5197 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
5198 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
5199 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
5200 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
5201 }{ #2 }
5202 }

```

`\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.

```

5203 \newcommand\assignment@title[3]{

```

```

5204 \tl_if_empty:NTF \l__hwexam_inclassassign_title_tl {
5205 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
5206 #1
5207 }{
5208 #2\l__hwexam_assign_title_tl#3
5209 }
5210 }{
5211 #2\l__hwexam_inclassassign_title_tl#3
5212 }
5213 }

```

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

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

```

5214 \newcommand\assignment@number{
5215 \int_compare:nNnTF \l__hwexam_inclassassign_number_int = {-1} {
5216 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
5217 \int_use:N \l__hwexam_assign_number_int
5218 }
5219 }{
5220 \int_use:N \l__hwexam_inclassassign_number_int
5221 }
5222 }

```

(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.

```

5223 \newenvironment{assignment}[1][]{
5224 \__hwexam_assignment_args:n { #1 }
5225 %\sref@target
5226 \let\__hwexamnum\l__hwexam_assign_number_int
5227 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
5228 \stepcounter{assignment}
5229 }{
5230 \setcounter{assignment}{\int_use:N\__hwexamnum}
5231 }
5232 \setcounter{problem}{0}
5233 \def\current@section@level{\document@hwexamtype}
5234 %\sref@label@id{\document@hwexamtype \thesection}
5235 \begin{@assignment}
5236 }{
5237 \end{@assignment}
5238 }

```

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

```

5239 \def\__hwexasstitle{
5240 \protect\document@hwexamtype~\arabic{assignment}
5241 \assignment@title{}\;{}{}\; -- \given@due{}\}
5242 }

```

```

5243 \ifmultiple
5244 \newenvironment{@assignment}{
5245 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
5246 \begin{omgroup}[loadmodules]{\__hwexasstitle}
5247 }{
5248 \begin{omgroup}{\__hwexasstitle}
5249 }
5250 }{
5251 \end{omgroup}
5252 }

```

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

```

5253 \else
5254 \newenvironment{@assignment}{
5255 \begin{center}\bf
5256 \Large\@title\strut\
5257 \document@hwexamtype~\arabic{assignment}\assignment@title{\;}{:};{\}\}
5258 \large\given@due{--\;}{\;}{--}
5259 \end{center}
5260 }{}
5261 \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.

```

5262 \keys_define:nn { hwexam / inclassignment } {
5263 %id .str_set_x:N = \l__hwexam_assign_id_str,
5264 number .int_set:N = \l__hwexam_inclassign_number_int,
5265 title .tl_set:N = \l__hwexam_inclassign_title_tl,
5266 type .tl_set:N = \l__hwexam_inclassign_type_tl,
5267 given .tl_set:N = \l__hwexam_inclassign_given_tl,
5268 due .tl_set:N = \l__hwexam_inclassign_due_tl,
5269 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
5270 }
5271 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
5272 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
5273 \tl_clear:N \l__hwexam_inclassign_title_tl
5274 \tl_clear:N \l__hwexam_inclassign_type_tl
5275 \tl_clear:N \l__hwexam_inclassign_given_tl
5276 \tl_clear:N \l__hwexam_inclassign_due_tl
5277 \str_clear:N \l__hwexam_inclassign_mhrepos_str
5278 \keys_set:nn { hwexam / inclassignment }{ #1 }
5279 }
5280 \__hwexam_inclassignment_args:n {}
5281
5282 \newcommand\inputassignment[2][ ]{
5283 \__hwexam_inclassignment_args:n { #1 }
5284 \str_if_empty:NTF \l__hwexam_inclassign_mhrepos_str {
5285 \input{#2}
5286 }{
5287 \stex_in_repository:nn{\l__hwexam_inclassign_mhrepos_str}{

```



```

5288 \input{\mhp\path{\l__hwexam_inclasssign_mhrepos_str}{#2}}
5289 }
5290 }
5291 \__hwexam_inclasssign_args:n {}
5292 }
5293 \newcommand\includeassignment[2][ ]{
5294 \newpage
5295 \inputassignment[#1]{#2}
5296 }

```

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

## 35.4 Typesetting Exams

\quizheading

```

5297 \ExplSyntaxOff
5298 \newcommand\quizheading[1]{%
5299 \def\@tas{#1}%
5300 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
5301 \ifx\@tas\empty\else%
5302 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
5303 \fi%
5304 }
5305 \ExplSyntaxOn

```

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

\testheading

```

5306 \keys_define:nn { hwexam / testheading } {
5307 min .tl_set:N = \l__hwexam_testheading_min_tl,
5308 duration .tl_set:N = \l__hwexam_testheading_duration_tl,
5309 reqpts .tl_set:N = \l__hwexam_testheading_reqpts_tl
5310 }
5311 \cs_new_protected:Nn \__hwexam_testheading_args:n {
5312 \tl_clear:N \l__hwexam_testheading_min_tl
5313 \tl_clear:N \l__hwexam_testheading_duration_tl
5314 \tl_clear:N \l__hwexam_testheading_reqpts_tl
5315 \keys_set:nn { hwexam / testheading }{ #1 }
5316 }
5317 \newenvironment{testheading}[1][ ]{
5318 \__hwexam_testheading_args:n{ #1 }
5319 \noindent\large{Name:~\hfill
5320 Matriculation Number:\hspace*{2cm}\strut}\[1ex]
5321 \begin{center}
5322 \Large\textbf{\@title}\[1ex]
5323 \large\@date\[3ex]
5324 \end{center}
5325 \textbf{You~have~
5326 \tl_if_empty:NTF \l__hwexam_testheading_duration_tl {
5327 \l__hwexam_testheading_min_tl~minutes
5328 }{
5329 \l__hwexam_testheading_duration_tl
5330 }~

```

```

5331 (sharp)~for~the~test
5332 };\
5333 Write~the~solutions~to~the~sheet.
5334 \par\noindent
5335 \newcount\check@time\check@time=\l__hwexam_testheading_min_tl
5336 \advance\check@time by -\theassignment@totalmin
5337 The~estimated~time~for~solving~this~exam~is~
5338 {\theassignment@totalmin}-minutes,~
5339 leaving~you~{\the\check@time}-minutes~for~revising~
5340 your~exam.
5341
5342 \par\noindent
5343 \newcount\bonus@pts\bonus@pts=\theassignment@totalpts
5344 \advance\bonus@pts by -\l__hwexam_testheading_reqpts_tl
5345 You~can~reach~{\theassignment@totalpts}-points~if~you~
5346 solve~all~problems.~You~will~only~need~
5347 {\l__hwexam_testheading_reqpts_tl}-points~for~a~perfect~score,~
5348 i.e.~\ {\the\bonus@pts}-points~are~bonus~points.
5349 \vfill
5350 \begin{center}
5351 {
5352 \Large\em You~have~ample~time,~so~take~it~slow~
5353 and~avoid~rushing~to~mistakes!\}[2ex]
5354 Different~problems~test~different~skills~and~
5355 knowledge,~so~do~not~get~stuck~on~one~problem.
5356 }
5357 \vfill\par\resizebox{\textwidth}{!}{\correction@table}\}[3ex]
5358 \end{center}
5359 }{
5360 \newpage
5361 }

```

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

\testspace

```

5362 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

5363 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

5364 \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.

```

5365 <@=problems>
5366 \renewcommand\@problem[3]{
5367 \stepcounter{assignment@probs}
5368 \def\__problemspts{#2}

```

```

5369 \ifx\__problemspts\@empty\else
5370 \addtocounter{assignment@totalpts}{#2}
5371 \fi
5372 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
5373 \xdef\correction@probs{\correction@probs & #1}%
5374 \xdef\correction@pts{\correction@pts & #2}
5375 \xdef\correction@reached{\correction@reached & }
5376 }
5377 <@@=hwexam>

```

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

\correction@table This macro generates the correction table

```

5378 \newcounter{assignment@probs}
5379 \newcounter{assignment@totalpts}
5380 \newcounter{assignment@totalmin}
5381 \def\correction@probs{\correction@probs@kw}%
5382 \def\correction@pts{\correction@pts@kw}%
5383 \def\correction@reached{\correction@reached@kw}%
5384 \def\after@correction@table{}%
5385 \stepcounter{assignment@probs}
5386 \newcommand\correction@table{
5387 \resizebox{\textwidth}{!}{%
5388 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
5389 &\multicolumn{\theassignment@probs}{c|}|%|
5390 {\footnotesize\correction@forgrading@kw} &\\ \hline
5391 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
5392 \correction@pts & \theassignment@totalpts & \\ \hline
5393 \correction@reached & & \[.7cm]\hline
5394 \end{tabular}}
5395 \ifx\after@correction@table\@empty\else\strut\par\noindent\after@correction@table\fi}
5396 </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}

```