

# The sTeX3 Package \*

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

2022-02-09

## Abstract

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

sTeX augments L<sup>A</sup>T<sub>E</sub>X with

- *Semantic macros* that denote and distinguish between mathematical concepts, operators, etc. independent of their notational presentation,
- A powerful *module system* that allows for authoring and importing individual fragments containing document text and/or semantic macros, independent of – and without hard coding – directory paths relative to the current document,
- A mechanism for exporting sTeX documents to (modular) XHTML, preserving all the semantic information for semantically informed knowledge management services.

This is the full documentation of sTeX. It consists of four parts:

- **Part I** is a general manual for the sTeX package and associated software. It is primarily directed at end-users who want to use sTeX to author semantically enriched documents.
- **Part II** documents the macros provided by the sTeX package. It is primarily directed at package authors who want to build on sTeX, but can also serve as a reference manual for end-users.
- **Part III** documents additional packages that build on sTeX, primarily its module system. These are not part of the sTeX package itself, but useful additions enabled by sTeX package functionality.
- **Part IV** is the detailed documentation of the sTeX package implementation.

---

\*Version 3.0 (last revised 2022-02-09)

# Contents

<b>I</b>	<b>Manual</b>	<b>1</b>
<b>1</b>	<b>What is sTeX?</b>	<b>2</b>
<b>2</b>	<b>Quickstart</b>	<b>3</b>
2.1	Setup	3
2.1.1	The sTeX IDE	3
2.1.2	Manual Setup	3
2.2	A First sTeX Document	4
<b>3</b>	<b>Using Semantic Macros</b>	<b>6</b>
<b>4</b>	<b>sTeX Archives</b>	<b>7</b>
4.1	The Local MathHub-Directory	7
4.2	The Structure of sTeX Archives	7
4.3	MANIFEST.MF-Files	8
<b>5</b>	<b>Creating New Modules and Symbols</b>	<b>9</b>
5.1	Advanced Structuring Mechanisms	9
5.2	Primitive Symbols (The sTeX Metatheory)	10
<b>6</b>	<b>sTeX Statements (Definitions, Theorems, Examples, ...)</b>	<b>11</b>
<b>7</b>	<b>Additional Packages</b>	<b>12</b>
7.1	Modular Document Structuring	12
7.2	Slides and Course Notes	12
7.3	Homework, Problems and Exams	12
<b>8</b>	<b>Stuff</b>	<b>13</b>
8.1	Modules	13
8.1.1	Semantic Macros and Notations	13
	Other Argument Types	15
	Precedences	17
8.1.2	Archives and Imports	17
	Namespaces	17
	Paths in Import-Statements	18
<b>II</b>	<b>Documentation</b>	<b>19</b>
<b>9</b>	<b>sTeX-Basics</b>	<b>20</b>
9.1	Macros and Environments	20
<b>10</b>	<b>sTeX-MathHub</b>	<b>22</b>
10.1	Macros and Environments	22
10.1.1	Files, Paths, URIs	22
10.1.2	MathHub Archives	23

<b>11</b>	<b>sTeX-References</b>	<b>25</b>
11.1	Macros and Environments . . . . .	25
<b>12</b>	<b>sTeX-Modules</b>	<b>26</b>
12.1	Macros and Environments . . . . .	26
12.1.1	The <code>module</code> -environment . . . . .	28
<b>13</b>	<b>sTeX-Module Inheritance</b>	<b>31</b>
13.1	Macros and Environments . . . . .	31
13.1.1	SMS Mode . . . . .	31
13.1.2	Imports and Inheritance . . . . .	32
<b>14</b>	<b>sTeX-Symbols</b>	<b>35</b>
14.1	Macros and Environments . . . . .	35
<b>15</b>	<b>sTeX-Terms</b>	<b>38</b>
15.1	Macros and Environments . . . . .	38
<b>16</b>	<b>sTeX-Structural Features</b>	<b>41</b>
16.1	Macros and Environments . . . . .	41
16.1.1	Structures . . . . .	41
<b>17</b>	<b>sTeX-Statements</b>	<b>42</b>
17.1	Macros and Environments . . . . .	42
<b>18</b>	<b>sTeX-Proofs: Structural Markup for Proofs</b>	<b>43</b>
18.1	Introduction . . . . .	45
18.2	The User Interface . . . . .	46
18.2.1	Package Options . . . . .	46
18.2.2	Proofs and Proof steps . . . . .	46
18.2.3	Justifications . . . . .	46
18.2.4	Proof Structure . . . . .	47
18.2.5	Proof End Markers . . . . .	48
18.2.6	Configuration of the Presentation . . . . .	48
18.3	Limitations . . . . .	48
<b>19</b>	<b>sTeX-Metatheory</b>	<b>50</b>
19.1	Symbols . . . . .	50
<b>III</b>	<b>Extensions</b>	<b>51</b>
<b>20</b>	<b>Tikzinput</b>	<b>52</b>
20.1	Macros and Environments . . . . .	52

<b>21</b>	<b>document-structure.sty: Semantic Markup for Open Mathematical Documents in L<sup>A</sup>T<sub>E</sub>X</b>	<b>53</b>
21.1	Introduction . . . . .	53
21.2	The User Interface . . . . .	54
21.2.1	Package and Class Options . . . . .	54
21.2.2	Document Structure . . . . .	54
21.2.3	Ignoring Inputs . . . . .	55
21.2.4	Structure Sharing . . . . .	56
21.2.5	Global Variables . . . . .	56
21.2.6	Colors . . . . .	57
21.3	Limitations . . . . .	57
<b>22</b>	<b>Slides and Course Notes</b>	<b>58</b>
22.1	Introduction . . . . .	58
22.2	The User Interface . . . . .	58
22.2.1	Package Options . . . . .	58
22.2.2	Notes and Slides . . . . .	59
22.2.3	Header and Footer Lines of the Slides . . . . .	60
22.2.4	Frame Images . . . . .	60
22.2.5	Colors and Highlighting . . . . .	61
22.2.6	Front Matter, Titles, etc. . . . .	61
22.2.7	Excursions . . . . .	61
22.2.8	Miscellaneous . . . . .	61
22.3	Limitations . . . . .	61
<b>23</b>	<b>problem.sty: An Infrastructure for formatting Problems</b>	<b>62</b>
23.1	Introduction . . . . .	62
23.2	The User Interface . . . . .	62
23.2.1	Package Options . . . . .	62
23.2.2	Problems and Solutions . . . . .	63
23.2.3	Multiple Choice Blocks . . . . .	64
23.2.4	Including Problems . . . . .	64
23.2.5	Reporting Metadata . . . . .	64
23.3	Limitations . . . . .	64
<b>24</b>	<b>hwexam.sty/cls: An Infrastructure for formatting Assignments and Exams</b>	<b>66</b>
24.1	Introduction . . . . .	67
24.2	The User Interface . . . . .	67
24.2.1	Package and Class Options . . . . .	67
24.2.2	Assignments . . . . .	67
24.2.3	Typesetting Exams . . . . .	67
24.2.4	Including Assignments . . . . .	68
24.3	Limitations . . . . .	68
<b>IV</b>	<b>Implementation</b>	<b>70</b>

<b>25</b>	<b>STeX-Basics Implementation</b>	<b>71</b>
25.1	The STeXDocument Class . . . . .	71
25.2	Preliminaries . . . . .	71
25.3	Messages and logging . . . . .	72
25.4	Persistence . . . . .	73
25.5	HTML Annotations . . . . .	73
25.6	Languages . . . . .	76
25.7	Activating/Deactivating Macros . . . . .	77
<b>26</b>	<b>STeX-MathHub Implementation</b>	<b>79</b>
26.1	Generic Path Handling . . . . .	79
26.2	PWD and kpsewhich . . . . .	81
26.3	File Hooks and Tracking . . . . .	82
26.4	MathHub Repositories . . . . .	83
<b>27</b>	<b>STeX-References Implementation</b>	<b>90</b>
27.1	Document URIs and URLs . . . . .	90
27.2	Setting Reference Targets . . . . .	92
27.3	Using References . . . . .	93
<b>28</b>	<b>STeX-Modules Implementation</b>	<b>95</b>
28.1	The module environment . . . . .	98
28.2	Invoking modules . . . . .	104
<b>29</b>	<b>STeX-Module Inheritance Implementation</b>	<b>106</b>
29.1	SMS Mode . . . . .	106
29.2	Inheritance . . . . .	110
<b>30</b>	<b>STeX-Symbols Implementation</b>	<b>115</b>
30.1	Symbol Declarations . . . . .	115
30.2	Notations . . . . .	122
<b>31</b>	<b>STeX-Terms Implementation</b>	<b>131</b>
31.1	Symbol Invocations . . . . .	131
31.2	Terms . . . . .	134
31.3	Notation Components . . . . .	140
<b>32</b>	<b>STeX-Structural Features Implementation</b>	<b>143</b>
32.1	Imports with modification . . . . .	143
32.2	The feature environment . . . . .	150
32.3	Features . . . . .	151
<b>33</b>	<b>STeX-Statements Implementation</b>	<b>157</b>
33.1	Definitions . . . . .	157
33.2	Assertions . . . . .	160
33.3	Examples . . . . .	162
33.4	Logical Paragraphs . . . . .	164

<b>34 The Implementation</b>	<b>167</b>
34.1 Package Options . . . . .	167
34.2 Proofs . . . . .	167
34.3 Justifications . . . . .	173
<b>35 <math>\text{\TeX}</math>-Others Implementation</b>	<b>175</b>
<b>36 <math>\text{\TeX}</math>-Metatheory Implementation</b>	<b>176</b>
<b>37 Tikzinput Implementation</b>	<b>179</b>
<b>38 document-structure.sty Implementation</b>	<b>181</b>
38.1 The OMDoc Class . . . . .	181
38.2 Class Options . . . . .	181
38.3 Beefing up the <code>document</code> environment . . . . .	182
38.4 Implementation: OMDoc Package . . . . .	182
38.5 Package Options . . . . .	182
38.6 Document Structure . . . . .	184
38.7 Front and Backmatter . . . . .	187
38.8 Global Variables . . . . .	189
<b>39 MiKoSlides – Implementation</b>	<b>190</b>
39.1 Class and Package Options . . . . .	190
39.2 Notes and Slides . . . . .	192
39.3 Header and Footer Lines . . . . .	196
39.4 Frame Images . . . . .	197
39.5 Colors and Highlighting . . . . .	198
39.6 Sectioning . . . . .	199
39.7 Excursions . . . . .	201
<b>40 The Implementation</b>	<b>203</b>
40.1 Package Options . . . . .	203
40.2 Problems and Solutions . . . . .	204
40.3 Multiple Choice Blocks . . . . .	209
40.4 Including Problems . . . . .	210
40.5 Reporting Metadata . . . . .	211
<b>41 Implementation: The hwexam Class</b>	<b>213</b>
41.1 Class Options . . . . .	213
<b>42 Implementation: The hwexam Package</b>	<b>215</b>
42.1 Package Options . . . . .	215
42.2 Assignments . . . . .	216
42.3 Including Assignments . . . . .	219
42.4 Typesetting Exams . . . . .	220
42.5 Leftovers . . . . .	222

**Part I**  
**Manual**

# Chapter 1

## What is sTeX?

Formal systems for mathematics (such as interactive theorem provers) have the potential to significantly increase both the accessibility of published knowledge, as well as the confidence in its veracity, by rendering the precise semantics of statements machine actionable. This allows for a plurality of added-value services, from semantic search up to verification and automated theorem proving. Unfortunately, their usefulness is hidden behind severe barriers to accessibility; primarily related to their surface languages reminiscent of programming languages and very unlike informal standards of presentation.

sTeX minimizes this gap between informal and formal mathematics by integrating formal methods into established and widespread authoring workflows, primarily L<sup>A</sup>T<sub>E</sub>X, via non-intrusive semantic annotations of arbitrary informal document fragments. That way formal knowledge management services become available for informal documents, accessible via an IDE for authors and via generated *active* documents for readers, while remaining fully compatible with existing authoring workflows and publishing systems.

Additionally, an extensible library of reusable document fragments is being developed, that serve as reference targets for global disambiguation, intermediaries for content exchange between systems and other services.

Every component of the system is designed modularly and extensibly, and thus lay the groundwork for a potential full integration of interactive theorem proving systems into established informal document authoring workflows.

The general sTeX workflow combines functionalities provided by several pieces of software:

- The sTeX package to use semantic annotations in L<sup>A</sup>T<sub>E</sub>X documents,
- RuS<sub>TeX</sub> to convert `tex` sources to (semantically enriched) `xhtml`,
- The MMT software, that extracts semantic information from the thus generated `xhtml` and provides semantically informed added value services.



# Chapter 2

## Quickstart

### 2.1 Setup

#### 2.1.1 The sTeX IDE

TODO: VSCode Plugin

#### 2.1.2 Manual Setup

Foregoing on the sTeX IDE, we will need several pieces of software; namely:

- **The sTeX-Package** available [here](#)<sup>1</sup>. Note, that the CTAN repository for L<sup>A</sup>T<sub>E</sub>X packages may contain outdated versions of the sTeX package, so make sure, that your TEXMF system variable is configured such that the packages available in the linked repository are prioritized over potential default packages that come with your T<sub>E</sub>X distribution.

- **The Mmt System** available [here](#)<sup>2</sup>. We recommend following the setup routine documented [here](#).

Following the setup routine (Step 3) will entail designating a MathHub-directory on your local file system, where the MMT system will look for sTeX/MMT content archives.

- To make sure that sTeX too knows where to find its archives, we need to set a global system variable MATHHUB, that points to your local MathHub-directory (see [chapter 4](#)).

- **sTeX Archives** If we only care about L<sup>A</sup>T<sub>E</sub>X and generating pdfs, we do not technically need MMT at all; however, we still need the MATHHUB system variable to be set. Furthermore, MMT can make downloading content archives we might want to use significantly easier, since it makes sure that all dependencies of (often highly interrelated) sTeX archives are cloned as well.

Once set up, we can run `mmt` in a shell and download an archive along with all of its dependencies like this: `lmh install <name-of-repository>`, or a whole *group* of archives; for example, `lmh install smglom` will download all smglom archives.

---

<sup>1</sup>EdNOTE: For now, we require the latex3-branch

<sup>2</sup>EdNOTE: For now, we require the sTeX-branch, requiring manually compiling the MMT sources

- **R<sub>U</sub>S<sub>T</sub>E<sub>X</sub>** The MMT system will also set up R<sub>U</sub>S<sub>T</sub>E<sub>X</sub> for you, which is used to generate (semantically annotated) `xhtml` from `tex` sources. In lieu of using MMT, you can also download and use R<sub>U</sub>S<sub>T</sub>E<sub>X</sub> directly [here](#).

## 2.2 A First s<sub>T</sub>E<sub>X</sub> Document

Having set everything up, we can write a first s<sub>T</sub>E<sub>X</sub> document. As an example, we will use the `smglom/calculus` and `smglom/arithmetics` archives, which should be present in the designated MathHub-folder.

The document we will consider is the following:

```
\documentclass{article}
\usepackage{stex}
\usepackage{xcolor}
\def\compemph#1{\textcolor{blue}{#1}}

\begin{document}
\usemodule[smglom/calculus]{series}
\usemodule[smglom/arithmetics]{realarith}

The \symref{series}{series}  $\sum_{n=1}^{\infty} \frac{1}{2^n}$ 
\realdivide[\frac]{1}{2}
\realpower{2}{n}
\symref{converges}{converges} towards 1$.
\end{document}
```

Compiling this document with `pdflatex` should yield the output

The **series**  $\sum_{n=1}^{\infty} \frac{1}{2^n}$  **converges** towards 1.

Note that the  $\sum$  and  $\infty$ -symbols are highlighted in blue, and the words “series” and “converges” in bold. This signifies that these words and symbols reference s<sub>T</sub>E<sub>X</sub> *symbols* formally declared somewhere; associating their *presentation* in the document with their (formal) definition - i.e. their semantics. The precise way in which they are highlighted (if at all) can of course be customized (see <sup>3</sup>).

### \usemodule

The command `\usemodule[some/archive]{modulename}` finds some module in the appropriate archive – in the first case (`\usemodule[smglom/calculus]{series}`), s<sub>T</sub>E<sub>X</sub> looks for the archive `smglom/calculus` in our local MathHub-directory (see [chapter 4](#)), and in its source-folder for a file `series.tex`. Since no such file exists, and by default the document is assumed to be in *english*, it picks the file `series.en.tex`, and indeed, in here we find a statement `\begin{module}{series}`.

s<sub>T</sub>E<sub>X</sub> now reads this file and makes all semantic macros therein available to use, along with all its dependencies. This enables the usage of `\infinitiesum` later on.

Analogously, `\usemodule[smglom/arithmetics]{realarith}` opens the file `realarith.en.tex` in the `.../smglom/arithmetics/source-folder` and makes its contents available, e.g. `\realdivide` and `\realpower`.

<sup>3</sup>EdNOTE: somewhere later

---

`\symref`  
`\symname`

---

The command `\symref{symbolname}{text}` marks the `text` in the second argument as representing the `symbolname` in the first argument – which is why the word “series” is set in boldface. In the pdf, this is all that happens. In the `xhtml` (which we will investigate shortly) however, we will note that the word “series” is now annotated with the full URI of the symbol denoting the *mathematical concept of a series*. In other words, the word is associated with an unambiguous semantics.

Notably, in both cases above (*series* and *converges*) the text that *references* the symbol and the name of the symbol are identical. Since this occurs quite often, the shorthand `\symname{converges}` would have worked as well, where `\symname{foo-bar}` behaves exactly like `\symref{foo-bar}{foo bar}` - i.e. the text is simply the name of the symbol with “-” replaced by a space.

---

`\importmodule`

---

If you investigated the contents of the imported modules (`realarith` and `series`) more closely, you’ll note that none of them contain a symbol “converges”. Yet, we can use `\symref` to refer to “converges”. That is because the symbol `converges` is found in `smglom/calculus/source/sequenceConvergence.en.tex`, and `series.en.tex` contains the line `\importmodule{sequenceConvergence}`. The `\importmodule`-statement makes the module referenced available to all documents that include the current module. As such, a “current module” has to exist for `\importmodule` to work, which is why the command is only allowed within a `module-environment`.

**TODO** explain `xhtml` conversion, MMT compilation (requires an archive...?).

## Chapter 3

# Using Semantic Macros

TODO

## Chapter 4

# TeX Archives

### 4.1 The Local MathHub-Directory

`\usemodule`, `\importmodule`, `\inputref` etc. allow for including content modularly without having to specify absolute paths, which would differ between users and machines. Instead, TeX uses *archives* that determine the global namespaces for symbols and statements and make it possible for TeX to find content referenced via such URIs.

All TeX archives need to exist in the local MathHub-directory. TeX knows where this folder is via one of three means:

1. If the TeX package is loaded with the option `mathhub=/path/to/mathhub`, then TeX will consider `/path/to/mathhub` as the local MathHub-directory.
2. If the `mathhub` package option is *not* set, but the macro `\mathhub` exists when the TeX-package is loaded, then this macro is assumed to point to the local MathHub-directory; i.e. `\def\mathhub{/path/to/mathhub}\usepackage{stex}` will set the MathHub-directory as `path/to/mathhub`.
3. Otherwise, TeX will attempt to retrieve the system variable `MATHHUB`, assuming it will point to the local MathHub-directory. Since this variant needs setting up only *once* and is machine-specific (rather than defined in tex code), it is compatible with collaborating and sharing tex content, and hence recommended.

### 4.2 The Structure of TeX Archives

An TeX archive `group/name` needs to be stored in the directory `/path/to/mathhub/group/name`; e.g. assuming your local MathHub-directory is set as `/user/foo/MathHub`, then in order for the `smglom/calculus`-archive to be found by the TeX system, it needs to be in `/user/foo/MathHub/smglo/calculus`.

Each such archive needs two subdirectories:

- `/source` – this is where all your tex files go.
- `/META-INF` – a directory containing a single file `MANIFEST.MF`, the content of which we will consider shortly

An additional `lib`-directory is optional, and is where  $\text{\TeX}$  will look for files included via `\libinput`.

Additionally a *group* of archives `group/name` may have an additional archive `group/meta-inf`. If this `meta-inf`-archive has a `/lib`-subdirectory, it too will be searched by `\libinput` from all tex files in any archive in the `group/*-group`.

### 4.3 MANIFEST.MF-Files

The `MANIFEST.MF` in the `META-INF`-directory consists of key-value-pairs, instructing  $\text{\TeX}$  (and associated software) of various properties of an archive. For example, the `MANIFEST.MF` of the `smglom/calculus`-archive looks like this:

```
id: smglom/calculus
source-base: http://mathhub.info/smglob/calculus
narration-base: http://mathhub.info/smglob/calculus
dependencies: smglom/arithmetic,smglom/sets,smglom/topology,
              smglom/mv,smglom/linear-algebra,smglom/algebra
responsible: Michael.Kohlhase@FAU.de
title: Elementary Calculus
teaser: Terminology for the mathematical study of change.
description: desc.html
```

Many of these are in fact ignored by  $\text{\TeX}$ , but some are important:

`id`: The name of the archive, including its group (e.g. `smglom/calculus`),

`source-base` or

`ns`: The namespace from which all symbol and module URIs in this repository are formed, see (TODO),

`narration-base`: The namespace from which all document URIs in this repository are formed, see (TODO),

`url`: The URL that is formed as a basis for *external references*, see (TODO),

`dependencies`: All archives that this archive depends on.  $\text{\TeX}$  ignores this field, but MMT can pick up on them to resolve dependencies, e.g. for `lmh install`.

## Chapter 5

# Creating New Modules and Symbols

TODO

### 5.1 Advanced Structuring Mechanisms

Given modules:

#### Example 1

```
\begin{module}{magma}
\symdef{universe}{\comp{\mathcal U}}
\symdef[args=2,op=\circ]{operation}{\#1 \comp\circ \#2}
\end{module}
\begin{module}{monoid}
\importmodule{magma}
\symdef{unit}{\comp e}
\end{module}
\begin{module}{group}
\importmodule{monoid}
\symdef[args=1]{inverse}{\#1^{\comp{-1}}}
\end{module}
```

Module 5.1.1[magma]

Module 5.1.2[monoid]

Module 5.1.3[group]

We can form a module for *rings* by “cloning” an instance of **group** (for addition) and **monoid** (for multiplication), respectively, and “glueing them together” to ensure they share the same universe:

## Example 2

```
\begin{module}{ring}
\begin{copymodule}{group}{addition}
\renamedcl[name=universe]{universe}{runiverse}
\renamedcl[name=plus]{operation}{rplus}
\renamedcl[name=zero]{unit}{rzero}
\renamedcl[name=uminus]{inverse}{rminus}
\end{copymodule}
\notation[plus,op=+,prec=60]{rplus}{#1 \comp+ #2}
\notation[zero]{rzero}{\comp0}
\notation[uminus,op=-]{rminus}{\comp- #1}
\begin{copymodule}{monoid}{multiplication}
\assign{universe}{runiverse}
\renamedcl[name=times]{operation}{rtimes}
\renamedcl[name=one]{unit}{rone}
\end{copymodule}
\notation[cdot,op=\cdot,prec=50]{rtimes}{#1 \comp\cdot #2}
\notation[one]{rone}{\comp1}

Test: $\rtimes a{\rplus c{\rtimes de}}$
\end{module}
```

Module 5.1.4[ring]  
Test:  $a \cdot (c + d \cdot e)$

TODO: explain donotclone

## Example 3

```
\begin{module}{int}
\symdef{Integers}{\comp{\mathbb Z}}
\symdef[args=2,op=+]{plus}{#1 \comp+ #2}
\symdef[zero]{\comp0}
\symdef[args=1,op=-]{uminus}{\comp-#1}

\begin{interpretmodule}{group}{intisgroup}
\assign{universe}{\Integers}
\assign{operation}{plus!}
\assign{unit}{zero}
\assign{inverse}{uminus!}
\end{interpretmodule}
\end{module}
```

Module 5.1.5[int]

## 5.2 Primitive Symbols (The sTeX Metatheory)



## Chapter 6

# **TeX Statements (Definitions, Theorems, Examples, ...)**

## Chapter 7

# Additional Packages

**7.1** Modular Document Structuring

**7.2** Slides and Course Notes

**7.3** Homework, Problems and Exams

# Chapter 8

## Stuff

### 8.1 Modules

---

`\sTeX`  
`\stex`

---

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

#### 8.1.1 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 4

```
\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 5

```
\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>4</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 6

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

$a * b$  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 7

```
\mult[\comp{Multiplying}]* $a$  $b$  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 8

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

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

<sup>4</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 9

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

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

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

### Example 10

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

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

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

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

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

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

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

## 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 11

```
\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 12

```
\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.

5 6

<sup>5</sup>EDNOTE: what about e.g.  $\int \int \int f \, dx \, dy \, dz$ ?

<sup>6</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 13

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

## 8.1.2 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 9

# 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.

### 9.1 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 R<sub>U</sub>S<sub>T</sub>E<sub>X</sub>) 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 10

## sTEX-MathHub

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

### 10.1 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.
--------------------------------	---

---

#### 10.1.1 Files, Paths, URIs

---

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

---

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

---

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

---

---

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

---

---

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

---

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

---

<code>\c_stex_pwd_seq</code>	Store the current working directory as path-sequence and string, respectively, and the (heuristically guessed) full path to the main file, based on the PWD and <code>\jobname</code> .
<code>\c_stex_pwd_str</code>	
<code>\c_stex_mainfile_seq</code>	
<code>\c_stex_mainfile_str</code>	

---

`\g_stex_currentfile_seq`

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

**Test 1**

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

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

**10.1.2 MathHub Archives**

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

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

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

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

`\l_stex_current_repository_prop`

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

<hr/> <hr/> <code>\stex_set_current_repository:n</code>	Sets the current repository to the one with the provided ID. calls <code>\__stex_mathhub_do_manifest:n</code> , so works whether this repository's MANIFEST.MF-file has already been read or not.
<hr/> <hr/> <code>\stex_require_repository:n</code>	Calls <code>\__stex_mathhub_do_manifest:n</code> iff the corresponding archive property list does not already exist, and adds a corresponding definition to the <code>.sms</code> -file.
<hr/> <hr/> <code>\stex_in_repository:nn</code>	<code>\stex_in_repository:nn{&lt;repository-name&gt;}{&lt;code&gt;}</code> Change the current repository to <code>{&lt;repository-name&gt;}</code> (or not, if <code>{&lt;repository-name&gt;}</code> is empty), and passes its ID on to <code>{&lt;code&gt;}</code> as #1. Switches back to the previous repository after executing <code>{&lt;code&gt;}</code> .
<hr/> <hr/> <code>\mhpath *</code>	<code>\mhpath{&lt;archive-ID&gt;}{&lt;filename&gt;}</code> Expands to the full path of file <code>&lt;filename&gt;</code> in repository <code>&lt;archive-ID&gt;</code> . Does not check whether the file or the repository exist.
<hr/> <hr/> <code>\inputref</code> <hr/> <code>\inputref:nn</code>	<code>\inputref[&lt;archive-ID&gt;]{&lt;filename&gt;}</code> <code>\inputs</code> the file <code>&lt;filename&gt;</code> in repository <code>&lt;archive-ID&gt;</code> .
<hr/> <hr/> <code>\libinput</code>	<code>\libinput{&lt;filename&gt;}</code> Inputs <code>&lt;filename&gt;.tex</code> from the <code>lib</code> folders in the current archive and the <code>meta-inf</code> -archive of the current archive group (if existent). Throws an error if no file by that name exists in either folder, includes both if both exist.

## Test 2

```

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

```

```

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

```

## Chapter 11

# sTeX-References

Code related to links and cross-references

### 11.1 Macros and Environments

# Chapter 12

## sTeX-Modules

Code related to Modules

### 12.1 Macros and Environments

---

`\l_stex_current_module_str`

---

All information of a module is stored as a property list. `\l_stex_current_module_str` 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

```

### 12.1.1 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_str` 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:cn {c_stex_module_\l_stex_current_module_str_prop} { ns }?
\prop_item:cn {c_stex_module_\l_stex_current_module_str_prop} { name }\\
Language:-\prop_item:cn {c_stex_module_\l_stex_current_module_str_prop} { lang }\\
Signature:-\prop_item:cn {c_stex_module_\l_stex_current_module_str_prop} { sig }\\
Metatheory:-\prop_item:cn {c_stex_module_\l_stex_current_module_str_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:cn {c_stex_module_\l_stex_current_module_str _prop} { ns }?
\prop_item:cn {c_stex_module_\l_stex_current_module_str _prop} { name }\\
Language:-\prop_item:cn {c_stex_module_\l_stex_current_module_str _prop} { lang }\\
Signature:-\prop_item:cn {c_stex_module_\l_stex_current_module_str _prop} { sig }\\
Metatheory:-\prop_item:cn {c_stex_module_\l_stex_current_module_str _prop} { meta }\\
\end{module}
\ExplSyntaxOff
```

```
Module 12.1.1[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`.

---

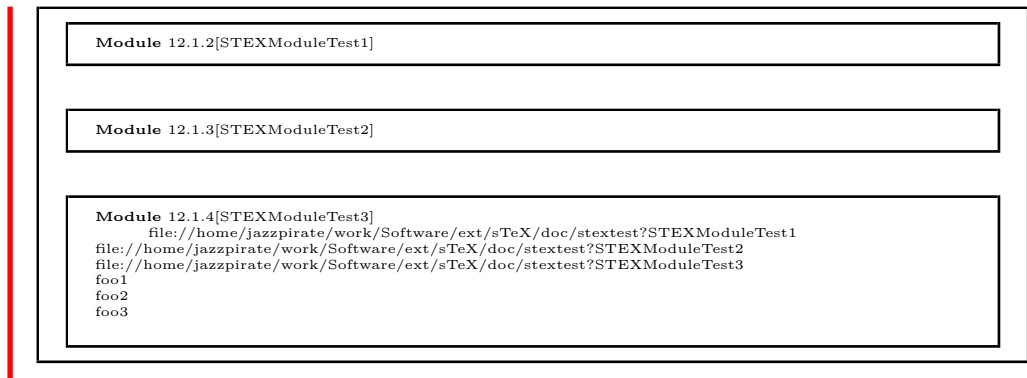
`\stex_invoke_module:n`

---

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 13

# STEX-Module Inheritance

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

### 13.1 Macros and Environments

#### 13.1.1 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
```

## 13.1.2 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 13.1.1[Foo]**  
Meaning: `\macro->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo?foo}<`

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

**Module 13.1.2[Importtest]**  
Meaning: `\macro->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo?foo}<`

**Module 13.1.3[Importtest2]**  
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 13.1.4[UseTest1]

Module 13.1.5[UseTest2]

Meaning: >macro->\stex\_invoke\_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest1?foo}<

Module 13.1.6[UseTest3]

Meaning: >undefined<

Meaning: >macro->\stex\_invoke\_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest2?bar}<

All modules: <http://mathhub.info/sTeX?Metatheory>, <file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest3>,  
 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?sequence-index>, <http://mathhub.info/sTeX?Metatheory?seqtype>,  
<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>,  
<http://mathhub.info/sTeX?Metatheory?dummyvar>,  
<http://mathhub.info/sTeX?Metatheory?fromto>, <http://mathhub.info/sTeX?Metatheory?apply>, <http://mathhub.info/sTeX?Metatheory?collect>,  
<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 13.1.7[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 14

## TeX-Symbols

Code related to symbol declarations and notations

### 14.1 Macros and Environments

---

$\backslash\text{symdecl}$	$\backslash\text{symdecl}[\langle\text{args}\rangle]\{\langle\text{macroname}\rangle\}$
----------------------------	---

---

Declares a new symbol with semantic macro  $\backslash\text{macroname}$ . Optional arguments are:

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

---

---

`\stex_symdecl_do:n`

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

Ultimately stores the symbol  $\langle URI \rangle$  in the property list `\l_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 14.1.1[SymdeclTest]
Meaning: >macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?SymdeclTest?foo}<
Result: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?SymdeclTest?foo
Meaning: >macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?SymdeclTest?bardef}<
```

---

---

`\l_stex_all_symbols_seq`

Stores full URIs for all modules currently in scope.

---

---

`\stex_get_symbol:n`

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

---

---

`\notation`

`\notation[⟨args⟩]{⟨symbol⟩}{⟨notations+⟩}`

Introduces a new notation for  $\langle symbol \rangle$ , see `\stex_notation_do:nn`

---

---

`\stex_notation_do:nn`

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

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

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

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

### Test 12

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

Module 14.1.2[NotationTest]

---

---

`\symdef`

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

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

### Test 13

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

Module 14.1.3[SymdefTest]  
 $a + b + c$

# Chapter 15

## STEX-Terms

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

### 15.1 Macros and Environments

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

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

### Test 14

```

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

```

Module 15.1.1[MathTest1]  
 $\langle a^b_c \rangle$  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 } }
$\foobar{a\{b,c,d,e,f\}g}$ and $\foobar[foo] a\{b,c\}g$ and $\foobar abc$

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

```

Module 15.1.2[MathTest2]  
 $\langle a \mid [b;c;d:e,f]^g \rangle$  and  $\langle a \mid [b;c]^g \rangle$  and  $\langle a \mid [b]^c \rangle$   
 $a + (b \cdot c)$  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 15.1.3[TextTest]
  some a and some b and also some c 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 16

# ST<sub>E</sub>X-Structural Features

Code related to structural features

### 16.1 Macros and Environments

#### 16.1.1 Structures

`mathstructure` TODO

## Chapter 17

# sTeX-Statements

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

### 17.1 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 18

# **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

## 18.1 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>7</sup>

<sup>7</sup>EDNOTE: talk a bit more about proofs and their structure,... maybe copy from OMDoc spec.

## 18.2 The User Interface

### 18.2.1 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).

### 18.2.2 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.

### 18.2.3 Justifications

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

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

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

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

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

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

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

**P.1.1**  $n > 1$ :

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

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

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

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

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

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

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

#### 18.2.4 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>method</b>	
<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>spfcase</b> environments that mark up the cases one by one.
<b>spfcase</b>	The content of a <b>pfcases</b> environment are a sequence of case proofs marked up in the <b>pfcase</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>pfcase</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>spfcase</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> .

### 18.2.5 Proof End Markers

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

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

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

### 18.2.6 Configuration of the Presentation

Finally, we provide configuration hooks in Figure 1 for the keywords in proofs. These are mainly intended for package authors building on `statements`, e.g. for multi-language support.<sup>8</sup> The proof step labels can be customized via the `\pstlabelstyle` macro:

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

Figure 1: Configuration Hooks for Semantic Proof Markup

`\pstlabelstyle`

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

style	example	configuration macro
long	0.8.1.5	<code>\def\pst@make@label@long#1#2{\@for\@I:=#1\do{\@I.}#2}</code>
angles	$\ggg 5$	<code>\def\pst@make@label@angles#1#2{\ensurermath{\@for\@I:=#1\do{\rangle}}#2}</code>
short	5	<code>\def\pst@make@label@short#1#2{#2}</code>
empty		<code>\def\pst@make@label@empty#1#2{}</code>

Figure 2: Configuration Proof Step Label Styles

### 18.3 Limitations

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

---

<sup>8</sup>EDNOTE: we might want to develop an extension `sproof-babel` in the future.

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

## Chapter 19

# 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.

### 19.1 Symbols



**Part III**  
**Extensions**

## Chapter 20

# Tikzinput

### 20.1 Macros and Environments

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

## Chapter 21

# 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.

### 21.1 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

21.2 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`.

21.2.1 Package and Class Options

The `omdoc` class accept the following options:

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

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

21.2.2 Document Structure

document

\documentkeys

id

omgroup

id

creators

contributors

short

loadmodules

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.

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.barderv,loadmodules]{Introducing $\protect\bar$ Derivations}

```

$\text{\TeX}$  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>9</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.

### 21.2.3 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].

#### 21.2.4 Structure Sharing

The `\STRlabel` macro takes two arguments: a label and the content and stores the content for later use by `\STRcopy` [`\URL`]{`label`}, which expands to the previously stored content. If the `\STRlabel` macro was in a different file, then we can give a URL [`URL`] that lets  $\text{\LaTeX}$ 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>10</sup>

#### 21.2.5 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{<vname>}{<text>}` to set the global variable `<vname>` to `<text>` and `\useSGvar{<vname>}` to reference it.

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

<sup>10</sup>EdNOTE: document LMID und LMXRef here if we decide to keep them.

### 21.2.6 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.

## 21.3 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 22

## Slides and Course Notes

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

### 22.1 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`.

### 22.2 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.

#### 22.2.1 Package Options

The `mikoslides` class takes a variety of class options:<sup>11</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 22.2.2).</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 22.2.4). 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>

### 22.2.2 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>11</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.

### 22.2.3 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.

### 22.2.4 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>12</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>12</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}`

## 22.2.5 Colors and Highlighting

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

## 22.2.6 Front Matter, Titles, etc.

### 22.2.7 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}
```

## 22.2.8 Miscellaneous

## 22.3 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 23

# 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.

### 23.1 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.

### 23.2 The User Interface

#### 23.2.1 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

### 23.2.2 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.

### 23.2.3 Multiple Choice Blocks

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

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

See Figure ?? for an example

### 23.2.4 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.

### 23.2.5 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.

## 23.3 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 24

# **`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



## 24.1 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.

## 24.2 The User Interface

### 24.2.1 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].

### 24.2.2 Assignments

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

### 24.2.3 Typesetting Exams

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

`test` Finally, there is the option `test` that modifies the behavior to facilitate formatting tests. Only in `test` mode, the macros `\testspace`, `\testnewpage`, and `\testemptypage` have an effect: they generate space for the students to solve the given problems. Thus they can be left in the L<sup>A</sup>T<sub>E</sub>X source.

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

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

### 24.2.4 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.

## 24.3 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 25

# ST<sub>E</sub>X -Basics Implementation

### 25.1 The ST<sub>E</sub>XDocument 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>
```

### 25.2 Preliminaries

```
15 <*package>
16
17 %%%%%%%%% basics.dtx %%%%%%%%%
18
19 \RequirePackage{expl3,l3keys2e,ltxcmds}
20 \ProvidesExplPackage{stex}{2021/08/01}{1.9}{bla}
21 \RequirePackage{expl-keystr-compatible}
22
23 %\RequirePackage{morewrites}
24 %\RequirePackage{amsmath}
25
```

Package options:

```

26 \keys_define:nn { stex } {
27   debug      .clist_set:N = \c_stex_debug_clist ,
28   showmods   .bool_set:N = \c_stex_showmods_bool ,
29   lang        .clist_set:N = \c_stex_languages_clist ,
30   mathhub     .tl_set_x:N = \mathhub ,
31   sms         .bool_set:N = \c_stex_persist_mode_bool ,
32   image       .bool_set:N = \c_tikzinput_image_bool ,
33   unknown     .code:n      = {}
34 }
35 \ProcessKeysOptions { stex }

```

**\stex** The sTeX logo:

**\sTeX**

```

36 \protected\def\stex{%
37   \ifundefined{texorpdfstring}%
38   {\let\texorpdfstring\@firstoftwo}%
39   }%
40   \texorpdfstring{\raisebox{-.5ex}{S}\kern-.5ex\TeX}{sTeX}\xspace%
41 }
42 \def\sTeX{\stex}

```

(End definition for `\stex` and `\sTeX`. These functions are documented on page 20.)

## 25.3 Messages and logging

```

43 <@@=stex_log>

Warnings and error messages
44 \msg_new:nnn{stex}{error/unknownlanguage}{
45   Unknown~language:~#1
46 }
47 \msg_new:nnn{stex}{warning/nomathhub}{
48   MATHHUB~system~variable~not~found~and~no~
49   \detokenize{\mathhub}~value~set!
50 }
51 \msg_new:nnn{stex}{error/deactivated-macro}{
52   The~\detokenize{#1}~command~is~only~allowed~in~#2!
53 }

```

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

```

54 \cs_new_protected:Nn \stex_debug:nn {
55   \clist_if_in:NnTF \c_stex_debug_clist { all } {
56     \exp_args:Nnnx\msg_set:nnn{stex}{debug / #1}{
57       \\Debug~#1:~#2\\
58     }
59     \msg_none:nn{stex}{debug / #1}
60   }{
61     \clist_if_in:NnT \c_stex_debug_clist { #1 } {
62       \exp_args:Nnnx\msg_set:nnn{stex}{debug / #1}{
63         \\Debug~#1:~#2\\
64       }
65       \msg_none:nn{stex}{debug / #1}
66     }
67   }
68 }

```

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

Redirecting messages:

```

69 \clist_if_in:NnTF \c_stex_debug_clist {all} {
70   \msg_redirect_module:nnn{ stex }{ none }{ term }
71 }{
72   \clist_map_inline:Nn \c_stex_debug_clist {
73     \msg_redirect_name:nnn{ stex }{ debug / ##1 }{ term }
74   }
75 }
76
77 \stex_debug:nn{log}{debug~mode~on}

```

## 25.4 Persistence

78 `<@=stex_persist>`

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

```

79 \iow_new:N \c__stex_persist_sms_iow
80 \AddToHook{begindocument}{
81   \bool_if:NTF \c_stex_persist_mode_bool {
82     \ExplSyntaxOn \input{\jobname.sms} \ExplSyntaxOff
83   } {
84     % \iow_open:Nn \c__stex_persist_sms_iow {\jobname.sms}
85   }
86 }
87 \AddToHook{enddocument}{
88   \bool_if:NF \c_stex_persist_mode_bool {
89     % \iow_close:N \c__stex_persist_sms_iow
90   }
91 }

```

(End definition for `\c__stex_persist_sms_iow`.)

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

```

92 \cs_new_protected:Nn \stex_add_to_sms:n {
93   \bool_if:NF \c_stex_persist_mode_bool {
94     % \iow_now:Nn \c__stex_persist_sms_iow { #1 }
95   }
96 }

```

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

## 25.5 HTML Annotations

97 `<@=stex_annotate>`  
98 `\RequirePackage{rustex}`

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

```

99 \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
100 \ifcsname if@latexml\endcsname\else

```

```

101 \expandafter\newif\csname if@latexml\endcsname\@latexmlfalse
102 \fi
103
104 \prg_new_conditional:Nnn \latexml_if: {p, T, F, TF} {
105   \if@latexml
106     \prg_return_true:
107   \else:
108     \prg_return_false:
109   \fi:
110 }

```

(End definition for `\if@latexml` and `\latexml_if:TF`. These functions are documented on page 20.)

`\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`

```

111 \tl_new:N \l__stex_annotate_arg_tl
112 \tl_const:Nx \c__stex_annotate_emptyarg_tl {
113   \rustex_if:TF {
114     \rustex_direct_HTML:n { \c_ampersand_str lrm; }
115   }{-}
116 }

```

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

`\_stex_annotate_checkempty:n`

```

117 \cs_new_protected:Nn \_stex_annotate_checkempty:n {
118   \tl_set:Nn \l__stex_annotate_arg_tl { #1 }
119   \tl_if_empty:NT \l__stex_annotate_arg_tl {
120     \tl_set_eq:NN \l__stex_annotate_arg_tl \c__stex_annotate_emptyarg_tl
121   }
122 }

```

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

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

```

\stex_if_do_html:
123 \bool_new:N \l_stex_html_do_output_bool
124 \bool_set_true:N \l_stex_html_do_output_bool
125 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
126   \bool_if:nTF \l_stex_html_do_output_bool
127     \prg_return_true: \prg_return_false:
128 }

```

(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

```

129 \cs_new_protected:Nn \stex_suppress_html:n {
130   \exp_args:Nne \use:nn {
131     \bool_set_false:N \l_stex_html_do_output_bool
132     #1
133   }{
134     \stex_if_do_html:T {
135       \bool_set_true:N \l_stex_html_do_output_bool
136     }
137   }
138 }

```



(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.

```

139 \rustex_if:TF{
140   \cs_new_protected:Nn \stex_annotate:nnn {
141     \__stex_annotate_checkempty:n { #3 }
142     \rustex_annotate_HTML:nn {
143       property="stex:#1" ~
144       resource="#2"
145     } {
146       \mode_if_vertical:TF{
147         \tl_use:N \l__stex_annotate_arg_tl\par
148       }{
149         \tl_use:N \l__stex_annotate_arg_tl
150       }
151     }
152   }
153   \cs_new_protected:Nn \stex_annotate_invisible:n {
154     \__stex_annotate_checkempty:n { #1 }
155     \rustex_annotate_HTML:nn {
156       stex:visible="false" ~
157       style:display="none"
158     } {
159       \mode_if_vertical:TF{
160         \tl_use:N \l__stex_annotate_arg_tl\par
161       }{
162         \tl_use:N \l__stex_annotate_arg_tl
163       }
164     }
165   }
166   \cs_new_protected:Nn \stex_annotate_invisible:nnn {
167     \__stex_annotate_checkempty:n { #3 }
168     \rustex_annotate_HTML:nn {
169       property="stex:#1" ~
170       resource="#2" ~
171       stex:visible="false" ~
172       style:display="none"
173     } {
174       \mode_if_vertical:TF{
175         \tl_use:N \l__stex_annotate_arg_tl\par
176       }{
177         \tl_use:N \l__stex_annotate_arg_tl
178       }
179     }
180   }
181   \NewDocumentEnvironment{stex_annotate_env} { m m } {
182     \par
183     \rustex_annotate_HTML_begin:n {
184       property="stex:#1" ~
185       resource="#2"
186   }

```

```

187   }{
188     \par\rustex_annotate_HTML_end:
189   }
190 }{
191   \latexml_if:TF {
192     \cs_new_protected:Nn \stex_annotate:nnn {
193       \__stex_annotate_checkempty:n { #3 }
194       \mode_if_math:TF {
195         \cs:w latexml@annotate@math\cs_end:{#1}{#2}{
196           \tl_use:N \l__stex_annotate_arg_tl
197         }
198       }{
199         \cs:w latexml@annotate@text\cs_end:{#1}{#2}{
200           \tl_use:N \l__stex_annotate_arg_tl
201         }
202       }
203     }
204     \cs_new_protected:Nn \stex_annotate_invisible:n {
205       \__stex_annotate_checkempty:n { #1 }
206       \mode_if_math:TF {
207         \cs:w latexml@invisible@math\cs_end:{
208           \tl_use:N \l__stex_annotate_arg_tl
209         }
210       } {
211         \cs:w latexml@invisible@text\cs_end:{
212           \tl_use:N \l__stex_annotate_arg_tl
213         }
214       }
215     }
216     \cs_new_protected:Nn \stex_annotate_invisible:nnn {
217       \__stex_annotate_checkempty:n { #3 }
218       \cs:w latexml@annotate@invisible\cs_end:{#1}{#2}{
219         \tl_use:N \l__stex_annotate_arg_tl
220       }
221     }
222     \NewDocumentEnvironment{stex_annotate_env} { m m } {
223       \par\begin{latexml@annotateenv}{#1}{#2}
224     }{
225       \par\end{latexml@annotateenv}
226     }
227   }{
228     \cs_new_protected:Nn \stex_annotate:nnn {#3}
229     \cs_new_protected:Nn \stex_annotate_invisible:n {}
230     \cs_new_protected:Nn \stex_annotate_invisible:nnn {}
231     \NewDocumentEnvironment{stex_annotate_env} { m m } {}{}
232   }
233 }

```

(End definition for `\stex_annotate:nnn`, `\stex_annotate_invisible:n`, and `\stex_annotate_invisible:nnn`. These functions are documented on page [21](#).)

## 25.6 Languages

```

234 <@=stex_language>

```

`\c_stex_languages_prop`  
`\c_stex_language_abbrevs_prop`

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

```

235 \prop_const_from_keyval:Nn \c_stex_languages_prop {
236   en = english ,
237   de = ngerman ,
238   ar = arabic ,
239   bg = bulgarian ,
240   ru = russian ,
241   fi = finnish ,
242   ro = romanian ,
243   tr = turkish ,
244   fr = french
245 }
246
247 \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop {
248   english   = en ,
249   ngerman   = de ,
250   arabic    = ar ,
251   bulgarian = bg ,
252   russian   = ru ,
253   finnish   = fi ,
254   romanian  = ro ,
255   turkish   = tr ,
256   french    = fr
257 }
258 % todo: chinese simplified (zhs)
259 %       chinese traditional (zht)

```

(End definition for `\c_stex_languages_prop` and `\c_stex_language_abbrevs_prop`. These variables are documented on page 21.)

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

```

260 \clist_if_empty:NF \c_stex_languages_clist {
261   \clist_clear:N \l_tmpa_clist
262   \clist_map_inline:Nn \c_stex_languages_clist {
263     \prop_get:NnNTF \c_stex_languages_prop { #1 } \l_tmpa_str {
264       \clist_put_right:No \l_tmpa_clist \l_tmpa_str
265     } {
266       \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
267     }
268   }
269   \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
270   \RequirePackage[\clist_use:Nn \l_tmpa_clist,]{babel}
271 }

```

## 25.7 Activating/Deactivating Macros

`\stex_deactivate_macro:Nn`

```

272 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
273   \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
274   \def#1{
275     \msg_error:nnnn{stex}{error/deactivated-macro}{#1}{#2}
276   }
277 }

```

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

`\stex_reactivate_macro:N`

```

278 \cs_new_protected:Nn \stex_reactivate_macro:N {
279   \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname
280 }

```

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

`\stex_do_aftergroup:nn`

```

281 <@@=stex_aftergroup>
282 \tl_new:N \l__stex_aftergroup_tl
283 \cs_new_protected:Nn \stex_do_aftergroup:n {
284   \int_compare:nNnTF \l_stex_module_group_depth_int = \currentgrouplevel {
285     #1
286   }{
287     #1
288     \expandafter \tl_gset:Nn \expandafter \l__stex_aftergroup_tl \expandafter { \l__stex_aft
289     \aftergroup\__stex_aftergroup_do:
290   }
291 }
292 \cs_new_protected:Nn \__stex_aftergroup_do: {
293   \int_compare:nNnTF \l_stex_module_group_depth_int = \currentgrouplevel {
294     \l__stex_aftergroup_tl
295     \tl_clear:N \l__stex_aftergroup_tl
296   }{
297     \l__stex_aftergroup_tl
298     \aftergroup\__stex_aftergroup_do:
299   }
300 }

```

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

```

301 </package>

```

## Chapter 26

# STEX -MathHub Implementation

```
302 <*package>
303
304 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
305
306 <@@=stex_path>
307
308 Warnings and error messages
309 \msg_new:nnn{stex}{error/norepository}{
310   No~archive~#1~found~in~#2
311 }
312 \msg_new:nnn{stex}{error/notinarchive}{
313   Not~currently~in~an~archive,~but~\detokenize{#1}~
314   needs~one!
315 }
316 \msg_new:nnn{stex}{error/nofile}{
317   \detokenize{#1}~could~not~find~file~#2
318 }
```

### 26.1 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
317 \cs_new_protected:Nn \stex_path_from_string:Nn {
318   \str_set:Nx \l_tmpa_str { #2 }
319   \str_if_empty:NTF \l_tmpa_str {
320     \seq_clear:N #1
321   }{
322     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
323     \sys_if_platform_windows:T{
324       \seq_clear:N \l_tmpa_tl
325       \seq_map_inline:Nn #1 {
326         \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
327         \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
328       }
329     }
330   }
```

```

328     }
329     \seq_set_eq:NN #1 \l_tmpa_tl
330   }
331   \stex_path_canonicalize:N #1
332 }
333 }
334 \cs_generate_variant:Nn \stex_path_from_string:Nn
335 { NV, cn, cV }

```

(End definition for `\stex_path_from_string:Nn`. This function is documented on page 22.)

```

\stex_path_to_string:NN
\stex_path_to_string:N
336 \cs_new_protected:Nn \stex_path_to_string:NN {
337   \exp_args:NNe \str_set:Nn #2 { \seq_use:Nn #1 / }
338 }
339
340 \cs_new:Nn \stex_path_to_string:N {
341   \seq_use:Nn #1 /
342 }

```

(End definition for `\stex_path_to_string:NN` and `\stex_path_to_string:N`. These functions are documented on page 22.)

```

\c__stex_path_dot_str . and .., respectively.
\c__stex_path_up_str
343 \str_const:Nn \c__stex_path_dot_str {.}
344 \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.

```

345 \cs_new_protected:Nn \stex_path_canonicalize:N {
346   \seq_if_empty:NF #1 {
347     \seq_clear:N \l_tmpa_seq
348     \seq_get_left:NN #1 \l_tmpa_tl
349     \str_if_empty:NT \l_tmpa_tl {
350       \seq_put_right:Nn \l_tmpa_seq {}
351     }
352     \seq_map_inline:Nn #1 {
353       \str_set:Nn \l_tmpa_tl { ##1 }
354       \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_dot_str {} {
355         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
356           \seq_if_empty:NNTF \l_tmpa_seq {
357             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
358               \c__stex_path_up_str
359             }
360           }{
361             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
362             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
363               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
364                 \c__stex_path_up_str
365               }
366             }{
367               \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
368             }

```

```

369     }
370   }{
371     \str_if_empty:NF \l_tmpa_tl {
372       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
373     }
374   }
375 }
376 }
377 \seq_gset_eq:NN #1 \l_tmpa_seq
378 }
379 }

```

(End definition for `\stex_path_canonicalize:N`. This function is documented on page 22.)

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

```

380 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
381   \seq_if_empty:NTF #1 {
382     \prg_return_false:
383   }{
384     \seq_get_left:NN #1 \l_tmpa_tl
385     \str_if_empty:NTF \l_tmpa_tl {
386       \prg_return_true:
387     }{
388       \prg_return_false:
389     }
390   }
391 }

```

(End definition for `\stex_path_if_absolute:NTF`. This function is documented on page 22.)

## 26.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

392 \str_new:N\l_stex_kpsewhich_return_str
393 \cs_new_protected:Nn \stex_kpsewhich:n {
394   \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
395   \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
396   \tl_trim_spaces:N \l_stex_kpsewhich_return_str
397 }

```

(End definition for `\stex_kpsewhich:n`. This function is documented on page 22.)

We determine the PWD

`\c_stex_pwd_seq`  
`\c_stex_pwd_str`

```

398 \sys_if_platform_windows:TF{
399   \stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
400 }{
401   \stex_kpsewhich:n{-var-value~PWD}
402 }
403
404 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
405 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
406 \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 22.)

## 26.3 File Hooks and Tracking

407 `<@@=stex_files>`

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

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

`\g__stex_files_stack` keeps track of file changes

408 `\seq_gclear_new:N\g__stex_files_stack`

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`

`\c_stex_mainfile_str`

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

410 `\stex_path_from_string:Nn \c_stex_mainfile_seq`

411 `\c_stex_mainfile_str`

(End definition for `\c_stex_mainfile_seq` and `\c_stex_mainfile_str`. These variables are documented on page 22.)

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

412 `\seq_gclear_new:N\g_stex_currentfile_seq`

413 `\AddToHook{file/before}{`

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

415 `\stex_path_if_absolute:NTF\g_stex_currentfile_seq{`

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

417 `}{`

418 `\stex_path_from_string:Nn\g_stex_currentfile_seq{`

419 `\c_stex_pwd_str/\CurrentFilePath/\CurrentFile`

420 `}`

421 `}`

422 `\seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq`

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

424 `}`

425 `\AddToHook{file/after}{`

426 `\seq_if_empty:NF\g__stex_files_stack{`

427 `\seq_gpop:Nn\g__stex_files_stack\l_tmpa_seq`

428 `}`

429 `\seq_if_empty:NTF\g__stex_files_stack{`

430 `\seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq`

431 `}{`

432 `\seq_get:NN\g__stex_files_stack\l_tmpa_seq`

433 `\seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq`

434 `}`

435 `}`

(End definition for `\g_stex_currentfile_seq`. This variable is documented on page 23.)



## 26.4 MathHub Repositories

```

436 <@@=stex_mathhub>

\mathhub
\c_stex_mathhub_seq
\c_stex_mathhub_str
437 \str_if_empty:NTF\mathhub{
438   \stex_kpsewhich:n{-var-value~MATHHUB}
439   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
440
441   \str_if_empty:NTF\c_stex_mathhub_str{
442     \msg_warning:nn{stex}{warning/nomathhub}
443   }{
444     \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
445     \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
446   }
447 }{
448   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
449   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
450     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
451       \c_stex_pwd_str/\mathhub
452     }
453   }
454   \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
455   \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
456 }

```

(End definition for `\mathhub`, `\c_stex_mathhub_seq`, and `\c_stex_mathhub_str`. These variables are documented on page 23.)

```

\__stex_mathhub_do_manifest:n
457 \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
458   \str_set:Nx \l_tmpa_str { #1 }
459   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
460     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
461     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
462     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
463     \__stex_mathhub_find_manifest:N \l_tmpa_seq
464     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
465       \msg_error:nnxx{stex}{error/norepository}{#1}{
466         \stex_path_to_string:N \c_stex_mathhub_str
467       }
468     } {
469       \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
470     }
471   }
472 }

```

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

```

\l__stex_mathhub_manifest_file_seq
473 \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`:

```

474 \cs_new_protected:Nn \_stex_mathhub_find_manifest:N {
475   \seq_set_eq:NN \l_tmpa_seq #1
476   \bool_set_true:N \l_tmpa_bool
477   \bool_while_do:Nn \l_tmpa_bool {
478     \seq_if_empty:NTF \l_tmpa_seq {
479       \bool_set_false:N \l_tmpa_bool
480     }{
481       \file_if_exist:nTF{
482         \stex_path_to_string:N \l_tmpa_seq/MANIFEST.MF
483       }{
484         \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
485         \bool_set_false:N \l_tmpa_bool
486       }{
487         \file_if_exist:nTF{
488           \stex_path_to_string:N \l_tmpa_seq/META-INF/MANIFEST.MF
489         }{
490           \seq_put_right:Nn \l_tmpa_seq{META-INF}
491           \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
492           \bool_set_false:N \l_tmpa_bool
493         }{
494           \file_if_exist:nTF{
495             \stex_path_to_string:N \l_tmpa_seq/meta-inf/MANIFEST.MF
496           }{
497             \seq_put_right:Nn \l_tmpa_seq{meta-inf}
498             \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
499             \bool_set_false:N \l_tmpa_bool
500           }{
501             \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
502           }
503         }
504       }
505     }
506   }
507   \seq_set_eq:NN \l__stex_mathhub_manifest_file_seq \l_tmpa_seq
508 }

```

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

`\c_stex_mathhub_manifest_ior` File variable used for MANIFEST-files

```

509 \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:

```

510 \cs_new_protected:Nn \_stex_mathhub_parse_manifest:n {
511   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
512   \ior_open:Nn \c_stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
513   \ior_map_inline:Nn \c_stex_mathhub_manifest_ior {
514     \str_set:Nn \l_tmpa_str {##1}
515     \exp_args:NNoo \seq_set_split:Nnn
516       \l_tmpb_seq \c_colon_str \l_tmpa_str
517     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {

```

```

518 \exp_args:NNe \str_set:Nn \l_tmpb_tl {
519 \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
520 }
521 \exp_args:No \str_case:nnTF \l_tmpa_tl {
522 {id} {
523 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
524 { id } \l_tmpb_tl
525 }
526 {narration-base} {
527 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
528 { narr } \l_tmpb_tl
529 }
530 {url-base} {
531 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
532 { docurl } \l_tmpb_tl
533 }
534 {source-base} {
535 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
536 { ns } \l_tmpb_tl
537 }
538 {ns} {
539 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
540 { ns } \l_tmpb_tl
541 }
542 {dependencies} {
543 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
544 { deps } \l_tmpb_tl
545 }
546 }{}{}
547 }{}
548 }
549 \ior_close:N \c__stex_mathhub_manifest_ior
550 }

```

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

`\stex_set_current_repository:n`

```

551 \cs_new_protected:Nn \stex_set_current_repository:n {
552 \stex_require_repository:n { #1 }
553 \prop_set_eq:Nc \l_stex_current_repository_prop {
554 c_stex_mathhub_#1_manifest_prop
555 }
556 }

```

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

`\stex_require_repository:n`

```

557 \cs_new_protected:Nn \stex_require_repository:n {
558 \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
559 \stex_debug:nn{mathhub}{Opening~archive:~#1}
560 \__stex_mathhub_do_manifest:n { #1 }
561 \exp_args:Nx \stex_add_to_sms:n {
562 \prop_const_from_keyval:cn { c_stex_mathhub_#1_manifest_prop } {
563 id = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { id } ,
564 ns = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { ns } ,

```

```

565     narr = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { narr } ,
566     deps = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { deps }
567   }
568 }
569 }
570 }

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

571 %\prop_new:N \l_stex_current_repository_prop
572
573 \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
574 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
575   \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
576 } {
577   \__stex_mathhub_parse_manifest:n { main }
578   \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
579   \l_tmpa_str
580   \prop_set_eq:cN { c_stex_mathhub_ \l_tmpa_str _manifest_prop }
581   \c_stex_mathhub_main_manifest_prop
582   \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
583   \stex_debug:nn{mathhub}{Current~repository:~
584   \prop_item:Nn \l_stex_current_repository_prop {id}
585   }
586 }

```

(End definition for `\l_stex_current_repository_prop`. This variable is documented on page 23.)

`\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.

```

587 \cs_new_protected:Nn \stex_in_repository:nn {
588   \str_set:Nx \l_tmpa_str { #1 }
589   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
590   \str_if_empty:NTF \l_tmpa_str {
591     \prop_if_exist:NTF \l_stex_current_repository_prop {
592       \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
593       \exp_args:Ne \l_tmpa_cs{
594         \prop_item:Nn \l_stex_current_repository_prop { id }
595       }
596     }{
597       \l_tmpa_cs{}
598     }
599   }{
600     \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}
601     \stex_require_repository:n \l_tmpa_str
602     \str_set:Nx \l_tmpa_str { #1 }
603     \exp_args:Nne \use:nn {
604       \stex_set_current_repository:n \l_tmpa_str
605       \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
606     }{
607       \stex_debug:nn{mathhub}{switching~back~to:~
608       \prop_if_exist:NTF \l_stex_current_repository_prop {
609         \prop_item:Nn \l_stex_current_repository_prop { id }::~

```

```

610         \meaning\l_stex_current_repository_prop
611     }{
612         no~repository
613     }
614 }
615 \prop_if_exist:NTF \l_stex_current_repository_prop {
616     \stex_set_current_repository:n {
617         \prop_item:Nn \l_stex_current_repository_prop { id }
618     }
619 }{
620     \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
621 }
622 }
623 }
624 }

```

(End definition for `\stex_in_repository:nn`. This function is documented on page [24](#).)

```

\inputref
\stex_inputref:nn
\mhinput\stex_mhinput:nn
625 \newif \ifinputref \inputreffalse
626
627 \cs_new_protected:Nn \stex_mhinput:nn {
628     \stex_in_repository:nn {#1} {
629         \ifinputref
630             \input{ \c_stex_mathhub_str / ##1 / source / #2 }
631         \else
632             \inputreftrue
633             \input{ \c_stex_mathhub_str / ##1 / source / #2 }
634             \inputreffalse
635         \fi
636     }
637 }
638 \NewDocumentCommand \mhinput { 0{} m}{
639     \stex_mhinput:nn{ #1 }{ #2 }
640 }
641
642 \cs_new_protected:Nn \stex_inputref:nn {
643     \stex_in_repository:nn {#1} {
644         \bool_lazy_any:nTF {
645             {\rustex_if_p:} {\latexml_if_p:}
646         } {
647             \str_clear:N \l_tmpa_str
648             \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
649                 \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
650             }
651             \stex_annotate_invisible:nnn{inputref}{
652                 \l_tmpa_str / #2
653             }{}
654         }{
655             \begingroup
656             \inputreftrue
657             \input{ \c_stex_mathhub_str / ##1 / source / #2 }
658             \endgroup
659         }

```

```

660 }
661 }
662
663 \NewDocumentCommand \inputref { 0{} m}{
664   \stex_inputref:nn{ #1 }{ #2 }
665 }
666
667 \cs_new_protected:Nn \stex_mhbibresource:nn {
668   \stex_in_repository:nn {#1} {
669     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
670   }
671 }
672 \newcommand\addmhbibresource[2][]{
673   \stex_mhbibresource:nn{ #1 }{ #2 }
674 }

```

(End definition for `\inputref`, `\stex_inputref:nn`, and `\mhinput\stex_mhinput:nn`. These functions are documented on page 24.)

#### `\mhpath`

```

675 \def \mhpath #1 #2 {
676   \exp_args:Ne \str_if_eq:nnTF{#1}{-}{
677     \c_stex_mathhub_str /
678     \prop_item:Nn \l_stex_current_repository_prop { id }
679     / source / #2
680   }{
681     \c_stex_mathhub_str / #1 / source / #2
682   }
683 }

```

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

#### `\libinput`

```

684 \cs_new_protected:Npn \libinput #1 {
685   \prop_if_exist:NF \l_stex_current_repository_prop {
686     \msg_error:nnn{stex}{error/notinarchive}\libinput
687   }
688   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
689     \msg_error:nnn{stex}{error/notinarchive}\libinput
690   }
691   \bool_set_false:N \l_tmpa_bool
692   \tl_clear:N \l_tmpa_tl
693   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
694   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
695   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str
696   \seq_pop_left:NNT \l_tmpb_seq \l_tmpb_str {
697     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
698     \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
699       / meta-inf / lib / #1.tex}{
700       \bool_set_true:N \l_tmpa_bool
701       \tl_put_right:Nx \l_tmpa_tl {
702         \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
703           / meta-inf / lib / #1.tex}
704       }
705     }{}

```

```

706 }
707 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
708   / \l_tmpa_str / lib / #1.tex
709 }{
710   \bool_set_true:N \l_tmpa_bool
711   \tl_put_right:Nx \l_tmpa_tl {
712     \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
713       / \l_tmpa_str / lib / #1.tex}
714   }
715 }{}
716 \bool_if:NF \l_tmpa_bool {
717   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
718 }
719 \l_tmpa_tl
720 }

```

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

```

721 </package>

```

## Chapter 27

# STEX -References Implementation

```
722 <*package>
723
724 %%%%%%%%%% references.dtx %%%%%%%%%%
725
726 %\RequirePackage{hyperref}
727 %\RequirePackage{cleveref}
728 <@@=stex_refs>
729
730 Warnings and error messages
731
732 \iow_new:N \c__stex_refs_refs_iow
733 \AddToHook{begindocument}{
734   \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
735 }
736 \AddToHook{enddocument}{
737   \iow_close:N \c__stex_refs_refs_iow
738 }
739
740 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
741
742 \NewDocumentCommand \STEXreftitle { m } {
743   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
744 }
745 }
```

### 27.1 Document URIs and URLs

```
743 \seq_new:N \g__stex_refs_all_refs_seq
744
745 \str_new:N \l_stex_current_docns_str
746
747 \cs_new_protected:Nn \stex_get_document_uri: {
748   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
749   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
750   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
751   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
752 }
```



```

752 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
753
754 \str_clear:N \l_tmpa_str
755 \prop_if_exist:NT \l_stex_current_repository_prop {
756   \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
757     \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
758   }
759 }
760
761 \str_if_empty:NTF \l_tmpa_str {
762   \str_set:Nx \l_stex_current_docns_str {
763     file:/\stex_path_to_string:N \l_tmpa_seq
764   }
765 }{
766   \bool_set_true:N \l_tmpa_bool
767   \bool_while_do:Nn \l_tmpa_bool {
768     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
769     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
770       {source} { \bool_set_false:N \l_tmpa_bool }
771     }{}{
772       \seq_if_empty:NT \l_tmpa_seq {
773         \bool_set_false:N \l_tmpa_bool
774       }
775     }
776   }
777
778   \seq_if_empty:NTF \l_tmpa_seq {
779     \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
780   }{
781     \str_set:Nx \l_stex_current_docns_str {
782       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
783     }
784   }
785 }
786 }
787
788 \str_new:N \l_stex_current_docurl_str
789 \cs_new_protected:Nn \stex_get_document_url: {
790   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
791   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
792   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
793   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
794   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
795
796   \str_clear:N \l_tmpa_str
797   \prop_if_exist:NT \l_stex_current_repository_prop {
798     \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
799       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
800         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
801       }
802     }
803
804     \str_if_empty:NTF \l_tmpa_str {
805       \str_set:Nx \l_stex_current_docurl_str {

```

```

806     file:/\stex_path_to_string:N \l_tmpa_seq
807   }
808 }{
809   \bool_set_true:N \l_tmpa_bool
810   \bool_while_do:Nn \l_tmpa_bool {
811     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
812     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
813       {source} { \bool_set_false:N \l_tmpa_bool }
814     }{}{
815       \seq_if_empty:NT \l_tmpa_seq {
816         \bool_set_false:N \l_tmpa_bool
817       }
818     }
819   }
820
821   \seq_if_empty:NTF \l_tmpa_seq {
822     \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
823   }{
824     \str_set:Nx \l_stex_current_docurl_str {
825       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
826     }
827   }
828 }
829 }

```

## 27.2 Setting Reference Targets

```

830 \str_const:Nn \c__stex_refs_url_str{URL}
831 \str_const:Nn \c__stex_refs_ref_str{REF}
832 % @currentlabel -> number
833 % @currentlabelname -> title
834 % @currentHref -> name.number <- id of some kind
835 % \theH# -> \arabic{section}
836 % \the# -> number
837 % \hyper@makecurrent{#}
838 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
839   \stex_get_document_uri:
840   \str_set:Nx \l_tmpa_str { #1 }
841   \str_if_empty:NT \l_tmpa_str {
842     \int_zero:N \l_tmpa_int
843     \bool_set_true:N \l_tmpa_bool
844     \bool_while_do:Nn \l_tmpa_bool {
845       \cs_if_exist:cTF {
846         sref_\l_stex_current_docns_str?? REF_\int_use:N \l_tmpa_int _type
847       }{
848         \int_incr:N \l_tmpa_int
849       }{
850         \str_set:Nx \l_tmpa_str { REF_\int_use:N \l_tmpa_int }
851         \bool_set_false:N \l_tmpa_bool
852       }
853     }
854   }
855   \str_set:Nx \l_tmpa_str {
856     \l_stex_current_docns_str??\l_tmpa_str

```

```

857 }
858 \seq_gput_right:No \g__stex_refs_all_refs_seq \l_tmpa_str
859 \stex_if_smsmode:TF {
860   \stex_get_document_url:
861   \str_gset_eq:cN {sref_url_\l_tmpa_str _str}\l_stex_current_docurl_str
862   \str_gset_eq:cN {sref_\l_tmpa_str _type}\c__stex_refs_url_str
863 }{
864   \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter{\@currentlabel\iffalse}{
865   \exp_args:Nx\label{sref_\l_tmpa_str}
866
867   \exp_args:NNNx\immediate\write\@auxout{\stexauxadddocref{\l_tmpa_str}}
868   \str_gset:cx {sref_\l_tmpa_str _type}\c__stex_refs_ref_str
869 }
870 }
871 \cs_new_protected:Npn \stexauxadddocref #1 {
872   \str_set:Nx \l_tmpa_str {#1}
873   \str_gset_eq:cN{sref_\l_tmpa_str _type}\c__stex_refs_ref_str
874   \seq_gput_right:Nx \g__stex_refs_all_refs_seq {\l_tmpa_str}
875 }
876 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
877   \str_gset_eq:cN {sref_sym_#1_uri} \l_stex_current_docns_str
878 }

```

## 27.3 Using References

```

879 \str_new:N \l__stex_refs_indocument_str
880 \keys_define:nn { stex / sref } {
881   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
882   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
883   pre           .tl_set:N = \l__stex_refs_pre_tl ,
884   post          .tl_set:N = \l__stex_refs_post_tl ,
885   %indoc        .str_set:x:N = \l__stex_refs_repo_str ,
886 }
887
888 \bool_new:N \c__stex_refs_hyperref_bool
889 \bool_set_false:N \c__stex_refs_hyperref_bool
890 \AddToHook{begindocument}{
891   \@ifpackageloaded{hyperref}{
892     \bool_set_true:N \c__stex_refs_hyperref_bool
893   }{}
894 }
895
896
897 \cs_new_protected:Nn \__stex_refs_args:n {
898   \tl_clear:N \l__stex_refs_linktext_tl
899   \tl_clear:N \l__stex_refs_fallback_tl
900   \tl_clear:N \l__stex_refs_pre_tl
901   \tl_clear:N \l__stex_refs_post_tl
902   \str_clear:N \l__stex_refs_repo_str
903   \keys_set:nn { stex / sref } { #1 }
904 }
905
906 \NewDocumentCommand \sref { 0{} m}{
907   \__stex_refs_args:n { #1 }

```

```

908 \str_if_empty:NTF \l__stex_refs_indocument_str {
909   \str_set:Nn \l_tmpa_str { #2 }
910   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
911   \tl_set:Nn \l_tmpa_tl {
912     \l__stex_refs_fallback_tl
913   }
914   \seq_map_inline:Nn \g__stex_refs_all_refs_seq {
915     \str_set:Nn \l_tmpb_str { ##1 }
916     \str_if_eq:eeT { \l_tmpa_str } {
917       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int }{-1 }
918     } {
919       \seq_map_break:n {
920         \tl_set:Nn \l_tmpa_tl {
921           % doc uri in \l_tmpb_str
922           \str_set:Nx \l_tmpa_str {\use:c{sref_\l_tmpb_str _type}}
923           \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
924             % reference
925             \cs_if_exist:cTF{autoref}{
926               \l__stex_refs_pre_tl\autoref{sref_\l_tmpb_str}\l__stex_refs_post_tl
927             }{
928               \l__stex_refs_pre_tl\ref{sref_\l_tmpb_str}\l__stex_refs_post_tl
929             }
930           }{
931             % URL
932             \if_bool:N \c__stex_refs_hyperref_bool {
933               \exp_args:Nx \href{\use:c{sref_url_\l_tmpb_str _str}}{\l__stex_refs_fallback
934             }{
935               \l__stex_refs_fallback_tl
936             }
937           }
938         }
939       }
940     }
941   }
942   \l_tmpa_tl
943 }{
944   % TODO
945 }
946 }
947
948 </package>

```

## Chapter 28

# STEX -Modules Implementation

```
949 <*package>
950
951 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
952
953 <@@=stex_modules>
954
955   Warnings and error messages
956 \msg_new:nnn{stex}{error/unknownmodule}{
957   No~module~#1~found
958 }
959 \msg_new:nnn{stex}{error/syntax}{
960   Syntax~error:~#1
961 }
962 \msg_new:nnn{stex}{error/siglanguage}{
963   Module~#1~declares~signature~#2,~but~does~not~
964   declare~its~language
965 }
966 \msg_new:nnn{stex}{error/concllictingmodules}{
967   Conflicting~imports~for~module~#1
968 }
969
970 \l_stex_current_module_str The current module:
971 \str_new:N \l_stex_current_module_str
972
973 (End definition for \l_stex_current_module_str. This variable is documented on page 26.)
974
975 \l_stex_all_modules_seq Stores all available modules
976 \seq_new:N \l_stex_all_modules_seq
977
978 (End definition for \l_stex_all_modules_seq. This variable is documented on page 26.)
979
980 \stex_if_in_module_p:
981 \stex_if_in_module:TF
982 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
983   \str_if_empty:NTF \l_stex_current_module_str
984     \prg_return_false: \prg_return_true:
985 }
```

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

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

```

974 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
975   \prop_if_exist:cTF { c_stex_module_#1_prop }
976   \prg_return_true: \prg_return_false:
977 }

```

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

`\stex_add_to_current_module:n`  
`\STEXexport`

Only allowed within modules:

```

978 \cs_new_protected:Nn \stex_add_to_current_module:n {
979   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
980 }
981 \cs_new_protected:Npn \STEXexport {
982   \begingroup
983   \newlinechar=-1\relax
984   \endlinechar=-1\relax
985   %\catcode'\ = 9\relax
986   \expandafter\endgroup\STEXexport:n
987 }
988 \cs_new_protected:Nn \STEXexport:n {
989   \ignorespaces #1
990   \stex_add_to_current_module:n { \ignorespaces #1 }
991   \stex_smsmode_set_codes:
992 }
993 \stex_deactivate_macro:Nn \STEXexport {module~environments}

```

(End definition for `\stex_add_to_current_module:n` and `\STEXexport`. These functions are documented on page 27.)

`\stex_add_constant_to_current_module:n`

```

994 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
995   \str_set:Nx \l_tmpa_str { #1 }
996   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
997 }
998
999 %\cs_new_protected:Nn \stex_add_field_to_current_module:n {
1000 % \str_set:Nx \l_tmpa_str { #1 }
1001 % \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _fields} { \l_tmpa_str }
1002 %}

```

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

`\stex_collect_imports:n`

```

1003 \cs_new_protected:Nn \stex_collect_imports:n {
1004   \seq_clear:N \l_stex_collect_imports_seq
1005   \__stex_modules_collect_imports:n {#1}
1006 }
1007 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1008   \seq_map_inline:cn {c_stex_module_#1_imports} {
1009     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1010       \__stex_modules_collect_imports:n { ##1 }
1011     }

```

```

1012 }
1013 \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1014   \seq_put_right:Nn \l_stex_collect_imports_seq { #1 }
1015 }
1016 }

```

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

`\stex_add_import_to_current_module:n`

```

1017 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1018   \str_set:Nx \l_tmpa_str { #1 }
1019   \exp_args:Nno
1020   \seq_if_in:cnF{c_stex_module\l_stex_current_module_str_imports}\l_tmpa_str{
1021     \seq_gput_right:co{c_stex_module\l_stex_current_module_str_imports}\l_tmpa_str
1022   }
1023 }

```

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

`\stex_modules_compute_namespace:nN`

Computes the appropriate namespace from the top-level namespace of a repository (#1) and a file path (#2).

```

1024 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
1025   \str_set:Nx \l_tmpa_str { #1 }
1026   \seq_set_eq:NN \l_tmpa_seq #2
1027   % split off file extension
1028   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1029   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1030   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1031   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1032
1033   \bool_set_true:N \l_tmpa_bool
1034   \bool_while_do:Nn \l_tmpa_bool {
1035     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1036     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1037       {source} { \bool_set_false:N \l_tmpa_bool }
1038     }{}{
1039       \seq_if_empty:NT \l_tmpa_seq {
1040         \bool_set_false:N \l_tmpa_bool
1041       }
1042     }
1043   }
1044
1045   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
1046   \str_if_empty:NTF \l_stex_modules_subpath_str {
1047     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
1048   }{
1049     \str_set:Nx \l_stex_modules_ns_str {
1050       \l_tmpa_str/\l_stex_modules_subpath_str
1051     }
1052   }
1053 }

```

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

Stores its return values in:

```

\l_stex_modules_ns_str
\l_stex_modules_subpath_str
1054 \str_new:N \l_stex_modules_ns_str
1055 \str_new:N \l_stex_modules_subpath_str

(End definition for \l_stex_modules_ns_str and \l_stex_modules_subpath_str. These variables are
documented on page ??.)

```

**\stex\_modules\_current\_namespace:** Computes the current namespace based on the current MathHub repository (if existent) and the current file.

```

1056 \cs_new_protected:Nn \stex_modules_current_namespace: {
1057   \str_clear:N \l_stex_modules_subpath_str
1058   \prop_if_exist:NTF \l_stex_current_repository_prop {
1059     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1060     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1061   }{
1062     % split off file extension
1063     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1064     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1065     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1066     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1067     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1068     \str_set:Nx \l_stex_modules_ns_str {
1069       file:/\stex_path_to_string:N \l_tmpa_seq
1070     }
1071   }
1072 }

```

(End definition for \stex\_modules\_current\_namespace:. This function is documented on page 27.)

## 28.1 The module environment

module arguments:

```

1073 \keys_define:nn { stex / module } {
1074   title      .str_set_x:N = \l_stex_module_title_str ,
1075   ns         .str_set_x:N = \l_stex_module_ns_str ,
1076   lang       .str_set_x:N = \l_stex_module_lang_str ,
1077   sig        .str_set_x:N = \l_stex_module_sig_str ,
1078   creators   .str_set_x:N = \l_stex_module_creators_str ,
1079   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1080   meta       .str_set_x:N = \l_stex_module_meta_str ,
1081   srccite    .str_set_x:N = \l_stex_module_srccite_str
1082 }
1083
1084 \cs_new_protected:Nn \__stex_modules_args:n {
1085   \str_clear:N \l_stex_module_title_str
1086   \str_clear:N \l_stex_module_ns_str
1087   \str_clear:N \l_stex_module_lang_str
1088   \str_clear:N \l_stex_module_sig_str
1089   \str_clear:N \l_stex_module_creators_str
1090   \str_clear:N \l_stex_module_contributors_str
1091   \str_clear:N \l_stex_module_meta_str
1092   \str_clear:N \l_stex_module_srccite_str
1093   \keys_set:nn { stex / module } { #1 }

```



```

1094 }
1095
1096 % module parameters here? In the body?
1097

```

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

```

1098 \cs_new_protected:Nn \stex_module_setup:nn {
1099   \str_set:Nx \l_stex_module_name_str { #2 }
1100   \__stex_modules_args:n { #1 }

```

First, we set up the name and namespace of the module.

Are we in a nested module?

```

1101 \stex_if_in_module:TF {
1102   % Nested module
1103   \prop_get:cnN {c_stex_module\_l_stex_current_module_str _prop}
1104   { ns } \l_stex_module_ns_str
1105   \str_set:Nx \l_stex_module_name_str {
1106     \prop_item:cn {c_stex_module\_l_stex_current_module_str _prop}
1107     { name } / \l_stex_module_name_str
1108   }
1109 }{
1110   % not nested:
1111   \str_if_empty:NT \l_stex_module_ns_str {
1112     \stex_modules_current_namespace:
1113     \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1114     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1115     / {\l_stex_module_ns_str}
1116     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1117     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1118       \str_set:Nx \l_stex_module_ns_str {
1119         \stex_path_to_string:N \l_tmpa_seq
1120       }
1121     }
1122   }
1123 }

```

Next, we determine the language of the module:

```

1124 \str_if_empty:NT \l_stex_module_lang_str {
1125   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1126   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1127   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1128   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1129   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1130     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1131       inferred~from~file~name}
1132     \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1133   }
1134 }
1135
1136 \str_if_empty:NF \l_stex_module_lang_str {
1137   \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1138   \l_tmpa_str {
1139     \ltx@ifpackageloaded{babel}{
1140       \exp_args:Nx \selectlanguage { \l_tmpa_str }

```

```

1141     }{}
1142   } {
1143     \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
1144   }
1145 }

```

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

```

1146 \str_if_empty:NTF \l_stex_module_sig_str {
1147   \exp_args:Nnx \prop_gset_from_keyval:cn {
1148     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1149   } {
1150     name      = \l_stex_module_name_str ,
1151     ns        = \l_stex_module_ns_str ,
1152     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1153     lang      = \l_stex_module_lang_str ,
1154     sig       = \l_stex_module_sig_str ,
1155     meta      = \l_stex_module_meta_str
1156   }
1157   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1158   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _fields}
1159   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1160   \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1161   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1162 \str_if_empty:NT \l_stex_module_meta_str {
1163   \str_set:Nx \l_stex_module_meta_str {
1164     \c_stex_metatheory_ns_str ? Metatheory
1165   }
1166 }
1167 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1168   \bool_set_true:N \l_stex_in_meta_bool
1169   \exp_args:Nx \stex_add_to_current_module:n {
1170     \bool_set_true:N \l_stex_in_meta_bool
1171     \stex_activate_module:n {\l_stex_module_meta_str}
1172     \bool_set_false:N \l_stex_in_meta_bool
1173   }
1174   \stex_activate_module:n {\l_stex_module_meta_str}
1175   \bool_set_false:N \l_stex_in_meta_bool
1176 }
1177 }{
1178   \str_if_empty:NT \l_stex_module_lang_str {
1179     \msg_error:nnxx{stex}{error/siglanguage}{
1180       \l_stex_module_ns_str?\l_stex_module_name_str
1181     }{\l_stex_module_sig_str}
1182   }
1183
1184   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1185   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1186   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1187   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1188   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1189   \str_set:Nx \l_tmpa_str {

```

```

1190     \stex_path_to_string:N \l_tmpa_seq /
1191     \l_tmpa_str . \l_stex_module_sig_str .tex
1192 }
1193 \IfFileExists \l_tmpa_str {
1194     \exp_args:No \stex_in_smsmode:nn { \l_tmpa_str } {
1195         \seq_clear:N \l_stex_all_modules_seq
1196         %\prop_clear:N \l_stex_current_module_prop
1197         \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1198         \input { \l_tmpa_str }
1199     }
1200 }{
1201     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1202 }
1203 \stex_activate_module:n {
1204     \l_stex_module_ns_str ? \l_stex_module_name_str
1205 }
1206 %\prop_set_eq:Nc \l_stex_current_module_prop {
1207 %   c_stex_module_
1208 %   \l_stex_module_ns_str ?
1209 %   \l_stex_module_name_str
1210 %   _prop
1211 %}
1212 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1213 }
1214 }

```

(End definition for `\stex_module_setup:nn`. This function is documented on page 28.)

**module** The module environment.

```

\__stex_modules_begin_module:nn implements \begin{module}

1215 \int_new:N \l_stex_module_group_depth_int
1216 \cs_new_protected:Nn \__stex_modules_begin_module:nn {
1217     \stex_reactivate_macro:N \STEXexport
1218     \stex_reactivate_macro:N \importmodule
1219     \stex_reactivate_macro:N \symdecl
1220     \stex_reactivate_macro:N \notation
1221     \stex_reactivate_macro:N \symdef
1222     \stex_module_setup:nn{#1}{#2}
1223 }
1224 \stex_debug:nn{modules}{
1225     New~module:\\
1226     Namespace:~\l_stex_module_ns_str\\
1227     Name:~\l_stex_module_name_str\\
1228     Language:~\l_stex_module_lang_str\\
1229     Signature:~\l_stex_module_sig_str\\
1230     Metatheory:~\l_stex_module_meta_str\\
1231     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1232 }
1233 }
1234 \seq_put_right:Nx \l_stex_all_modules_seq {
1235     \l_stex_module_ns_str ? \l_stex_module_name_str
1236 }
1237 }

```

```

1238 % \seq_gput_right:Nx \g_stex_modules_in_file_seq
1239 % { \l_stex_module_ns_str ? \l_stex_module_name_str }
1240
1241
1242 \stex_if_smsmode:TF {
1243   \stex_smsmode_set_codes:
1244 } {
1245   \begin{stex_annotate_env} {theory} {
1246     \l_stex_module_ns_str ? \l_stex_module_name_str
1247   }
1248
1249   \stex_annotate_invisible:nnn{header}{} {
1250     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1251     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1252     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1253       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1254     }
1255   }
1256 }
1257 \int_set:Nn \l_stex_module_group_depth_int {\currentgrouplevel}
1258 % TODO: Inherit metatheory for nested modules?
1259 }
1260 \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}

1261 \cs_new_protected:Nn \_stex_modules_end_module: {
1262 % \str_set:Nx \l_tmpa_str {
1263 %   c_stex_module_
1264 %   \prop_item:Nn \l_stex_current_module_prop { ns } ?
1265 %   \prop_item:Nn \l_stex_current_module_prop { name }
1266 %   _prop
1267 % }
1268 %^^A \prop_new:c { \l_tmpa_str }
1269 % \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
1270 \stex_debug:nn{modules}{Closing module~\prop_item:cn {c_stex_module_\l_stex_current_module_
1271 }

```

(End definition for `\_stex_modules_end_module:.`.)

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

```

1272 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1273 \NewDocumentEnvironment { @module } { 0 } { m } {
1274   \par
1275   \_stex_modules_begin_module:nn{#1}{#2}
1276 } {
1277   \_stex_modules_end_module:
1278   \stex_if_smsmode:TF {
1279 % \exp_args:Nx \stex_add_to_sms:n {
1280 %   \prop_gset_from_keyval:cn {
1281 %     c_stex_module_
1282 %     \prop_item:Nn \l_stex_current_module_prop { ns } ?
1283 %     \prop_item:Nn \l_stex_current_module_prop { name }

```

```

1284 %      _prop
1285 %      } {
1286 %      name      = \prop_item:cn { \l_tmpa_str } { name } ,
1287 %      ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
1288 %      file      = \prop_item:cn { \l_tmpa_str } { file } ,
1289 %      lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
1290 %      sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
1291 %      meta      = \prop_item:cn { \l_tmpa_str } { meta }
1292 %      }
1293 %      }
1294 }{
1295   \end{stex_annotate_env}
1296 }
1297 }

```

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

```

1298 \cs_if_exist:NTF \thesection {
1299   \newcounter{module}[section]
1300 }{
1301   \newcounter{module}
1302 }
1303
1304 \bool_if:NT \c_stex_showmods_bool {
1305   \latexml_if:F { \RequirePackage{mdframed} }
1306 }
1307
1308 \cs_new_protected:Nn \stex_modules_heading: {
1309   \stepcounter{module}
1310   \par
1311   \bool_if:NT \c_stex_showmods_bool {
1312     \noindent{\textbf{Module} ~
1313       \cs_if_exist:NT \thesection {\thesection.}
1314       \themodule ~ [\l_stex_module_name_str]
1315     }
1316     \str_if_empty:NTF \l_stex_module_title_str {
1317       }{
1318         \quad(\l_stex_module_title_str)\hfill
1319       }\par
1320     }
1321     \edef\@currentlabel{Module~\thesection.\themodule~[\l_stex_module_name_str]}
1322     % TODO
1323     \stex_ref_new_doc_target:n \l_stex_module_name_str
1324   }

```

(End definition for \stex\_modules\_heading:. This function is documented on page 28.)

Finally:

```

1325 \NewDocumentEnvironment { module } { 0 } { m } {
1326   \bool_if:NT \c_stex_showmods_bool {
1327     \begin{mdframed}
1328   }
1329   \begin{@module}[#1]{#2}
1330     \stex_modules_heading:
1331   }{
1332     \end{@module}

```

```

1333 \bool_if:NT \c_stex_showmods_bool {
1334   \end{mdframed}
1335 }
1336 }

```

## 28.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n
1337 \NewDocumentCommand \STEXModule { m } {
1338   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1339   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1340   \tl_set:Nn \l_tmpa_tl {
1341     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1342   }
1343   \seq_map_inline:Nn \l_stex_all_modules_seq {
1344     \str_set:Nn \l_tmpb_str { ##1 }
1345     \str_if_eq:eeT { \l_tmpa_str } {
1346       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1347     } {
1348       \seq_map_break:n {
1349         \tl_set:Nn \l_tmpa_tl {
1350           \stex_invoke_module:n { ##1 }
1351         }
1352       }
1353     }
1354   }
1355   \l_tmpa_tl
1356 }
1357
1358 \cs_new_protected:Nn \stex_invoke_module:n {
1359   \stex_debug:nn{modules}{Invoking~module~#1}
1360   \peek_charcode_remove:NTF ! {
1361     \__stex_modules_invoke_uri:nN { #1 }
1362   } {
1363     \peek_charcode_remove:NTF ? {
1364       \__stex_modules_invoke_symbol:nn { #1 }
1365     } {
1366       \msg_error:nnx{stex}{error/syntax}{
1367         ?~or~!~expected~after~
1368         \c_backslash_str STEXModule{#1}
1369       }
1370     }
1371   }
1372 }
1373
1374 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1375   \str_set:Nn #2 { #1 }
1376 }
1377
1378 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1379   \stex_invoke_symbol:n{#1?#2}
1380 }

```

(End definition for `\STEXModule` and `\stex_invoke_module:n`. These functions are documented on page [29](#).)

`\stex_activate_module:n`

```

1381 \bool_new:N \l_stex_in_meta_bool
1382 \bool_set_false:N \l_stex_in_meta_bool
1383 \cs_new_protected:Nn \stex_activate_module:n {
1384   \stex_debug:nn{modules}{Activating~module~#1}
1385   \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1386     \msg_error:nnn{stex}{error/conclictingmodules}{ #1 }
1387   }
1388   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1389     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1390     \use:c{ c_stex_module_#1_code }
1391   }
1392 }

```

(End definition for `\stex_activate_module:n`. This function is documented on page [30](#).)

```

1393 </package>

```

## Chapter 29

# sTeX -Module Inheritance Implementation

```
1394 <*package>
1395
1396 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1397
```

### 29.1 SMS Mode

```
1398 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1399 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1400 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1401 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1402
1403 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1404   \makeatletter
1405   \makeatother
1406   \ExplSyntaxOn
1407   \ExplSyntaxOff
1408 }
1409
1410 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1411   \symdef
1412   \importmodule
1413   \notation
1414   \symdecl
1415   \STEXexport
1416 }
1417
1418 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1419   \tl_to_str:n {
1420     module,
1421     @module
```



```

1422 }
1423 }

```

(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 31.)

```

\stex_if_smsmode_p:
\stex_if_smsmode:TF

```

```

1424 \bool_new:N \g__stex_smsmode_bool
1425 \bool_set_false:N \g__stex_smsmode_bool
1426 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1427   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1428 }

```

(End definition for `\stex_if_smsmode:TF`. This function is documented on page 31.)

```

\__stex_smsmode_if_catcodes_p:

```

Checks whether the SMS mode category code scheme is active.

```

\__stex_smsmode_if_catcodes:TF

```

```

1429 \bool_new:N \g__stex_smsmode_catcode_bool
1430 \bool_set_false:N \g__stex_smsmode_catcode_bool
1431 \prg_new_conditional:Nnn \__stex_smsmode_if_catcodes: { p, T, F, TF } {
1432   \bool_if:NTF \g__stex_smsmode_catcode_bool
1433   \prg_return_true: \prg_return_false:
1434 }

```

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

```

\stex_smsmode_set_codes:

```

```

1435 \cs_new_protected:Nn \stex_smsmode_set_codes: {
1436   \stex_if_smsmode:T {
1437     \__stex_smsmode_if_catcodes:F {
1438       \bool_gset_true:N \g__stex_smsmode_catcode_bool
1439       \exp_after:wN \char_gset_active_eq:NN
1440       \c_backslash_str \__stex_smsmode_cs:
1441       \tex_global:D \char_set_catcode_active:N \
1442       \tex_global:D \char_set_catcode_other:N $
1443       \tex_global:D \char_set_catcode_other:N ^
1444       \tex_global:D \char_set_catcode_other:N _
1445       \tex_global:D \char_set_catcode_other:N &
1446       \tex_global:D \char_set_catcode_other:N ##
1447     }
1448   }
1449 } \iffalse $ \fi % to make syntax highlighting work again

```

(End definition for `\stex_smsmode_set_codes:.` This function is documented on page 31.)

```

\__stex_smsmode_unset_codes:

```

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

```

1450 \cs_new_protected:Nn \__stex_smsmode_unset_codes: {
1451   \__stex_smsmode_if_catcodes:T {
1452     \bool_gset_false:N \g__stex_smsmode_catcode_bool
1453     \exp_after:wN \tex_global:D \exp_after:wN
1454     \char_set_catcode_escape:N \c_backslash_str
1455     \tex_global:D \char_set_catcode_math_toggle:N $
1456     \tex_global:D \char_set_catcode_math_superscript:N ^
1457     \tex_global:D \char_set_catcode_math_subscript:N _
1458     \tex_global:D \char_set_catcode_alignment:N &
1459     \tex_global:D \char_set_catcode_parameter:N ##
1460   }
1461 } \iffalse $ \fi % to make syntax highlighting work again

```

(End definition for `\_stex_smsmode_unset_codes:`.)

`\stex_in_smsmode:nn`

```

1462 \cs_new_protected:Nn \stex_in_smsmode:nn {
1463   \vbox_set:Nn \l_tmpa_box {
1464     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1465     \bool_gset_true:N \g__stex_smsmode_bool
1466     \stex_smsmode_set_codes:
1467     #2
1468     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1469     \stex_if_smsmode:F {
1470       \__stex_smsmode_unset_codes:
1471     }
1472   }
1473   \box_clear:N \l_tmpa_box
1474 }

```

(End definition for `\stex_in_smsmode:nn`. This function is documented on page 32.)

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

```

1475 \cs_new_protected:Nn \_stex_smsmode_cs: {
1476   \str_clear:N \l_tmpa_str
1477   \peek_analysis_map_inline:n {
1478     % #1: token (one expansion)
1479     % #2: charcode
1480     % #3 catcode
1481     \token_if_eq_charcode:NNTF ##3 B {
1482       % token is a letter
1483       \exp_args:NNNo \str_put_right:Nn \l_tmpa_str { ##1 }
1484     } {
1485       \str_if_empty:NTF \l_tmpa_str {
1486         % we don't allow (or need) single non-letter CSs
1487         % for now
1488         \peek_analysis_map_break:
1489       }{
1490         \str_if_eq:onTF \l_tmpa_str { begin } {
1491           \peek_analysis_map_break:n {
1492             \exp_after:wN \_stex_smsmode_checkbegin:n ##1
1493           }
1494         } {
1495           \str_if_eq:onTF \l_tmpa_str { end } {
1496             \peek_analysis_map_break:n {
1497               \exp_after:wN \_stex_smsmode_checkend:n ##1
1498             }
1499           } {
1500             \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
1501             \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1502             \g_stex_smsmode_allowedmacros_tl
1503             { \use:c{\l_tmpa_str} } {
1504               \stex_debug:nn{modules}{Executing-1:~\l_tmpa_str}
1505               \peek_analysis_map_break:n {
1506                 \exp_after:wN \l_tmpa_tl ##1
1507               }
1508             }
1509           }
1510         }
1511       }
1512     }
1513   }
1514 }

```

```

1508     } {
1509         \exp_args:NNo \exp_args:NNo \tl_if_in:NnTF
1510         \g_stex_smsmode_allowedmacros_escape_tl
1511         { \use:c{\l_tmpa_str} } {
1512             \__stex_smsmode_unset_codes:
1513             \stex_debug:nn{modules}{Executing~2:~\l_tmpa_str}
1514             % TODO \__stex_smsmode_rescan_cs:
1515             % \int_compare:nNnTF {##2} = {92} {
1516             %     \peek_analysis_map_break:n {
1517             %         \__stex_smsmode_unset_codes:
1518             %         \__stex_smsmode_rescan_cs:
1519             %     }
1520             % } {
1521             %     \peek_analysis_map_break:n {
1522             %         \exp_after:wN \l_tmpa_tl ##1
1523             %     }
1524             % }
1525             } {
1526                 \int_compare:nNnTF {##2} = {92} {
1527                     \peek_analysis_map_break:n { \__stex_smsmode_cs: }
1528                 }{
1529                     \peek_analysis_map_break:n { \exp_after:wN\relax ##1 }
1530                 }
1531             }
1532         }
1533     }
1534 }
1535 }
1536 }
1537 }
1538 }

```

(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:

```

1539 \cs_new_protected:Nn \__stex_smsmode_rescan_cs: {
1540     \str_clear:N \l_tmpb_str
1541     \peek_analysis_map_inline:n {
1542         \token_if_eq_charcode:NNTF ##3 B {
1543             % token is a letter
1544             \exp_args:NNo \str_put_right:Nn \l_tmpb_str { ##1 }
1545         } {
1546             \peek_analysis_map_break:n {
1547                 \exp_after:wN \use:c \exp_after:wN {
1548                     \exp_after:wN \l_tmpa_str\exp_after:wN
1549                 } \use:c { \l_tmpb_str \exp_after:wN } ##1
1550             }
1551         }
1552     }
1553 }

```

(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.

```

1554 \cs_new_protected:Nn \__stex_smsmode_checkbegin:n {
1555   \str_set:Nn \l_tmpa_str { #1 }
1556   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1557     \__stex_smsmode_unset_codes:
1558     \begin{#1}
1559   }
1560 }
```

(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.

```

1561 \cs_new_protected:Nn \__stex_smsmode_checkend:n {
1562   \str_set:Nn \l_tmpa_str { #1 }
1563   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1564     \end{#1}
1565   }
1566 }
```

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

## 29.2 Inheritance

1567 `<@@=stex_importmodule>`

`\stex_import_module_uri:nn`

```

1568 \cs_new_protected:Nn \stex_import_module_uri:nn {
1569   \str_set:Nx \l_stex_import_archive_str { #1 }
1570   \str_set:Nn \l_stex_import_path_str { #2 }
1571
1572   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1573   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1574   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1575
1576   \stex_modules_current_namespace:
1577   \bool_lazy_all:nTF {
1578     {\str_if_empty_p:N \l_stex_import_archive_str}
1579     {\str_if_empty_p:N \l_stex_import_path_str}
1580     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1581   }{
1582     \str_set_eq:NN \l_stex_import_path_str \l_stex_modules_subpath_str
1583     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1584   }{
1585     \str_if_empty:NT \l_stex_import_archive_str {
1586       \prop_if_exist:NT \l_stex_current_repository_prop {
1587         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1588       }
1589     }
1590     \str_if_empty:NTF \l_stex_import_archive_str {
1591       \str_if_empty:NF \l_stex_import_path_str {
1592         \str_set:Nx \l_stex_import_ns_str {
1593           \l_stex_module_ns_str / \l_stex_import_path_str
1594         }
1595       }
1596     }
```

```

1596   }{
1597     \stex_require_repository:n \l_stex_import_archive_str
1598     \prop_get:cnN { c_stex_mathhub\_l_stex_import_archive_str _manifest_prop } { ns }
1599     \l_stex_import_ns_str
1600     \str_if_empty:NF \l_stex_import_path_str {
1601       \str_set:Nx \l_stex_import_ns_str {
1602         \l_stex_import_ns_str / \l_stex_import_path_str
1603       }
1604     }
1605   }
1606 }
1607 }

```

(End definition for `\stex_import_module_uri:nn`. This function is documented on page 34.)

```

\l_stex_import_name_str Store the return values of \stex_import_module_uri:nn.
\l_stex_import_archive_str 1608 \str_new:N \l_stex_import_name_str
\l_stex_import_path_str    1609 \str_new:N \l_stex_import_archive_str
\l_stex_import_ns_str      1610 \str_new:N \l_stex_import_path_str
                           1611 \str_new:N \l_stex_import_ns_str

```

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

```

\stex_import_require_module:nnnnn {<ns>} {<archive-ID>} {<path>} {<name>}
1612 \cs_new_protected:Nn \stex_import_require_module:nnnnn {
1613   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1614
1615     % archive
1616     \str_set:Nx \l_tmpa_str { #2 }
1617     \str_if_empty:NTF \l_tmpa_str {
1618       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1619     } {
1620       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1621       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1622       \seq_put_right:Nn \l_tmpa_seq { source }
1623     }
1624
1625     % path
1626     \str_set:Nx \l_tmpb_str { #3 }
1627     \str_if_empty:NTF \l_tmpb_str {
1628       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1629
1630       \ltx@ifpackageloaded{babel} {
1631         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1632           { \language } \l_tmpb_str {
1633           \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1634         }
1635       } {
1636         \str_clear:N \l_tmpb_str
1637       }
1638
1639       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1640       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1641         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }

```

```

1642     }{
1643       \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1644       \IfFileExists{ \l_tmpa_str.tex }{
1645         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1646       }{
1647         % try english as default
1648         \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1649         \IfFileExists{ \l_tmpa_str.en.tex }{
1650           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1651         }{
1652           \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1653         }
1654       }
1655     }
1656
1657   } {
1658     \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1659     \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1660
1661     \ltx@ifpackageloaded{babel} {
1662       \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1663       { \language } \l_tmpb_str {
1664         \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1665       }
1666     } {
1667       \str_clear:N \l_tmpb_str
1668     }
1669
1670     \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1671
1672     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1673     \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1674       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1675     }{
1676       \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1677       \IfFileExists{ \l_tmpa_str/#4.tex }{
1678         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1679       }{
1680         % try english as default
1681         \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1682         \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1683           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1684         }{
1685           \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1686           \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1687             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1688           }{
1689             \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1690             \IfFileExists{ \l_tmpa_str.tex }{
1691               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1692             }{
1693               % try english as default
1694               \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1695               \IfFileExists{ \l_tmpa_str.en.tex }{

```

```

1696         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1697     }{
1698         \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1699     }
1700 }
1701 }
1702 }
1703 }
1704 }
1705 }
1706
1707 \exp_args:No \stex_in_smsmode:nn { \g__stex_importmodule_file_str } {
1708     \seq_clear:N \l_stex_all_modules_seq
1709     \str_clear:N \l_stex_current_module_str
1710     \str_set:Nx \l_tmpb_str { #2 }
1711     \str_if_empty:NF \l_tmpb_str {
1712         \stex_set_current_repository:n { #2 }
1713     }
1714     \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1715     \input { \g__stex_importmodule_file_str }
1716 }
1717
1718 \stex_if_module_exists:nF { #1 ? #4 } {
1719     \msg_error:nnx{stex}{error/unknownmodule}{
1720         #1?#4~(in~file~\g__stex_importmodule_file_str)
1721     }
1722 }
1723 }
1724 \stex_activate_module:n { #1 ? #4 }
1725 }

```

(End definition for `\stex_import_require_module:nnnn`. This function is documented on page 34.)

## `\importmodule`

```

1726 \NewDocumentCommand \importmodule { 0{} m } {
1727     \stex_import_module_uri:nn { #1 } { #2 }
1728     \stex_debug:nn{modules}{Importing~module:~
1729         \l_stex_import_ns_str ? \l_stex_import_name_str
1730     }
1731     \stex_if_smsmode:F {
1732         \stex_import_require_module:nnnn
1733         { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1734         { \l_stex_import_path_str } { \l_stex_import_name_str }
1735         \stex_annotate_invisible:nnn
1736         {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1737     }
1738     \exp_args:Nx \stex_add_to_current_module:n {
1739         \stex_import_require_module:nnnn
1740         { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1741         { \l_stex_import_path_str } { \l_stex_import_name_str }
1742     }
1743     \exp_args:Nx \stex_add_import_to_current_module:n {
1744         \l_stex_import_ns_str ? \l_stex_import_name_str
1745     }

```

```

1746 \stex_smsmode_set_codes:
1747 }
1748 \stex_deactivate_macro:Nn \importmodule {module-environments}

```

*(End definition for \importmodule. This function is documented on page 32.)*

## **\usemodule**

```

1749 \NewDocumentCommand \usemodule { 0{} m } {
1750 \stex_if_smsmode:F {
1751 \stex_import_module_uri:nn { #1 } { #2 }
1752 \stex_import_require_module:nnnn
1753 { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1754 { \l_stex_import_path_str } { \l_stex_import_name_str }
1755 \stex_annotate_invisible:nnn
1756 {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1757 }
1758 \stex_smsmode_set_codes:
1759 }

```

*(End definition for \usemodule. This function is documented on page 33.)*

```

1760 \endpackage

```



## Chapter 30

# STEX -Symbols Implementation

```
1761 <*package>
1762
1763 %%%%%%%%%%%%% symbols.dtx %%%%%%%%%%%%%
1764
```

Warnings and error messages

```
1765
```

### 30.1 Symbol Declarations

```
1766 <@@=stex_symdecl>
```

`\l_stex_all_symbols_seq` Stores all available symbols

```
1767 \seq_new:N \l_stex_all_symbols_seq
```

*(End definition for \l\_stex\_all\_symbols\_seq. This variable is documented on page 36.)*

`\STEXsymbol`

```
1768 \NewDocumentCommand \STEXsymbol { m } {
1769   \stex_get_symbol:n { #1 }
1770   \exp_args:No
1771   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1772 }
```

*(End definition for \STEXsymbol. This function is documented on page 38.)*

symdecl arguments:

```
1773 \keys_define:nn { stex / symdecl } {
1774   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1775   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1776   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1777   type      .tl_set:N   = \l_stex_symdecl_type_tl ,
1778   align     .str_set:N   = \l_stex_symdecl_align_str , % TODO(?)
1779   gfc       .str_set:N   = \l_stex_symdecl_gfc_str , % TODO(?)
1780   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
1781   def       .tl_set:N   = \l_stex_symdecl_definiens_tl
1782 }
```

```

1783
1784 \bool_new:N \l_stex_symdecl_make_macro_bool
1785
1786 \cs_new_protected:Nn \__stex_symdecl_args:n {
1787   \str_clear:N \l_stex_symdecl_name_str
1788   \str_clear:N \l_stex_symdecl_args_str
1789   \bool_set_false:N \l_stex_symdecl_local_bool
1790   \tl_clear:N \l_stex_symdecl_type_tl
1791   \tl_clear:N \l_stex_symdecl_definiens_tl
1792
1793   \keys_set:nn { stex / symdecl } { #1 }
1794 }

```

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

```

1795
1796 \NewDocumentCommand \symdecl { s O{} m } {
1797   \__stex_symdecl_args:n { #2 }
1798   \IfBooleanTF #1 {
1799     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1800   } {
1801     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1802   }
1803   \stex_symdecl_do:n { #3 }
1804   \stex_smsmode_set_codes:
1805 }
1806 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

(End definition for `\symdecl`. This function is documented on page 35.)

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

```

1807 \cs_new_protected:Nn \stex_symdecl_do:n {
1808   \stex_if_in_module:F {
1809     % TODO throw error? some default namespace?
1810   }
1811
1812   \str_if_empty:NT \l_stex_symdecl_name_str {
1813     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1814   }
1815
1816   \prop_if_exist:cT { l_stex_symdecl_
1817     \l_stex_current_module_str ?
1818     \l_stex_symdecl_name_str
1819     _prop
1820   }{
1821     % TODO throw error (beware of circular dependencies)
1822   }
1823
1824   \prop_clear:N \l_tmpa_prop
1825   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
1826   \seq_clear:N \l_tmpa_seq
1827   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1828   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1829

```

```

1830 \exp_args:No \stex_add_constant_to_current_module:n {
1831   \l_stex_symdecl_name_str
1832 }
1833
1834 % arity/args
1835 \int_zero:N \l_tmpb_int
1836
1837 \bool_set_true:N \l_tmpa_bool
1838 \str_map_inline:Nn \l_stex_symdecl_args_str {
1839   \token_case_meaning:NnF ##1 {
1840     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1841     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1842     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1843     {\tl_to_str:n a} {
1844       \bool_set_false:N \l_tmpa_bool
1845       \int_incr:N \l_tmpb_int
1846     }
1847     {\tl_to_str:n B} {
1848       \bool_set_false:N \l_tmpa_bool
1849       \int_incr:N \l_tmpb_int
1850     }
1851   }{
1852     \msg_set:nnn{stex}{error/wrongargs}{
1853       args~value~in~symbol~declaration~for~
1854       \l_stex_current_module_str ?
1855       \l_stex_symdecl_name_str ~
1856       needs~to~be~
1857       i,~a,~b~or~B,~but~##1~given
1858     }
1859     \msg_error:nn{stex}{error/wrongargs}
1860   }
1861 }
1862 \bool_if:NTF \l_tmpa_bool {
1863   % possibly numeric
1864   \str_if_empty:NTF \l_stex_symdecl_args_str {
1865     \prop_put:Nnn \l_tmpa_prop { args } {}
1866     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1867   }{
1868     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1869     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1870     \str_clear:N \l_tmpa_str
1871     \int_step_inline:nn \l_tmpa_int {
1872       \str_put_right:Nn \l_tmpa_str i
1873     }
1874     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1875   }
1876 } {
1877   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1878   \prop_put:Nnx \l_tmpa_prop { arity }
1879     { \str_count:N \l_stex_symdecl_args_str }
1880 }
1881 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
1882
1883

```

```

1884 % semantic macro
1885
1886 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1887   \exp_args:Nx \stex_do_aftergroup:n {
1888     \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1889       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1890     }}
1891   }
1892
1893   \bool_if:NF \l_stex_symdecl_local_bool {
1894     \exp_args:Nx \stex_add_to_current_module:n {
1895       \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1896         \l_stex_current_module_str ? \l_stex_symdecl_name_str
1897       } }
1898     }
1899   }
1900 }
1901
1902 % add to all symbols
1903
1904 \bool_if:NF \l_stex_symdecl_local_bool {
1905   \exp_args:Nx \stex_add_to_current_module:n {
1906     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1907       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1908     }
1909   }
1910 %   \exp_args:Nx \stex_add_field_to_current_module:n {
1911 %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1912 %   }
1913 }
1914
1915 \stex_debug:nn{symbols}{New~symbol:~
1916   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
1917   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
1918   Args:~\prop_item:Nn \l_tmpa_prop { args }
1919 }
1920
1921 % circular dependencies require this:
1922
1923 \prop_if_exist:cF {
1924   l_stex_symdecl_
1925   \l_stex_current_module_str ? \l_stex_symdecl_name_str
1926   _prop
1927 } {
1928   \prop_set_eq:cN {
1929     l_stex_symdecl_
1930     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1931     _prop
1932   } \l_tmpa_prop
1933 }
1934
1935 \seq_clear:c {
1936   l_stex_symdecl_
1937   \l_stex_current_module_str ? \l_stex_symdecl_name_str

```

```

1938   _notations
1939 }
1940
1941 \bool_if:NF \l_stex_symdecl_local_bool {
1942   \exp_args:Nx
1943   \stex_add_to_current_module:n {
1944     \seq_clear:c {
1945       l_stex_symdecl_
1946       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1947       _notations
1948     }
1949     \prop_set_from_keyval:cn {
1950       l_stex_symdecl_
1951       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1952       _prop
1953     } {
1954       name      = \prop_item:Nn \l_tmpa_prop { name }      ,
1955       module    = \prop_item:Nn \l_tmpa_prop { module }    ,
1956       type      = \prop_item:Nn \l_tmpa_prop { type }      ,
1957       args      = \prop_item:Nn \l_tmpa_prop { args }      ,
1958       arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
1959       assocs    = \prop_item:Nn \l_tmpa_prop { assocs }
1960     }
1961   }
1962 }
1963
1964 \stex_if_smsmode:TF {
1965   \bool_if:NF \l_stex_symdecl_local_bool {
1966     % \exp_args:Nx \stex_add_to_sms:n {
1967     %   \prop_set_from_keyval:cn {
1968     %     l_stex_symdecl_
1969     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1970     %     _prop
1971     %   } {
1972     %     name      = \prop_item:Nn \l_tmpa_prop { name }      ,
1973     %     module    = \prop_item:Nn \l_tmpa_prop { module }    ,
1974     %     local     = \prop_item:Nn \l_tmpa_prop { local }     ,
1975     %     type      = \prop_item:Nn \l_tmpa_prop { type }      ,
1976     %     args      = \prop_item:Nn \l_tmpa_prop { args }      ,
1977     %     arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
1978     %     assocs    = \prop_item:Nn \l_tmpa_prop { assocs }
1979     %   }
1980     %   \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1981     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1982     %   }
1983     % }
1984   }
1985 }{
1986   \exp_args:Nx \stex_do_aftergroup:n {
1987     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1988       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1989     }
1990   }
1991   \stex_if_do_html:T {

```

```

1992 \stex_annotate_invisible:nnn {symdecl} {
1993   \l_stex_current_module_str ? \l_stex_symdecl_name_str
1994 } {
1995   \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}{}}{ $\l_st
1996   \stex_annotate_invisible:nnn{args}{}{
1997     \prop_item:Nn \l_tmpa_prop { args }
1998   }
1999   \stex_annotate_invisible:nnn{macroname}{#1}{}
2000   \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2001     \stex_annotate_invisible:nnn{definiens}{}
2002     { $\l_stex_symdecl_definiens_tl$ }
2003   }
2004 }
2005 }
2006 }
2007 }

```

(End definition for `\stex_symdecl_do:n`. This function is documented on page 36.)

`\stex_get_symbol:n`

```

2008 \str_new:N \l_stex_get_symbol_uri_str
2009
2010 \cs_new_protected:Nn \stex_get_symbol:n {
2011   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2012     \__stex_symdecl_get_symbol_from_cs:n { #1 }
2013   }{
2014     % argument is a string
2015     % is it a command name?
2016     \cs_if_exist:cTF { #1 }{
2017       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2018       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2019       \str_if_empty:NNTF \l_tmpa_str {
2020         \exp_args:Nx \cs_if_eq:NNTF {
2021           \tl_head:N \l_tmpa_tl
2022         } \stex_invoke_symbol:n {
2023           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
2024         }{
2025           \__stex_symdecl_get_symbol_from_string:n { #1 }
2026         }
2027       } {
2028         \__stex_symdecl_get_symbol_from_string:n { #1 }
2029       }
2030     }{
2031       % argument is not a command name
2032       \__stex_symdecl_get_symbol_from_string:n { #1 }
2033       % \l_stex_all_symbols_seq
2034     }
2035   }
2036 }
2037
2038 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2039   \str_set:Nn \l_tmpa_str { #1 }
2040   \bool_set_false:N \l_tmpa_bool
2041   \stex_if_in_module:T {

```

```

2042 \exp_args:Nno \seq_if_in:cnT {c_stex_module_\l_stex_current_module_str _constants} { \l_
2043 \bool_set_true:N \l_tmpa_bool
2044 \str_set:Nx \l_stex_get_symbol_uri_str {
2045 \l_stex_current_module_str ? #1
2046 }
2047 }
2048 }
2049 \bool_if:NF \l_tmpa_bool {
2050 \tl_set:Nn \l_tmpa_tl {
2051 \msg_set:nnn{stex}{error/unknownsymbol}{
2052 No~symbol~#1~found!
2053 }
2054 \msg_error:nn{stex}{error/unknownsymbol}
2055 }
2056 \str_set:Nn \l_tmpa_str { #1 }
2057 \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2058 \seq_map_inline:Nn \l_stex_all_symbols_seq {
2059 \str_set:Nn \l_tmpb_str { ##1 }
2060 \str_if_eq:eeT { \l_tmpa_str } {
2061 \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
2062 } {
2063 \seq_map_break:n {
2064 \tl_set:Nn \l_tmpa_tl {
2065 \str_set:Nn \l_stex_get_symbol_uri_str {
2066 ##1
2067 }
2068 }
2069 }
2070 }
2071 }
2072 \l_tmpa_tl
2073 }
2074 }
2075
2076 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
2077 \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2078 { \tl_tail:N \l_tmpa_tl }
2079 \tl_if_single:NTF \l_tmpa_tl {
2080 \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2081 \exp_after:wN \str_set:Nn \exp_after:wN
2082 \l_stex_get_symbol_uri_str \l_tmpa_tl
2083 }{
2084 % TODO
2085 % tail is not a single group
2086 }
2087 }{
2088 % TODO
2089 % tail is not a single group
2090 }
2091 }

```

(End definition for `\stex_get_symbol:n`. This function is documented on page 36.)

## 30.2 Notations

```

2092 <@@=stex_notation>

      notation arguments:
2093 \keys_define:nn { stex / notation } {
2094   lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2095   variant   .tl_set_x:N = \l__stex_notation_variant_str ,
2096   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2097   op        .tl_set:N   = \l__stex_notation_op_tl ,
2098   primary   .bool_set:N = \l__stex_notation_primary_bool ,
2099   primary   .default:n  = {true} ,
2100   unknown   .code:n     = \str_set:Nx
2101             \l__stex_notation_variant_str \l_keys_key_str
2102 }
2103
2104 \cs_new_protected:Nn \stex_notation_args:n {
2105   \str_clear:N \l__stex_notation_lang_str
2106   \str_clear:N \l__stex_notation_variant_str
2107   \str_clear:N \l__stex_notation_prec_str
2108   \tl_clear:N \l__stex_notation_op_tl
2109   \bool_set_false:N \l__stex_notation_primary_bool
2110
2111   \keys_set:nn { stex / notation } { #1 }
2112 }

```

**\notation**

```

2113 \NewDocumentCommand \notation { 0{ } m } {
2114   \stex_notation_args:n { #1 }
2115   \tl_clear:N \l_stex_symdecl_definiens_tl
2116   \stex_get_symbol:n { #2 }
2117   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
2118 }
2119 \stex_deactivate_macro:Nn \notation {module-environments}

```

(End definition for \notation. This function is documented on page 36.)

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

```

2120 \cs_new_protected:Nn \stex_notation_do:nn {
2121   \let\l_stex_current_symbol_str\relax
2122   \prop_set_eq:Nc \l_tmpa_prop {
2123     l_stex_symdecl_ #1 _prop
2124   }
2125
2126   \prop_clear:N \l_tmpb_prop
2127   \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
2128   \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
2129   \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
2130
2131   % precedences
2132   \seq_clear:N \l_tmpb_seq
2133   \exp_args:NNno
2134   \str_if_empty:NTF \l__stex_notation_prec_str {
2135     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2136     \int_compare:nNnTF \l_tmpa_str = 0 {

```



```

2137     \exp_args:NNnx
2138     \prop_put:Nno \l_tmpb_prop { opprec }
2139     { \neginfprec }
2140   }{
2141     \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
2142   }
2143 } {
2144   \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2145     \exp_args:NNnx
2146     \prop_put:Nno \l_tmpb_prop { opprec }
2147     { \neginfprec }
2148     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2149     \int_step_inline:nn { \l_tmpa_str } {
2150       \exp_args:NNx
2151       \seq_put_right:Nn \l_tmpb_seq { \infprec }
2152     }
2153   }{
2154     \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2155     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2156       \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
2157       \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2158         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2159         \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2160         \seq_map_inline:Nn \l_tmpa_seq {
2161           \seq_put_right:Nn \l_tmpb_seq { ##1 }
2162         }
2163       }
2164       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2165     }{
2166       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2167       \int_compare:nNnTF \l_tmpa_str = 0 {
2168         \exp_args:NNnx
2169         \prop_put:Nno \l_tmpb_prop { opprec }
2170         { \infprec }
2171       }{
2172         \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
2173       }
2174     }
2175   }
2176 }
2177
2178 \seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
2179 \int_step_inline:nn { \l_tmpa_str } {
2180   \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
2181     \exp_args:NNx
2182     \seq_put_right:Nn \l_tmpb_seq {
2183       \prop_item:Nn \l_tmpb_prop { opprec }
2184     }
2185   }
2186 }
2187
2188 \prop_put:Nno \l_tmpb_prop { argprecs } \l_tmpb_seq
2189 \tl_clear:N \l_tmpa_tl
2190

```

```

2191 \int_compare:nNnTF \l_tmpa_str = 0 {
2192   \exp_args:NNe
2193   \cs_set:Npn \l__stex_notation_macrocode_cs {
2194     \_stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2195     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2196     { \prop_item:Nn \l_tmpb_prop { opprec } }
2197     { \exp_not:n { #2 } }
2198   }
2199   \__stex_notation_final:
2200 }{
2201   \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
2202   \str_if_in:NnTF \l_tmpb_str b {
2203     \exp_args:Nne \use:nn
2204     {
2205       \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2206       \cs_set:Npn \l_tmpa_str } { {
2207         \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2208         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2209         { \prop_item:Nn \l_tmpb_prop { opprec } }
2210         { \exp_not:n { #2 } }
2211       }}
2212   }{
2213     \str_if_in:NnTF \l_tmpb_str B {
2214       \exp_args:Nne \use:nn
2215       {
2216         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2217         \cs_set:Npn \l_tmpa_str } { {
2218           \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2219           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2220           { \prop_item:Nn \l_tmpb_prop { opprec } }
2221           { \exp_not:n { #2 } }
2222         } }
2223     }{
2224       \exp_args:Nne \use:nn
2225       {
2226         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2227         \cs_set:Npn \l_tmpa_str } { {
2228           \_stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2229           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2230           { \prop_item:Nn \l_tmpb_prop { opprec } }
2231           { \exp_not:n { #2 } }
2232         } }
2233     }
2234   }
2235
2236   \int_zero:N \l_tmpa_int
2237   \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2238   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2239   \__stex_notation_arguments:
2240 }
2241 }

```

(End definition for `\stex_notation_do:nn`. This function is documented on page 37.)

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

```

2242 \cs_new_protected:Nn \_stex_notation_arguments: {
2243   \int_incr:N \l_tmpa_int
2244   \str_if_empty:NTF \l_tmpa_str {
2245     \_stex_notation_final:
2246   }{
2247     \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2248     \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2249     \str_if_eq:VnTF \l_tmpb_str a {
2250       \_stex_notation_argument_assoc:n
2251     }{
2252       \str_if_eq:VnTF \l_tmpb_str B {
2253         \_stex_notation_argument_assoc:n
2254       }{
2255         \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2256         \tl_put_right:Nx \l_tmpa_tl {
2257           { \_stex_term_math_arg:nnn
2258             { \int_use:N \l_tmpa_int }
2259             { \l_tmpb_str }
2260             { ####\int_use:N \l_tmpa_int }
2261           }
2262         }
2263         \_stex_notation_arguments:
2264       }
2265     }
2266   }
2267 }
```

(End definition for `\_stex_notation_arguments:.`)

`\_stex_notation_argument_assoc:n`

```

2268 \cs_new_protected:Nn \_stex_notation_argument_assoc:n {
2269   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2270   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
2271   \tl_put_right:Nx \l_tmpa_tl {
2272     { \_stex_term_math_assoc_arg:nnnn
2273       { \int_use:N \l_tmpa_int }
2274       { \l_tmpb_str }
2275       \exp_args:No \exp_not:n
2276       {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
2277       { ####\int_use:N \l_tmpa_int }
2278     }
2279   }
2280   \_stex_notation_arguments:
2281 }
```

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

`\_stex_notation_final:` Called after processing all notation arguments

```

2282 \cs_new_protected:Nn \_stex_notation_final: {
2283   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
2284   \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
2285   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2286   \exp_args:Nne \use:nn
```

```

2287 {
2288 \cs_generate_from_arg_count:cNnn {
2289   stex_notation_ \l_tmpa_str \c_hash_str
2290   \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2291   _cs
2292 }
2293 \cs_set:Npn \l_tmpb_str } { {
2294   \exp_after:wN \exp_after:wN \exp_after:wN
2295   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2296   { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2297 } }
2298
2299 \tl_if_empty:NF \l__stex_notation_op_tl {
2300   \cs_set:cpx {
2301     stex_op_notation_ \l_tmpa_str \c_hash_str
2302     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2303     _cs
2304   } {
2305     \_stex_term_oms:nnn {
2306       \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2307       \l__stex_notation_lang_str
2308     }{
2309       \l_tmpa_str
2310     }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2311   }
2312 }
2313
2314 \exp_args:Ne
2315 \stex_add_to_current_module:n {
2316   \cs_generate_from_arg_count:cNnn {
2317     stex_notation_ \l_tmpa_str \c_hash_str
2318     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2319     _cs
2320   } \cs_set:Npn {\l_tmpb_str} {
2321     \exp_after:wN \exp_after:wN \exp_after:wN
2322     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2323     { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2324   }
2325   \tl_if_empty:NF \l__stex_notation_op_tl {
2326     \cs_set:cpn {
2327       stex_op_notation_ \l_tmpa_str \c_hash_str
2328       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2329       _cs
2330     } {
2331       \_stex_term_oms:nnn {
2332         \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2333         \l__stex_notation_lang_str
2334       }{
2335         \l_tmpa_str
2336       }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2337     }
2338   }
2339 }
2340

```

```

2341 \seq_put_right:cx {
2342   l_stex_symdecl_
2343   \prop_item:Nn \l_tmpb_prop { symbol }
2344   _notations
2345 } {
2346   \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2347 }
2348
2349 \stex_debug:nn{symbols}{
2350   Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2351   ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
2352   Operator~precedence:~
2353   \prop_item:Nn \l_tmpb_prop { opprec }^^J
2354   Argument~precedences:~
2355   \seq_use:Nn \l_tmpa_seq {,~}^^J
2356   Notation: \cs_meaning:c {
2357     stex_notation_ \l_tmpa_str \c_hash_str
2358     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2359     _cs
2360   }
2361 }
2362
2363 \prop_set_eq:cN {
2364   l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2365   \c_hash_str \l__stex_notation_lang_str _prop
2366 } \l_tmpb_prop
2367
2368 \exp_args:Ne
2369 \stex_add_to_current_module:n {
2370   \seq_put_right:cn {
2371     l_stex_symdecl_
2372     \prop_item:Nn \l_tmpb_prop { symbol }
2373     _notations
2374   } {
2375     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2376   }
2377   \prop_set_from_keyval:cn {
2378     l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2379     \c_hash_str \l__stex_notation_lang_str _prop
2380   } {
2381     symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2382     language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2383     variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,
2384     opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2385     argprecs    = \prop_item:Nn \l_tmpb_prop { argprecs }    ,
2386   }
2387 }
2388
2389 \stex_if_smsmode:TF {
2390   \stex_smsmode_set_codes:
2391   % \exp_args:Nx \stex_add_to_sms:n {
2392   %   \prop_set_from_keyval:cn {
2393   %     l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2394   %     \c_hash_str \l__stex_notation_lang_str _prop

```

```

2395 %      } {
2396 %          symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2397 %          language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2398 %          variant      = \prop_item:Nn \l_tmpb_prop { variant }      ,
2399 %          opprec       = \prop_item:Nn \l_tmpb_prop { opprec }       ,
2400 %          argprec      = \prop_item:Nn \l_tmpb_prop { argprec }      ,
2401 %      }
2402 %  }
2403 }{
2404
2405 % HTML annotations
2406 \stex_if_do_html:T {
2407     \stex_annotate_invisible:nnn { notation }
2408     { \prop_item:Nn \l_tmpb_prop { symbol } } {
2409         \stex_annotate_invisible:nnn { notationfragment }
2410         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
2411         \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2412         \stex_annotate_invisible:nnn { precedence }
2413         { \prop_item:Nn \l_tmpb_prop { opprec };
2414           \seq_use:Nn \l_tmpa_seq { x }
2415         }{}
2416
2417         \int_zero:N \l_tmpa_int
2418         \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2419         \tl_clear:N \l_tmpa_tl
2420         \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
2421             \int_incr:N \l_tmpa_int
2422             \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2423             \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2424             \str_if_eq:VnTF \l_tmpb_str a {
2425                 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2426                     \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2427                     \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2428                 } }
2429             }{
2430                 \str_if_eq:VnTF \l_tmpb_str B {
2431                     \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2432                         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2433                         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2434                     } }
2435                 }{
2436                     \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2437                         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2438                     } }
2439                 }
2440             }
2441         }
2442         \stex_annotate_invisible:nnn { notationcomp }{}{
2443             \str_set:Nx \l_stex_current_symbol_str {\prop_item:Nn \l_tmpb_prop { symbol }}
2444             $ \exp_args:Nno \use:nn { \use:c {
2445                 stex_notation_ \l_stex_current_symbol_str
2446                 \c_hash_str \l__stex_notation_variant_str
2447                 \c_hash_str \l__stex_notation_lang_str _cs
2448             } } { \l_tmpa_tl } $

```

```

2449     }
2450   }
2451 }
2452 }
2453 }

```

(End definition for `\_stex_notation_final:`.)

`\setnotation`

```

2454 \keys_define:nn { stex / setnotation } {
2455   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2456   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2457   unknown .code:n = \str_set:Nx
2458     \l__stex_notation_variant_str \l_keys_key_str
2459 }
2460
2461 \cs_new_protected:Nn \_stex_setnotation_args:n {
2462   \str_clear:N \l__stex_notation_lang_str
2463   \str_clear:N \l__stex_notation_variant_str
2464   \keys_set:nn { stex / setnotation } { #1 }
2465 }
2466
2467 \NewDocumentCommand \setnotation {m m} {
2468   \stex_get_symbol:n { #1 }
2469   \_stex_setnotation_args:n { #2 }
2470   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl\_l_stex_get_symbol_uri_str _notations }
2471     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2472     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notation
2473       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2474     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notation
2475       { \c_hash_str }
2476     \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notations
2477       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2478     \exp_args:Nx \stex_add_to_current_module:n {
2479       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notati
2480         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2481     \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notation
2482       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2483     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notati
2484       { \c_hash_str }
2485   }
2486   \stex_debug:nn {notations}{
2487     Setting~default~notation~
2488     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }~for~
2489     \l_stex_get_symbol_uri_str \
2490     \expandafter\meaning\csname
2491     l_stex_symdecl\_l_stex_get_symbol_uri_str _notations\endcsname
2492   }
2493   }{
2494     % todo throw error
2495   }
2496 }
2497

```

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

**\symdef**

```
2498 \keys_define:nn { stex / symdef } {
2499   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2500   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2501   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2502   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
2503   def       .tl_set:N    = \l_stex_symdecl_definiens_tl ,
2504   op        .tl_set:N    = \l__stex_notation_op_tl ,
2505   lang      .str_set_x:N = \l__stex_notation_lang_str ,
2506   variant   .str_set_x:N = \l__stex_notation_variant_str ,
2507   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2508   unknown   .code:n      = \str_set:Nx
2509             \l__stex_notation_variant_str \l_keys_key_str
2510 }
2511
2512 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2513   \str_clear:N \l_stex_symdecl_name_str
2514   \str_clear:N \l_stex_symdecl_args_str
2515   \bool_set_false:N \l_stex_symdecl_local_bool
2516   \tl_clear:N \l_stex_symdecl_type_tl
2517   \tl_clear:N \l_stex_symdecl_definiens_tl
2518   \str_clear:N \l__stex_notation_lang_str
2519   \str_clear:N \l__stex_notation_variant_str
2520   \str_clear:N \l__stex_notation_prec_str
2521   \tl_clear:N \l__stex_notation_op_tl
2522
2523   \keys_set:nn { stex / symdef } { #1 }
2524 }
2525
2526 \NewDocumentCommand \symdef { 0{} m } {
2527   \__stex_notation_symdef_args:n { #1 }
2528   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2529   \stex_symdecl_do:n { #2 }
2530   \exp_args:Nx \stex_notation_do:nn {
2531     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2532   }
2533 }
2534 \stex_deactivate_macro:Nn \symdef {module~environments}
2535
2536 (End definition for \symdef. This function is documented on page 37.)
2537 \endpackage
```



## Chapter 31

# STEX -Terms Implementation

```
2536 <*package>
2537
2538 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2539
2540 <@@=stex_terms>
2541
2542 Warnings and error messages
2543 \msg_new:nnn{stex}{error/nonotation}{
2544   Symbol~#1~invoked,~but~has~no~notation~#2!
2545 }
2546 \msg_new:nnn{stex}{error/notationarg}{
2547   Error~in~parsing~notation~#1
2548 }
2549 \msg_new:nnn{stex}{error/noop}{
2550   Symbol~#1~has~no~operator~notation~for~notation~#2
2551 }
```

### 31.1 Symbol Invocations

Arguments:

```
2551 \keys_define:nn { stex / terms } {
2552   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2553   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2554   unknown .code:n = \str_set:Nx
2555     \l__stex_terms_variant_str \l_keys_key_str
2556 }
2557
2558 \cs_new_protected:Nn \__stex_terms_args:n {
2559   \str_clear:N \l__stex_terms_lang_str
2560   \str_clear:N \l__stex_terms_variant_str
2561   \str_clear:N \l__stex_terms_prec_str
2562   \tl_clear:N \l__stex_terms_op_tl
2563 }
2564 \keys_set:nn { stex / terms } { #1 }
```

2565 }

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

```
2566 \cs_new_protected:Nn \stex_invoke_symbol:n {
2567   \if_mode_math:
2568     \exp_after:wN \__stex_terms_invoke_math:n
2569   \else:
2570     \exp_after:wN \__stex_terms_invoke_text:n
2571   \fi: { #1 }
2572 }
```

(End definition for \stex\_invoke\_symbol:n. This function is documented on page 38.)

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

```
2573 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2574   \peek_charcode_remove:NTF ! {
2575     \peek_charcode:NTF [ {
2576       \__stex_terms_invoke_op:nw { #1 }
2577     }{
2578       \peek_charcode_remove:NTF ! {
2579         \peek_charcode:NTF [ {
2580           \__stex_terms_invoke_op_custom:nw
2581         }{
2582           % TODO throw error
2583         }
2584       }{
2585         \__stex_terms_invoke_op:nw { #1 } []
2586       }
2587     }
2588   }{
2589     \peek_charcode_remove:NTF * {
2590       \__stex_terms_invoke_text:n { #1 }
2591     }{
2592       \peek_charcode:NTF [ {
2593         \__stex_terms_invoke_math:nw { #1 }
2594       }{
2595         \__stex_terms_invoke_math:nw { #1 } []
2596       }
2597     }
2598   }
2599 }
```

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

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

```
2600 \cs_new_protected:Npn \__stex_terms_invoke_op_custom:nw #1 [#2] {
2601   \stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
2602     \stex_highlight_term:nn{#1}{#2}
2603   }
2604 }
```

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

\\_stex\_terms\_invoke\_op:nw

```

2605 \cs_new_protected:Npn \_stex_terms_invoke_op:nw #1 [#2] {
2606   \_stex_terms_args:n { #2 }
2607   \cs_if_exist:cTF {
2608     stex_op_notation_ #1 \c_hash_str
2609     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2610   }{
2611     \csname stex_op_notation_ #1 \c_hash_str
2612       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2613     \endcsname
2614   }{
2615     \msg_error:nnxx{stex}{error/noop}{#1}{\l__stex_terms_variant_str \c_hash_str \l__stex_te
2616   }
2617 }

```

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

\\_stex\_terms\_invoke\_math:nw

```

2618 \cs_new_protected:Npn \_stex_terms_invoke_math:nw #1 [#2] {
2619   \_stex_terms_args:n { #2 }
2620   \seq_if_empty:cTF {
2621     l_stex_symdecl_ #1 _notations
2622   } {
2623     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
2624   } {
2625     \seq_if_in:cxTF {
2626       l_stex_symdecl_ #1 _notations
2627     }
2628     { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
2629       \str_set:Nn \l_stex_current_symbol_str { #1 }
2630       \use:c{
2631         stex_notation_ #1 \c_hash_str
2632         \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2633         _cs
2634       }
2635     }{
2636       \str_if_empty:NTF \l__stex_terms_variant_str {
2637         \str_if_empty:NTF \l__stex_terms_lang_str {
2638           \seq_get_left:cN {
2639             l_stex_symdecl_ #1 _notations
2640           } \l_tmpa_str
2641           \str_set:Nn \l_stex_current_symbol_str { #1 }
2642           \use:c{
2643             stex_notation_ #1 \c_hash_str \l_tmpa_str
2644             _cs
2645           }
2646         }{
2647           \msg_error:nnxx{stex}{error/nonotation}{#1}{
2648             ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2649           }
2650         }
2651       }{
2652         \msg_error:nnxx{stex}{error/nonotation}{#1}{
2653           ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str

```

```

2654     }
2655   }
2656 }
2657 }
2658 }

```

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

`\_stex_terms_invoke_text:n`

```

2659 \cs_new_protected:Nn \_stex_terms_invoke_text:n {
2660   \peek_charcode_remove:NTF ! {
2661     \stex_term_custom:nn { #1 } { }
2662   }{
2663     \prop_set_eq:Nc \l_tmpa_prop {
2664       l_stex_symdecl_ #1 _prop
2665     }
2666     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2667     \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2668   }
2669 }

```

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

## 31.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
2670 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2671 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2672 \int_new:N \l__stex_terms_downprec
2673 \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 39.)

Bracketing:

```

\l_stex_terms_left_bracket_str
\l_stex_terms_right_bracket_str
2674 \tl_set:Nn \l__stex_terms_left_bracket_str (
2675 \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

```

2676 \cs_new_protected:Nn \_stex_terms_maybe_brackets:nn {
2677   \bool_if:NTF \l__stex_terms_brackets_done_bool {
2678     \bool_set_false:N \l__stex_terms_brackets_done_bool
2679     #2
2680   } {
2681     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
2682       \bool_if:NTF \l_stex_inarray_bool { #2 }{
2683         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
2684         \dobrackets { #2 }
2685       }

```

```

2686     }{ #2 }
2687   }
2688 }

```

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

### `\dobrackets`

```

2689 \bool_new:N \l__stex_terms_brackets_done_bool
2690 %\RequirePackage{scalerel}
2691 \cs_new_protected:Npn \dobrackets #1 {
2692   %\ThisStyle{\if D\m@switch
2693   %   \exp_args:Nnx \use:nn
2694   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
2695   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
2696   % \else
2697   \exp_args:Nnx \use:nn
2698   {
2699     \bool_set_true:N \l__stex_terms_brackets_done_bool
2700     \int_set:Nn \l__stex_terms_downprec \infprec
2701     \l__stex_terms_left_bracket_str
2702     #1
2703   }
2704   {
2705     \bool_set_false:N \l__stex_terms_brackets_done_bool
2706     \l__stex_terms_right_bracket_str
2707     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2708   }
2709   %\fi}
2710 }

```

(End definition for `\dobrackets`. This function is documented on page 39.)

### `\withbrackets`

```

2711 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2712   \exp_args:Nnx \use:nn
2713   {
2714     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
2715     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
2716     #3
2717   }
2718   {
2719     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
2720     {\l__stex_terms_left_bracket_str}
2721     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
2722     {\l__stex_terms_right_bracket_str}
2723   }
2724 }

```

(End definition for `\withbrackets`. This function is documented on page 39.)

### `\STEXinvisible`

```

2725 \cs_new_protected:Npn \STEXinvisible #1 {
2726   \stex_annotate_invisible:n { #1 }
2727 }

```

(End definition for \STEXinvisible. This function is documented on page 40.)

OMDoc terms:

`\_stex_term_math_oms:nnnn`

```

2728 \cs_new_protected:Nn \_stex_term_oms:nnn {
2729   \stex_annotate:nnn{ OMID }{ #2 }{
2730     \stex_highlight_term:nn { #1 } { #3 }
2731   }
2732 }
2733
2734 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2735   \__stex_terms_maybe_brackets:nn { #3 }{
2736     \stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2737   }
2738 }

```

(End definition for \\_stex\_term\_math\_oms:nnnn. This function is documented on page 38.)

`\_stex_term_math_oma:nnnn`

```

2739 \cs_new_protected:Nn \_stex_term_oma:nnn {
2740   \stex_annotate:nnn{ OMA }{ #2 }{
2741     \stex_highlight_term:nn { #1 } { #3 }
2742   }
2743 }
2744
2745 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2746   \__stex_terms_maybe_brackets:nn { #3 }{
2747     \stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2748   }
2749 }

```

(End definition for \\_stex\_term\_math\_oma:nnnn. This function is documented on page 38.)

`\_stex_term_math_omb:nnnn`

```

2750 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2751   \stex_annotate:nnn{ OMBIND }{ #2 }{
2752     \stex_highlight_term:nn { #1 } { #3 }
2753   }
2754 }
2755
2756 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2757   \__stex_terms_maybe_brackets:nn { #3 }{
2758     \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2759   }
2760 }

```

(End definition for \\_stex\_term\_math\_omb:nnnn. This function is documented on page 38.)

`\_stex_term_math_arg:nnn`

```

2761 \cs_new_protected:Nn \_stex_term_arg:nn {
2762   \stex_unhighlight_term:n {
2763     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2764   }
2765 }

```

```

2766 \cs_new_protected:Nn \stex_term_math_arg:nnn {
2767   \exp_args:Nnx \use:nn
2768     { \int_set:Nn \l__stex_terms_downprec { #2 }
2769       \stex_term_arg:nn { #1 }{ #3 }
2770     }
2771   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2772 }

```

(End definition for `\stex_term_math_arg:nnn`. This function is documented on page 38.)

`\stex_term_math_assoc_arg:nnnn`

```

2773 \cs_new_protected:Nn \stex_term_math_assoc_arg:nnnn {
2774   \clist_set:Nn \l_tmpa_clist{ #4 }
2775   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
2776     \tl_set:Nn \l_tmpa_tl { #4 }
2777   }{
2778     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
2779     \clist_reverse:N \l_tmpa_clist
2780     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
2781
2782     \clist_map_inline:Nn \l_tmpa_clist {
2783       \exp_args:NNNo \exp_args:Nno \tl_set:Nn \l_tmpa_tl {
2784         \exp_args:Nno
2785           \l_tmpa_cs { ##1 } \l_tmpa_tl
2786       }
2787     }
2788
2789   }
2790   \exp_args:Nnno
2791   \stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2792 }

```

(End definition for `\stex_term_math_assoc_arg:nnnn`. This function is documented on page 38.)

`\stex_term_custom:nn`

```

2793 \cs_new_protected:Nn \stex_term_custom:nn {
2794   \str_set:Nn \l__stex_terms_custom_uri { #1 }
2795   \str_set:Nn \l_tmpa_str { #2 }
2796   \tl_clear:N \l_tmpa_tl
2797   \int_zero:N \l_tmpa_int
2798   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2799   \__stex_terms_custom_loop:
2800 }

```

(End definition for `\stex_term_custom:nn`. This function is documented on page 40.)

`\__stex_terms_custom_loop:`

```

2801 \cs_new_protected:Nn \__stex_terms_custom_loop: {
2802   \bool_set_false:N \l_tmpa_bool
2803   \bool_while_do:nn {
2804     \str_if_eq_p:ee X {
2805       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2806     }
2807   }{
2808     \int_incr:N \l_tmpa_int

```

```

2809 }
2810
2811 \peek_charcode:NTF [ {
2812   % notation/text component
2813   \__stex_terms_custom_component:w
2814 } {
2815   \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2816     % all arguments read => finish
2817     \__stex_terms_custom_final:
2818   } {
2819     % arguments missing
2820     \peek_charcode_remove:NTF * {
2821       % invisible, specific argument position or both
2822       \peek_charcode:NTF [ {
2823         % visible specific argument position
2824         \__stex_terms_custom_arg:wn
2825       } {
2826         % invisible
2827         \peek_charcode_remove:NTF * {
2828           % invisible specific argument position
2829           \__stex_terms_custom_arg_inv:wn
2830         } {
2831           % invisible next argument
2832           \__stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2833         }
2834       }
2835     } {
2836       % next normal argument
2837       \__stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
2838     }
2839   }
2840 }
2841 }

```

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

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

```

2842 \cs_new_protected:Npn \__stex_terms_custom_arg_inv:wn [ #1 ] #2 {
2843   \bool_set_true:N \l_tmpa_bool
2844   \__stex_terms_custom_arg:wn [ #1 ] { #2 }
2845 }

```

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

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

```

2846 \cs_new_protected:Npn \__stex_terms_custom_arg:wn [ #1 ] #2 {
2847   \str_set:Nx \l_tmpb_str {
2848     \str_item:Nn \l_tmpa_str { #1 }
2849   }
2850   \str_case:VnTF \l_tmpb_str {
2851     { X } {
2852       \msg_error:nnx{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2853     }
2854     { i } { \__stex_terms_custom_set_X:n { #1 } }
2855     { b } { \__stex_terms_custom_set_X:n { #1 } }

```



```

2856     { a } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
2857     { B } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
2858   }{}{
2859     \msg_error:nnx{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2860   }
2861
2862   \bool_if:nTF \l_tmpa_bool {
2863     \tl_put_right:Nx \l_tmpa_tl {
2864       \stex_annotate_invisible:n {
2865         \stex_term_arg:nn { \int_eval:n { #1 } }
2866         \exp_not:n { { #2 } }
2867       }
2868     }
2869   } {
2870     \tl_put_right:Nx \l_tmpa_tl {
2871       \stex_term_arg:nn { \int_eval:n { #1 } }
2872       \exp_not:n { { #2 } }
2873     }
2874   }
2875
2876   \__stex_terms_custom_loop:
2877 }

```

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

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

```

2878 \cs_new_protected:Nn \__stex_terms_custom_set_X:n {
2879   \str_set:Nx \l_tmpa_str {
2880     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
2881     X
2882     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
2883   }
2884 }

```

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

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

```

2885 \cs_new_protected:Npn \__stex_terms_custom_component:w [ #1 ] {
2886   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
2887   \__stex_terms_custom_loop:
2888 }

```

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

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

```

2889 \cs_new_protected:Nn \__stex_terms_custom_final: {
2890   \int_compare:nNnTF \l_tmpb_int = 0 {
2891     \exp_args:Nnno \stex_term_oms:nnn
2892   }{
2893     \str_if_in:NnTF \l_tmpa_str {b} {
2894       \exp_args:Nnno \stex_term_ombind:nnn
2895     } {
2896       \exp_args:Nnno \stex_term_oma:nnn
2897     }
2898   }

```

```

2899 { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
2900 }

```

(End definition for `\_stex_terms_custom_final:`.)

`\symref`

`\symname`

```

2901 \NewDocumentCommand \symref { m m }{
2902   \let\compemph_uri_prev:\compemph@uri
2903   \let\compemph@uri\symrefemph@uri
2904   \STEXsymbol{#1}! [#2]
2905   \let\compemph@uri\compemph_uri_prev:
2906 }
2907
2908 \keys_define:nn { stex / symname } {
2909   post      .str_set_x:N    = \l_stex_symname_post_str
2910 }
2911
2912 \cs_new_protected:Nn \stex_symname_args:n {
2913   \str_clear:N \l_stex_symname_post_str
2914   \keys_set:nn { stex / symname } { #1 }
2915 }
2916
2917 \NewDocumentCommand \symname { 0{} m }{
2918   \stex_symname_args:n { #1 }
2919   \stex_get_symbol:n { #2 }
2920   \str_set:Nx \l_tmpa_str {
2921     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
2922   }
2923   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
2924
2925   \let\compemph_uri_prev:\compemph@uri
2926   \let\compemph@uri\symrefemph@uri
2927   \exp_args:NNx \use:nn
2928   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }! [
2929     \l_tmpa_str \l_stex_symname_post_str
2930   ] }
2931   \let\compemph@uri\compemph_uri_prev:
2932 }

```

(End definition for `\symref` and `\symname`. These functions are documented on page 38.)

### 31.3 Notation Components

```

2933 <@@=stex_notationcomps>

```

`\stex_highlight_term:nn`

```

2934
2935 \str_new:N \l_stex_current_symbol_str
2936 \cs_new_protected:Nn \stex_highlight_term:nn {
2937   \exp_args:Nnx
2938   \use:nn {
2939     \str_set:Nx \l_stex_current_symbol_str { #1 }
2940     #2
2941   } {

```

```

2942 \str_set:Nx \exp_not:N \l_stex_current_symbol_str
2943 { \l_stex_current_symbol_str }
2944 }
2945 }
2946
2947 \cs_new_protected:Nn \stex_unhighlight_term:n {
2948 % \latexml_if:TF {
2949 % #1
2950 % } {
2951 % \rustex_if:TF {
2952 % #1
2953 % } {
2954 #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
2955 % }
2956 % }
2957 }

```

(End definition for `\stex_highlight_term:nn`. This function is documented on page 40.)

```

\comp
\compemph@uri
\compemph
\defemph
\defemph@uri
\symrefemph
\symrefemph@uri
2958 \cs_new_protected:Npn \comp #1 {
2959 \str_if_empty:NF \l_stex_current_symbol_str {
2960 \rustex_if:TF {
2961 \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
2962 }{
2963 \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
2964 }
2965 }
2966 }
2967
2968 \cs_new_protected:Npn \compemph@uri #1 #2 {
2969 \compemph{ #1 }
2970 }
2971
2972
2973 \cs_new_protected:Npn \compemph #1 {
2974 #1
2975 }
2976
2977 \cs_new_protected:Npn \defemph@uri #1 #2 {
2978 \defemph{#1}
2979 }
2980
2981 \cs_new_protected:Npn \defemph #1 {
2982 \textbf{#1}
2983 }
2984
2985 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
2986 \symrefemph{#1}
2987 }
2988
2989 \cs_new_protected:Npn \symrefemph #1 {
2990 \textbf{#1}
2991 }

```

(End definition for `\comp` and others. These functions are documented on page 40.)

## `\ellipses`

```
2992 \NewDocumentCommand \ellipses {} { \ldots }
```

(End definition for `\ellipses`. This function is documented on page 40.)

```

\parray
\prmatrix 2993 \bool_new:N \l_stex_inarray_bool
\parrayline 2994 \bool_set_false:N \l_stex_inarray_bool
\parraylineh 2995 \NewDocumentCommand \parray { m m } {
\parraycell 2996 \begin{group}
2997 \bool_set_true:N \l_stex_inarray_bool
2998 \begin{array}{#1}
2999 #2
3000 \end{array}
3001 \end{group}
3002 }
3003
3004 \NewDocumentCommand \prmatrix { m } {
3005 \begin{group}
3006 \bool_set_true:N \l_stex_inarray_bool
3007 \begin{matrix}
3008 #1
3009 \end{matrix}
3010 \end{group}
3011 }
3012
3013 \def \maybepline {
3014 \bool_if:NT \l_stex_inarray_bool {\hline}
3015 }
3016
3017 \def \parrayline #1 #2 {
3018 #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3019 }
3020
3021 \def \pmrow #1 { \parrayline{}{ #1 } }
3022
3023 \def \parraylineh #1 #2 {
3024 #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3025 }
3026
3027 \def \parraycell #1 {
3028 #1 \bool_if:NT \l_stex_inarray_bool {&}
3029 }

```

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

```
3030 \end{package}
```

## Chapter 32

# STEX -Structural Features Implementation

```
3031 <*package>
3032
3033 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3034
3035 <@@=stex_features>
3036
3037   Warnings and error messages
3038 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3039   Symbol~#1~can~not~be~assigned~in~copymodule~#2
3040 }
3041 \msg_new:nnn{stex}{error/interpretmodule/noddefinens}{
3042   Symbol~#1~not~assigned~in~interpretmodule~#2
3043 }
```

### 32.1 Imports with modification

```
3043 \cs_new_protected:Nn \stex_get_symbol_in_copymodule:n {
3044   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3045     \__stex_features_get_symbol_from_cs:n { #1 }
3046   }{
3047     % argument is a string
3048     % is it a command name?
3049     \cs_if_exist:cTF { #1 }{
3050       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3051       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3052       \str_if_empty:NNTF \l_tmpa_str {
3053         \exp_args:Nx \cs_if_eq:NNTF {
3054           \tl_head:N \l_tmpa_tl
3055         } \stex_invoke_symbol:n {
3056           \exp_args:No \__stex_features_get_symbol_from_cs:n { \use:c { #1 } }
3057         }{
3058           \__stex_features_get_symbol_from_string:n { #1 }
3059         }
3060       }
3061     }
3062   }
```

```

3059     }
3060   } {
3061     \__stex_features_get_symbol_from_string:n { #1 }
3062   }
3063   ){
3064     % argument is not a command name
3065     \__stex_features_get_symbol_from_string:n { #1 }
3066     % \l_stex_all_symbols_seq
3067   }
3068 }
3069 }
3070
3071 \cs_new_protected:Nn \__stex_features_get_symbol_from_string:n {
3072   \str_set:Nn \l_tmpa_str { #1 }
3073   \bool_set_false:N \l_tmpa_bool
3074   \bool_if:NF \l_tmpa_bool {
3075     \tl_set:Nn \l_tmpa_tl {
3076       \msg_set:nnn{stex}{error/unknownsymbol}{
3077         No~symbol~#1~found!
3078       }
3079       \msg_error:nn{stex}{error/unknownsymbol}
3080     }
3081     \str_set:Nn \l_tmpa_str { #1 }
3082     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3083     \seq_map_inline:Nn \l__stex_features_copymodule_fields_seq {
3084       \str_set:Nn \l_tmpb_str { ##1 }
3085       \str_if_eq:eeT { \l_tmpa_str } {
3086         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3087       } {
3088         \seq_map_break:n {
3089           \tl_set:Nn \l_tmpa_tl {
3090             \str_set:Nn \l_stex_get_symbol_uri_str {
3091               ##1
3092             }
3093             \__stex_features_get_symbol_check:
3094           }
3095         }
3096       }
3097     }
3098     \l_tmpa_tl
3099   }
3100 }
3101
3102 \cs_new_protected:Nn \__stex_features_get_symbol_from_cs:n {
3103   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3104   { \tl_tail:N \l_tmpa_tl }
3105   \tl_if_single:NTF \l_tmpa_tl {
3106     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3107       \exp_after:wN \str_set:Nn \exp_after:wN
3108       \l_stex_get_symbol_uri_str \l_tmpa_tl
3109       \__stex_features_get_symbol_check:
3110     }{
3111       % TODO
3112       % tail is not a single group

```

```

3113     }
3114   }{
3115     % TODO
3116     % tail is not a single group
3117   }
3118 }
3119
3120 \cs_new_protected:Nn \__stex_features_get_symbol_check: {
3121   \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq {?} \l_stex_get_symbol_uri_str
3122   \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 3 {
3123     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
3124     \str_set:Nx \l_tmpa_str {\seq_use:Nn \l_tmpa_seq ?}
3125     \seq_if_in:NoF \l__stex_features_copymodule_modules_seq \l_tmpa_str {
3126       \msg_error:nxxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{\l_stex_c
3127     }
3128   }{
3129     \msg_error:nxxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{\l_stex_c
3130   }
3131 }
3132
3133 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
3134   \stex_import_module_uri:nn { #1 } { #2 }
3135   \stex_deactivate_macro:Nn \symdecl {module~environments}
3136   \stex_deactivate_macro:Nn \symdef {module~environments}
3137   \stex_deactivate_macro:Nn \notation {module~environments}
3138   \stex_reactivate_macro:N \assign
3139   \stex_reactivate_macro:N \renamedec1
3140   \stex_reactivate_macro:N \donotcopy
3141   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3142   \stex_import_require_module:nnnn
3143     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3144     { \l_stex_import_path_str } { \l_stex_import_name_str }
3145   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3146   \seq_set_eq:NN \l__stex_features_copymodule_modules_seq \l_stex_collect_imports_seq
3147   \seq_put_left:Nx \l__stex_features_copymodule_modules_seq {\l_stex_import_ns_str ?\l_stex_
3148   \seq_clear:N \l__stex_features_copymodule_fields_seq
3149   \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3150     \seq_map_inline:cn {c_stex_module_##1 constants}{
3151       \exp_args:NNx \seq_put_right:Nn \l__stex_features_copymodule_fields_seq {
3152         ##1 ? ####1
3153       }
3154     }
3155   }
3156   \seq_clear:N \l_tmpa_seq
3157   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3158     name      = \l_stex_current_copymodule_name_str ,
3159     module    = \l_stex_current_module_str ,
3160     from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3161     includes  = \l_tmpa_seq ,
3162     fields    = \l_tmpa_seq
3163   }
3164   \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3165     as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3166   \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_features_copymodule_fields_seq {,~}

```

```

3167 \stex_if_smsmode:TF {
3168   \stex_smsmode_set_codes:
3169 } {
3170   \begin{stex_annotate_env} {#4} {
3171     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3172   }
3173   \stex_annotate_invisible:nnn{from}{\l_stex_import_ns_str ?\l_stex_import_name_str}{\}
3174 }
3175 \bool_set_eq:NN \l__stex_features_oldhtml_bool \l_stex_html_do_output_bool
3176 \bool_set_false:N \l_stex_html_do_output_bool
3177 }
3178 \cs_new_protected:Nn \stex_copymodule_end:n {
3179   \bool_set_eq:NN \l_stex_html_do_output_bool \l__stex_features_oldhtml_bool
3180   \tl_clear:N \l_tmpa_tl
3181   \prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3182   \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3183     \seq_map_inline:cn {c_stex_module_##1_constants}{\stex_annotate:nnn{assignment} {##1?###
3184       #1
3185       \str_if_exist:cTF {l__stex_features_copymodule_##1?####1_name_str} {
3186         \tl_put_right:Nx \l_tmpa_tl {
3187           \prop_set_from_keyval:cn {
3188             l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_#
3189           }}{
3190             \exp_after:wN \prop_to_keyval:N \csname
3191               l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule
3192             \endcsname
3193           }
3194         \seq_clear:c {
3195           l_stex_symdecl_
3196           \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_name_s
3197         _notations
3198       }
3199     }
3200     \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_features_copymodule_##1?####1_name
3201     \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{l__stex_features_
3202     \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3203       \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?###
3204       \tl_put_right:Nx \l_tmpa_tl {
3205         \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{\
3206         \stex_invoke_symbol:n {
3207           \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_na
3208         }
3209       }
3210     }
3211   }
3212 }{
3213   \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
3214   \prop_put:Nnx \l_tmpa_prop { name }{\l_stex_current_copymodule_name_str / ####1 }
3215   \prop_put:Nnx \l_tmpa_prop { module }{\l_stex_current_module_str }
3216   \tl_put_right:Nx \l_tmpa_tl {
3217     \prop_set_from_keyval:cn {
3218       l_stex_symdecl_\l_stex_current_module_str ? \l_stex_current_copymodule_name_str
3219     }}{
3220       \prop_to_keyval:N \l_tmpa_prop

```



```

3221     }
3222     \seq_clear:c {
3223         l_stex_symdecl_
3224         \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / #####1
3225         _notations
3226     }
3227 }
3228 \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str / #####1}
3229 \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3230     \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3231         \tl_put_right:Nx \l_tmpa_tl {
3232             \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3233                 \stex_invoke_symbol:n {
3234                     \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / #####1
3235                 }
3236             }
3237         }
3238     }
3239 }
3240 \tl_if_exist:cT {l__stex_features_copymodule_##1?####1_def_tl}{
3241     \stex_annotate_invisible:nnn{definiens}{\use:c{l__stex_features_copymodule_##1?####1_def_tl}}{
3242     }
3243     % todo notations
3244 }
3245 }
3246 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3247 \tl_put_left:Nx \l_tmpa_tl {
3248     \prop_set_from_keyval:cn {
3249         l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _pro
3250     }{
3251         \prop_to_keyval:N \l_stex_current_copymodule_prop
3252     }
3253 }
3254 \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl
3255 \stex_debug:nn{copymodule}{result:\meaning \l_tmpa_tl}
3256 \exp_args:Nx \stex_do_aftergroup:n {
3257     \exp_args:No \exp_not:n \l_tmpa_tl
3258 }
3259 \stex_if_smsmode:F {
3260     \end{stex_annotate_env}
3261 }
3262 }
3263
3264 \NewDocumentEnvironment {copymodule} { 0{} m m}{
3265     \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ structure }
3266 }{
3267     \stex_copymodule_end:n {}
3268 }
3269
3270 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
3271     \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ realization }
3272 }{
3273     \stex_copymodule_end:n {
3274         \tl_if_exist:cF {

```

```

3275     \l__stex_features_copymodule_#####1?#####1_def_tl
3276   }{
3277     \msg_error:nnxx{stex}{error/interpretmodule/nodedefiniens}{
3278       #####1?#####1
3279     }\l_stex_current_copymodule_name_str}
3280   }
3281 }
3282 }
3283
3284 \NewDocumentCommand \donotcopy { 0{} m}{
3285   \stex_import_module_uri:nn { #1 } { #2 }
3286   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3287   \seq_map_inline:Nn \l_stex_collect_imports_seq {
3288     \seq_remove_all:Nn \l__stex_features_copymodule_modules_seq { ##1 }
3289     \seq_map_inline:cn {c_stex_module_##1_constants}{
3290       \seq_remove_all:Nn \l__stex_features_copymodule_fields_seq { ##1 ? #####1 }
3291       \bool_lazy_any_p:nT {
3292         { \cs_if_exist_p:c {\l__stex_features_copymodule_##1?#####1_name_str}}
3293         { \cs_if_exist_p:c {\l__stex_features_copymodule_##1?#####1_macroname_str}}
3294         { \cs_if_exist_p:c {\l__stex_features_copymodule_##1?#####1_def_tl}}
3295       }{
3296         % TODO throw error
3297       }
3298     }
3299   }
3300
3301   \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
3302   \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_ns_str ?\l_stex_import_name_str }
3303   \prop_put:Nnx \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
3304 }
3305
3306 \NewDocumentCommand \assign { m m m}{
3307   \stex_get_symbol_in_copymodule:n {#1}
3308   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
3309   \tl_set:cn {\l__stex_features_copymodule_##1?#####1_def_tl}{#2}
3310 }
3311
3312 \keys_define:nn { stex / renamedec1 } {
3313   name .str_set_x:N = \l_stex_renamedec1_name_str
3314 }
3315 \cs_new_protected:Nn \__stex_features_renamedec1_args:n {
3316   \str_clear:N \l_stex_renamedec1_name_str
3317
3318   \keys_set:nn { stex / renamedec1 } { #1 }
3319 }
3320
3321 \NewDocumentCommand \renamedec1 { 0{} m m}{
3322   \__stex_features_renamedec1_args:n { #1 }
3323   \stex_get_symbol_in_copymodule:n {#2}
3324   \stex_debug:nn{renamedec1}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
3325   \str_set:cx {\l__stex_features_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
3326   \str_if_empty:NTF \l_stex_renamedec1_name_str {
3327     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3328       \l_stex_get_symbol_uri_str

```

```

3329     } }
3330   } {
3331     \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str_name_str}{\l_stex_r
3332     \stex_debug:nn{renamedecl}{@~\l_stex_current_module_str ? \l_stex_renamedecl_name_str}
3333     \prop_set_eq:cc {l_stex_symdecl_
3334       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
3335       _prop
3336     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
3337     \seq_set_eq:cc {l_stex_symdecl_
3338       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
3339       _notations
3340     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
3341     \prop_put:cnx {l_stex_symdecl_
3342       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
3343       _prop
3344     }{ name }{ \l_stex_renamedecl_name_str }
3345     \prop_put:cnx {l_stex_symdecl_
3346       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
3347       _prop
3348     }{ module }{ \l_stex_current_module_str }
3349     \exp_args:NNx \seq_put_left:Nn \l_stex_features_copymodule_fields_seq {
3350       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
3351     }
3352     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3353       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
3354     } }
3355   }
3356 }
3357 %\NewDocumentCommand \notation_in_copymodules: { 0{} m } {
3358 % \_stex_notation_args:n { #1 }
3359 % \tl_clear:N \l_stex_symdecl_definiens_tl
3360 % \stex_get_symbol_in_copymodule:n { #2 }
3361 % \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
3362 % % todo
3363 %}
3364 \stex_deactivate_macro:Nn \assign {copymodules}
3365 \stex_deactivate_macro:Nn \renamedecl {copymodules}
3366 \stex_deactivate_macro:Nn \donotcopy {copymodules}
3367
3368
3369 \seq_new:N \l_stex_implicit_morphisms_seq
3370 \NewDocumentCommand \implicitmorphism { 0{} m m }{
3371   \stex_import_module_uri:nn { #1 } { #2 }
3372   \stex_debug:nn{implicits}{
3373     Implicit-morphism:~
3374     \l_stex_module_ns_str ? \l__stex_features_name_str
3375   }
3376   \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
3377     \l_stex_module_ns_str ? \l__stex_features_name_str
3378   }{
3379     \msg_error:nnn{stex}{error/conflictingmodules}{
3380       \l_stex_module_ns_str ? \l__stex_features_name_str
3381     }
3382   }

```

```

3383
3384 % TODO
3385
3386
3387
3388 \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
3389   \l_stex_module_ns_str ? \l__stex_features_name_str
3390 }
3391 }
3392

```

## 32.2 The feature environment

structural@feature

```

3393
3394 \NewDocumentEnvironment{structural@feature}{ m m m }{
3395   \stex_if_in_module:F {
3396     \msg_set:nnn{stex}{error/nomodule}{
3397       Structural~Feature~has~to~occur~in~a~module:\\
3398       Feature~#2~of~type~#1\\
3399       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
3400     }
3401     \msg_error:nn{stex}{error/nomodule}
3402   }
3403
3404   \str_set:Nx \l_stex_module_name_str {
3405     \prop_item:Nn \l_stex_current_module_prop
3406       { name } / #2 - feature
3407   }
3408
3409   \str_set:Nx \l_stex_module_ns_str {
3410     \prop_item:Nn \l_stex_current_module_prop
3411       { ns }
3412   }
3413
3414
3415   \str_clear:N \l_tmpa_str
3416   \seq_clear:N \l_tmpa_seq
3417   \tl_clear:N \l_tmpa_tl
3418   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
3419     origname = #2,
3420     name     = \l_stex_module_name_str ,
3421     ns       = \l_stex_module_ns_str ,
3422     imports  = \exp_not:o { \l_tmpa_seq } ,
3423     constants = \exp_not:o { \l_tmpa_seq } ,
3424     content  = \exp_not:o { \l_tmpa_tl } ,
3425     file     = \exp_not:o { \g_stex_currentfile_seq } ,
3426     lang     = \l_stex_module_lang_str ,
3427     sig      = \l_tmpa_str ,
3428     meta     = \l_tmpa_str ,
3429     feature  = #1 ,
3430   }
3431
3432   \stex_if_smsmode:TF {

```

```

3433 \stex_smsmode_set_codes:
3434 } {
3435 \begin{stex_annotate_env}{ feature:#1 }{ }
3436 \stex_annotate_invisible:nnn{header}{ }{ #3 }
3437 }
3438 }{
3439 \str_set:Nx \l_tmpa_str {
3440 c_stex_feature_
3441 \prop_item:Nn \l_stex_current_module_prop { ns } ?
3442 \prop_item:Nn \l_stex_current_module_prop { name }
3443 _prop
3444 }
3445 \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
3446 \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
3447 \stex_if_smsmode:TF {
3448 \exp_args:Nx \stex_add_to_sms:n {
3449 \prop_gset_from_keyval:cn {
3450 c_stex_feature_
3451 \prop_item:Nn \l_stex_current_module_prop { ns } ?
3452 \prop_item:Nn \l_stex_current_module_prop { name }
3453 _prop
3454 } {
3455 origname = #2,
3456 name = \prop_item:cn { \l_tmpa_str } { name } ,
3457 ns = \prop_item:cn { \l_tmpa_str } { ns } ,
3458 imports = \prop_item:cn { \l_tmpa_str } { imports } ,
3459 constants = \prop_item:cn { \l_tmpa_str } { constants } ,
3460 content = \prop_item:cn { \l_tmpa_str } { content } ,
3461 file = \prop_item:cn { \l_tmpa_str } { file } ,
3462 lang = \prop_item:cn { \l_tmpa_str } { lang } ,
3463 sig = \prop_item:cn { \l_tmpa_str } { sig } ,
3464 meta = \prop_item:cn { \l_tmpa_str } { meta } ,
3465 feature = \prop_item:cn { \l_tmpa_str } { feature }
3466 }
3467 }
3468 } {
3469 \end{stex_annotate_env}
3470 }
3471 }
3472

```

## 32.3 Features

structure

```

3473
3474 \prop_new:N \l_stex_all_structures_prop
3475
3476 \keys_define:nn { stex / features / structure } {
3477 name .str_set_x:N = \l__stex_features_structure_name_str ,
3478 }
3479
3480 \cs_new_protected:Nn \__stex_features_structure_args:n {
3481 \str_clear:N \l__stex_features_structure_name_str

```

```

3482 \keys_set:nn { stex / features / structure } { #1 }
3483 }
3484
3485 %\stex_new_feature:nnnn { structure } { 0{} m } {
3486 % \__stex_features_structure_args:n { ##1 }
3487 % \str_if_empty:NT \l__stex_features_structure_name_str {
3488 % \str_set:Nx \l__stex_features_structure_name_str { ##2 }
3489 % }
3490 %} {
3491 %
3492 %}
3493
3494 \NewDocumentEnvironment{mathstructure}{ 0{} m }{
3495 \__stex_features_structure_args:n { #1 }
3496 \str_if_empty:NT \l__stex_features_structure_name_str {
3497 \str_set:Nx \l__stex_features_structure_name_str { #2 }
3498 }
3499 \exp_args:Nnnx
3500 \begin{structural@feature}{ structure }
3501 { \l__stex_features_structure_name_str }{}
3502 \seq_clear:N \l_tmpa_seq
3503 \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
3504
3505 }{
3506 \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
3507 \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
3508 \str_set:Nx \l_tmpa_str {
3509 \prop_item:Nn \l_stex_current_module_prop { ns } ?
3510 \prop_item:Nn \l_stex_current_module_prop { name }
3511 }
3512 \seq_map_inline:Nn \l_tmpa_seq {
3513 \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
3514 }
3515 \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
3516 \exp_args:Nnx
3517 \AddToHookNext { env / mathstructure / after }{
3518 \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}}{
3519 \stex_term_math_oms:nnnn { \l_tmpa_str }{}{0}{}
3520 }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
3521 \STEXexport {
3522 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3523 {\prop_item:Nn \l_stex_current_module_prop { origname }}
3524 {\l_tmpa_str}
3525 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3526 {#2}{\l_tmpa_str}
3527 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3528 % \prop_item:Nn \l_stex_current_module_prop { origname },
3529 % \l_tmpa_str
3530 % }
3531 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3532 % #2,\l_tmpa_str
3533 % }
3534 % \tl_set:cx { #2 } {
3535 % \stex_invoke_structure:n { \l_tmpa_str }

```

```

3536     }
3537 }
3538
3539 \end{structural@feature}
3540 % \g_stex_last_feature_prop
3541 }

\instantiate

3542 \seq_new:N \l__stex_features_structure_field_seq
3543 \str_new:N \l__stex_features_structure_field_str
3544 \str_new:N \l__stex_features_structure_def_tl
3545 \prop_new:N \l__stex_features_structure_prop
3546 \NewDocumentCommand \instantiate { m O{} m }{
3547   \stex_smsmode_set_codes:
3548   \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
3549   \prop_set_eq:Nc \l__stex_features_structure_prop {
3550     c_stex_feature_\l_tmpa_str _prop
3551   }
3552   \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
3553   \seq_map_inline:Nn \l__stex_features_structure_field_seq {
3554     \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
3555     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3556       \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
3557       \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
3558         {!} \l_tmpa_tl
3559       \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3560         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
3561         \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3562         \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3563       }{
3564         \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
3565         \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3566         \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
3567           \l_tmpa_tl
3568         \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3569           \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
3570           \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3571         }{
3572           \tl_clear:N \l_tmpb_tl
3573         }
3574       }
3575     }{
3576       \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
3577       \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3578         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
3579         \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
3580         \tl_clear:N \l_tmpa_tl
3581       }{
3582         % TODO throw error
3583       }
3584     }
3585     % \l_tmpa_str: name
3586     % \l_tmpa_tl: definiens
3587     % \l_tmpb_tl: notation

```

```

3588 \tl_if_empty:NT \l__stex_features_structure_field_str {
3589   % TODO throw error
3590 }
3591 \str_clear:N \l_tmpb_str
3592
3593 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3594 \seq_map_inline:Nn \l_tmpa_seq {
3595   \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
3596   \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
3597   \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
3598     \seq_map_break:n {
3599       \str_set:Nn \l_tmpb_str { ####1 }
3600     }
3601   }
3602 }
3603 \prop_get:cnN { l_stex_symdecl_ \l_tmpb_str _prop } {args}
3604 \l_tmpb_str
3605
3606 \tl_if_empty:NTF \l_tmpb_tl {
3607   \tl_if_empty:NF \l_tmpa_tl {
3608     \exp_args:Nx \use:n {
3609       \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fe
3610     }
3611   }
3612 }{
3613   \tl_if_empty:NTF \l_tmpa_tl {
3614     \exp_args:Nx \use:n {
3615       \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\
3616     }
3617   }{
3618     \exp_args:Nx \use:n {
3619       \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
3620       \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
3621     }
3622   }
3623 }
3624 }
3625 % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
3626 % \prop_item:Nn \l_stex_current_module_prop {name} ?
3627 % #3/\l__stex_features_structure_field_str
3628 % \par
3629 % \expandafter\present\csname
3630 %   l_stex_symdecl_
3631 %   \prop_item:Nn \l_stex_current_module_prop {ns} ?
3632 %   \prop_item:Nn \l_stex_current_module_prop {name} ?
3633 %   #3/\l__stex_features_structure_field_str
3634 %   _prop
3635 % \endcsname
3636 }
3637
3638 \tl_clear:N \l__stex_features_structure_def_tl
3639
3640 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3641 \seq_map_inline:Nn \l_tmpa_seq {

```



```

3642 \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3643 \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3644 \exp_args:Nx \use:n {
3645   \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
3646
3647   }
3648 }
3649
3650 \prop_if_exist:cF {
3651   \l_stex_symdecl_
3652   \prop_item:Nn \l_stex_current_module_prop {ns} ?
3653   \prop_item:Nn \l_stex_current_module_prop {name} ?
3654   #3/\l_tmpa_str
3655   _prop
3656 }{
3657   \prop_get:cnN { \l_stex_symdecl_ ##1 _prop } {args}
3658   \l_tmpb_str
3659   \exp_args:Nx \use:n {
3660     \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
3661   }
3662 }
3663 }
3664
3665 \symdecl*[type={\STEXsymbol{module-type}}{
3666   \_stex_term_math_oms:nnnn {
3667     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3668     \prop_item:Nn \l__stex_features_structure_prop {name}
3669     }{}{0}{}}
3670 }{}{#3}
3671
3672 % TODO: -> sms file
3673
3674 \tl_set:cx{ #3 }{
3675   \stex_invoke_structure:nnn {
3676     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3677     \prop_item:Nn \l_stex_current_module_prop {name} ? #3
3678   } {
3679     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3680     \prop_item:Nn \l__stex_features_structure_prop {name}
3681   }
3682 }
3683
3684 }

```

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

\stex\_invoke\_structure:nnn

```

3685 % #1: URI of the instance
3686 % #2: URI of the instantiated module
3687 \cs_new_protected:Nn \stex_invoke_structure:nnn {
3688   \tl_if_empty:nTF{ #3 }{
3689     \prop_set_eq:Nc \l__stex_features_structure_prop {
3690       c_stex_feature_ #2 _prop
3691     }

```

```

3692 \tl_clear:N \l_tmpa_tl
3693 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3694 \seq_map_inline:Nn \l_tmpa_seq {
3695   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3696   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3697   \cs_if_exist:cT {
3698     stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
3699   }{
3700     \tl_if_empty:NF \l_tmpa_tl {
3701       \tl_put_right:Nn \l_tmpa_tl {,}
3702     }
3703     \tl_put_right:Nx \l_tmpa_tl {
3704       \stex_invoke_symbol:n {#1/\l_tmpa_str}!
3705     }
3706   }
3707 }
3708 \exp_args:No \mathstruct \l_tmpa_tl
3709 }{
3710   \stex_invoke_symbol:n{#1/#3}
3711 }
3712 }

```

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

```

3713 </package>

```

## Chapter 33

# STEX -Statements Implementation

```
3714 <*package>
3715
3716 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3717
3718 \protected\def\ignorespacesandpars{
3719   \begingroup\catcode13=10\relax
3720   \@ifnextchar\par{
3721     \endgroup\expandafter\ignorespacesandpars\@gobble
3722   }{
3723     \endgroup
3724   }
3725 }
3726
3727 <@@=stex_statements>
3728
3729 Warnings and error messages
```

\titleemph

```
3729 \def\titleemph#1{\textbf{#1}}
```

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

### 33.1 Definitions

definiendum

```
3730 \keys_define:nn {stex / definiendum }{
3731   post      .tl_set:N      = \l__stex_statements_definiendum_post_tl,
3732   root      .str_set_x:N    = \l__stex_statements_definiendum_root_str,
3733   gfa       .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
3734 }
3735 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
3736   \str_clear:N \l__stex_statements_definiendum_root_str
3737   \tl_clear:N \l__stex_statements_definiendum_post_tl
3738   \str_clear:N \l__stex_statements_definiendum_gfa_str
```

```

3739 \keys_set:nn { stex / definiendum }{ #1 }
3740 }
3741 \NewDocumentCommand \definiendum { 0{ } m m } {
3742   \__stex_statements_definiendum_args:n { #1 }
3743   \stex_get_symbol:n { #2 }
3744   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3745   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
3746     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
3747       \tl_set:Nn \l_tmpa_tl { #3 }
3748     } {
3749       \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
3750       \tl_set:Nn \l_tmpa_tl {
3751         \l__stex_statements_definiendum_root_str\l__stex_statements_definiendum_post_tl
3752       }
3753     }
3754   } {
3755     \tl_set:Nn \l_tmpa_tl { #3 }
3756   }
3757
3758   % TODO root
3759   \rustex_if:TF {
3760     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
3761   } {
3762     \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
3763   }
3764 }
3765 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

#### definame

```

3766 \NewDocumentCommand \definame { 0{ } m } {
3767   \__stex_statements_definiendum_args:n { #1 }
3768   % TODO: root
3769   \stex_get_symbol:n { #2 }
3770   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3771   \str_set:Nx \l_tmpa_str {
3772     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3773   }
3774   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3775   \rustex_if:TF {
3776     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3777       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3778     }
3779   } {
3780     \defemph@uri {
3781       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3782     } { \l_stex_get_symbol_uri_str }
3783   }
3784 }
3785 \stex_deactivate_macro:Nn \definame {definition~environments}

```

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

sdefinition

```

3786
3787 \keys_define:nn {stex / sdefinition }{
3788   type      .str_set_x:N = \sdefinitiontype,
3789   id        .str_set_x:N = \sdefinitionid,
3790   title     .tl_set:N    = \sdefinitiontitle
3791 }
3792 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
3793   \str_clear:N \sdefinitiontype
3794   \str_clear:N \sdefinitionid
3795   \tl_clear:N \sdefinitiontitle
3796   \keys_set:nn { stex / sdefinition }{ #1 }
3797 }
3798
3799 \NewDocumentEnvironment{sdefinition}{0{}}{
3800   \__stex_statements_sdefinition_args:n{ #1 }
3801   \stex_reactivate_macro:N \definiendum
3802   \stex_reactivate_macro:N \definame
3803   \stex_smsmode_set_codes:
3804   \clist_set:No \l_tmpa_clist \sdefinitiontype
3805   \tl_clear:N \l_tmpa_tl
3806   \clist_map_inline:Nn \l_tmpa_clist {
3807     \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
3808       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
3809     }
3810   }
3811   \stex_if_smsmode:F {
3812     \exp_args:Nnnx
3813     \begin{stex_annotate_env}{definition}{}
3814     \str_if_empty:NF \sdefinitiontype {
3815       \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{}
3816     }
3817   }
3818   \tl_if_empty:NTF \l_tmpa_tl {
3819     \__stex_statements_sdefinition_start:
3820   }{
3821     \l_tmpa_tl
3822   }
3823   \stex_ref_new_doc_target:n \sdefinitionid
3824 }{
3825   \clist_set:No \l_tmpa_clist \sdefinitiontype
3826   \tl_clear:N \l_tmpa_tl
3827   \clist_map_inline:Nn \l_tmpa_clist {
3828     \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
3829       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
3830     }
3831   }
3832   \tl_if_empty:NTF \l_tmpa_tl {
3833     \__stex_statements_sdefinition_end:
3834   }{
3835     \l_tmpa_tl
3836   }
3837   \stex_if_smsmode:F {
3838     \end{stex_annotate_env}

```

```

3839 }
3840 }

```

`\stexpatchdefinition`

```

3841 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
3842   \par\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
3843     ~(\sdefinitiontitle)
3844   }~}
3845 }
3846 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\par\medskip}
3847
3848 \newcommand\stexpatchdefinition[3] [] {
3849   \str_set:Nx \l_tmpa_str{ #1 }
3850   \str_if_empty:NTF \l_tmpa_str {
3851     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
3852     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
3853   }{
3854     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
3855     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
3856   }
3857 }

```

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

`\inlinedef inline:`

```

3858 \NewDocumentCommand \inlinedef { m } {
3859   \begingroup
3860   \stex_reactivate_macro:N \definiendum
3861   \stex_reactivate_macro:N \definame
3862   \stex_ref_new_doc_target:n{
3863     #1
3864   }
3865 }

```

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

## 33.2 Assertions

`sassertion`

```

3866
3867 \keys_define:nn {stex / sassertion }{
3868   type      .str_set_x:N = \sassertiontype,
3869   id        .str_set_x:N = \sassertionid,
3870   title     .tl_set:N     = \sassertiontitle ,
3871   name      .str_set_x:N = \sassertionname
3872 }
3873 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
3874   \str_clear:N \sassertiontype
3875   \str_clear:N \sassertionid
3876   \str_clear:N \sassertionname
3877   \tl_clear:N \sassertiontitle
3878   \keys_set:nn { stex / sassertion }{ #1 }
3879 }

```

```

3880
3881 %\tl_new:N \g__stex_statements_aftergroup_tl
3882
3883 \NewDocumentEnvironment{sassertion}{0{}}{
3884   \__stex_statements_sassertion_args:n{ #1 }
3885   \stex_smsmode_set_codes:
3886   \clist_set:No \l_tmpa_clist \sassertiontype
3887   \tl_clear:N \l_tmpa_tl
3888   \clist_map_inline:Nn \l_tmpa_clist {
3889     \tl_if_exist:cT {\__stex_statements_sassertion_##1_start:}{
3890       \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_statements_sassertion_##1_start:}}
3891     }
3892   }
3893   \stex_if_smsmode:F {
3894     \exp_args:Nnnx
3895     \begin{stex_annotate_env}{assertion}{}
3896     \str_if_empty:NF \sassertiontype {
3897       \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
3898     }
3899   }
3900   \tl_if_empty:NTF \l_tmpa_tl {
3901     \__stex_statements_sassertion_start:
3902   }{
3903     \l_tmpa_tl
3904   }
3905   \stex_ref_new_doc_target:n \sassertionid
3906 }{
3907   \clist_set:No \l_tmpa_clist \sassertiontype
3908   \tl_clear:N \l_tmpa_tl
3909   \clist_map_inline:Nn \l_tmpa_clist {
3910     \tl_if_exist:cT {\__stex_statements_sassertion_##1_end:}{
3911       \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_statements_sassertion_##1_end:}}
3912     }
3913   }
3914   \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
3915   \tl_if_empty:NTF \l_tmpa_tl {
3916     \__stex_statements_sassertion_end:
3917   }{
3918     \l_tmpa_tl
3919   }
3920   \stex_if_smsmode:F {
3921     \end{stex_annotate_env}
3922   }
3923 }

```

\stexpatchassertion

```

3924
3925 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
3926   \par\noindent\titllemph{Assertion~\tl_if_empty:NF \sassertiontitle {
3927     (\sassertiontitle)
3928   }~}
3929 }
3930 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
3931

```

```

3932 \newcommand\stexpatchassertion[3] [] {
3933   \str_set:Nx \l_tmpa_str{ #1 }
3934   \str_if_empty:NTF \l_tmpa_str {
3935     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
3936     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
3937   }{
3938     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
3939     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
3940   }
3941 }

```

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

\inlineass inline:

```

3942 \NewDocumentCommand \inlineass { m } {
3943   \beginngroup
3944   \stex_ref_new_doc_target:n{
3945     #1
3946   \endgroup
3947 }

```

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

## 33.3 Examples

sexample

```

3948
3949 \keys_define:nn {stex / sexample }{
3950   type      .str_set_x:N = \exampletype,
3951   id        .str_set_x:N = \sexampleid,
3952   title     .tl_set:N     = \sexampletitle,
3953   for       .clist_set:N  = \sexamplefor,
3954 }
3955 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
3956   \str_clear:N \sexampletype
3957   \str_clear:N \sexampleid
3958   \tl_clear:N \sexampletitle
3959   \clist_clear:N \sexamplefor
3960   \keys_set:nn { stex / sexample }{ #1 }
3961 }
3962
3963 \NewDocumentEnvironment{sexample}{0{}}{
3964   \__stex_statements_sexample_args:n{ #1 }
3965   \stex_smsmode_set_codes:
3966   \clist_set:Nn \l_tmpa_clist \sexampletype
3967   \tl_clear:N \l_tmpa_tl
3968   \clist_map_inline:Nn \l_tmpa_clist {
3969     \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
3970       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
3971     }
3972   }
3973   \stex_if_smsmode:F {
3974     \seq_clear:N \l_tmpa_seq

```



```

3975 \clist_map_inline:Nn \sexamplefor {
3976 \str_if_eq:nnF{ ##1 }{}{
3977 \stex_get_symbol:n { ##1 }
3978 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
3979 \l_stex_get_symbol_uri_str
3980 }
3981 }
3982 }
3983 \exp_args:Nnnx
3984 \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
3985 \str_if_empty:NF \sexamplotype {
3986 \stex_annotate_invisible:nnn{type}{\sexampletype}{}}
3987 }
3988 }
3989 \tl_if_empty:NTF \l_tmpa_tl {
3990 \__stex_statements_sexample_start:
3991 }{
3992 \l_tmpa_tl
3993 }
3994 \stex_ref_new_doc_target:n \sexampleid
3995 }{
3996 \clist_set:Nn \l_tmpa_clist \sexamplotype
3997 \tl_clear:N \l_tmpa_tl
3998 \clist_map_inline:Nn \l_tmpa_clist {
3999 \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
4000 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
4001 }
4002 }
4003 \tl_if_empty:NTF \l_tmpa_tl {
4004 \__stex_statements_sexample_end:
4005 }{
4006 \l_tmpa_tl
4007 }
4008 \stex_if_smsmode:F {
4009 \end{stex_annotate_env}
4010 }
4011 }

```

\stexpatchexample

```

4012
4013 \cs_new_protected:Nn \__stex_statements_sexample_start: {
4014 \par\noindent\titl{Example~\tl_if_empty:NF \sexamplotype {
4015 (\sexamplotype)
4016 }~}
4017 }
4018 \cs_new_protected:Nn \__stex_statements_sexample_end: {\par\medskip}
4019
4020 \newcommand\stexpatchexample[3]{} {
4021 \str_set:Nx \l_tmpa_str{ #1 }
4022 \str_if_empty:NTF \l_tmpa_str {
4023 \tl_set:Nn \__stex_statements_sexample_start: { #2 }
4024 \tl_set:Nn \__stex_statements_sexample_end: { #3 }
4025 }{
4026 \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }

```

```

4027     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
4028   }
4029 }

```

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

`\inlineex` inline:

```

4030 \NewDocumentCommand \inlineex { m } {
4031   \begingroup
4032   \stex_ref_new_doc_target:n{
4033     #1
4034   \endgroup
4035 }

```

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

## 33.4 Logical Paragraphs

`sparagraph`

```

4036 \keys_define:nn { stex / sparagraph } {
4037   id      .str_set:x:N = \sparagraphid ,
4038   title   .tl_set:N    = \l_stex_sparagraph_title_tl ,
4039   type    .str_set:x:N = \sparagraphtype ,
4040   for     .str_set:x:N = \sparagraphfor ,
4041   from    .tl_set:x:N  = \sparagraphfrom ,
4042   start   .tl_set:N    = \l_stex_sparagraph_start_tl ,
4043   name    .str_set:N   = \sparagraphname
4044 }
4045
4046 \cs_new_protected:Nn \stex_sparagraph_args:n {
4047   \tl_clear:N \l_stex_sparagraph_title_tl
4048   \tl_clear:N \sparagraphfrom
4049   \tl_clear:N \l_stex_sparagraph_start_tl
4050   \str_clear:N \sparagraphid
4051   \str_clear:N \sparagraphtype
4052   \str_clear:N \sparagraphfor
4053   \str_clear:N \sparagraphname
4054   \keys_set:nn { stex / sparagraph }{ #1 }
4055 }
4056 \newif\if@in@omtext\@in@omtextfalse
4057
4058 \NewDocumentEnvironment {sparagraph} { 0{} } {
4059   \stex_sparagraph_args:n { #1 }
4060   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4061     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
4062   }{
4063     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
4064   }
4065   \@in@omtexttrue
4066   \stex_smsmode_set_codes:
4067   \clist_set:No \l_tmpa_clist \sparagraphtype
4068   \tl_clear:N \l_tmpa_tl
4069   \clist_map_inline:Nn \l_tmpa_clist {

```

```

4070 \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
4071 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
4072 }
4073 }
4074 \stex_if_smsmode:F {
4075 \exp_args:Nnnx
4076 \begin{stex_annotate_env}{paragraph}{}
4077 \str_if_empty:NF \sparagraphtype {
4078 \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}
4079 }
4080 }
4081 \tl_if_empty:NTF \l_tmpa_tl {
4082 \__stex_statements_sparagraph_start:
4083 }{
4084 \l_tmpa_tl
4085 }
4086 \stex_ref_new_doc_target:n \sparagraphid
4087 \ignorespacesandpars
4088 }{
4089 \clist_set:No \l_tmpa_clist \sparagraphtype
4090 \tl_clear:N \l_tmpa_tl
4091 \clist_map_inline:Nn \l_tmpa_clist {
4092 \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
4093 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
4094 }
4095 }
4096 \str_if_empty:NF \sparagraphname { \symdecl*{\sparagraphname} }
4097 \tl_if_empty:NTF \l_tmpa_tl {
4098 \__stex_statements_sparagraph_end:
4099 }{
4100 \l_tmpa_tl
4101 }
4102 \stex_if_smsmode:F {
4103 \end{stex_annotate_env}
4104 }
4105 }

```

# \stexpatchparagraph

```

4106
4107 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
4108 \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4109 \tl_if_empty:NF \l_stex_sparagraph_title_tl {
4110 \titleemph{\l_stex_sparagraph_title_tl}:~
4111 }
4112 }{
4113 \titleemph{\l_stex_sparagraph_start_tl}~
4114 }
4115 }
4116 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
4117
4118 \newcommand\stexpatchparagraph[3]{} {
4119 \str_set:Nx \l_tmpa_str{ #1 }
4120 \str_if_empty:NTF \l_tmpa_str {
4121 \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }

```

```

4122     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
4123   }{
4124     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
4125     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
4126   }
4127 }

```

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

symboldoc

```

4128 \NewDocumentEnvironment{symboldoc}{ m }{
4129   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
4130   \seq_clear:N \l_tmpb_seq
4131   \seq_map_inline:Nn \l_tmpa_seq {
4132     \str_if_eq:nnF{ ##1 }{}{
4133       \stex_get_symbol:n { ##1 }
4134       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
4135         \l_stex_get_symbol_uri_str
4136       }
4137     }
4138   }
4139   \par
4140   \exp_args:Nnnx
4141   \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
4142 }{
4143   \end{stex_annotate_env}
4144 }
4145 \</package>

```

# Chapter 34

## The Implementation

### 34.1 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>13</sup>

```
4146 <*package>
4147 <@@=stex_sproof>
4148
4149 %%%%%%%%%%% sproof.dtx %%%%%%%%%%%
4150
```

### 34.2 Proofs

We first define some keys for the proof environment.

```
4151 \keys_define:nn { stex / spf } {
4152   id          .str_set:N = \l__stex_sproof_spf_id_str,
4153   display     .tl_set:N  = \l__stex_sproof_spf_display_tl,
4154   for         .tl_set:N  = \l__stex_sproof_spf_for_tl ,
4155   from        .tl_set:N  = \l__stex_sproof_spf_from_tl ,
4156   proofend    .tl_set:N  = \l__stex_sproof_spf_proofend_tl,
4157   type        .tl_set:N  = \l__stex_sproof_spf_type_tl,
4158   title       .tl_set:N  = \l__stex_sproof_spf_title_tl,
4159   continues   .tl_set:N  = \l__stex_sproof_spf_continues_tl,
4160   functions   .tl_set:N  = \l__stex_sproof_spf_functions_tl,
4161   method      .tl_set:N  = \l__stex_sproof_spf_method_tl
4162 }
4163 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
4164   \str_clear:N \l__stex_sproof_spf_id_str
4165   \tl_clear:N \l__stex_sproof_spf_display_tl
4166   \tl_clear:N \l__stex_sproof_spf_for_tl
4167   \tl_clear:N \l__stex_sproof_spf_from_tl
4168   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
4169   \tl_clear:N \l__stex_sproof_spf_type_tl
4170   \tl_clear:N \l__stex_sproof_spf_title_tl

```

---

<sup>13</sup>EDNOTE: need an implementation for L<sup>A</sup>T<sub>E</sub>X<sub>M</sub>L

```

4171 \tl_clear:N \l__stex_sproof_spf_continues_tl
4172 \tl_clear:N \l__stex_sproof_spf_functions_tl
4173 \tl_clear:N \l__stex_sproof_spf_method_tl
4174 \keys_set:nn { stex / spf }{ #1 }
4175 }

```

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

```

4176 \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.

```

4177 \newcount\count_ten
4178 \newenvironment{pst@with@label}[1]{
4179   \edef\pst@label{#1}
4180   \advance\count_ten by 1\relax
4181   \count_ten=1
4182 }{
4183   \advance\count_ten by -1\relax
4184 }

```

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

```

4185 \def\the@pst@label{
4186   \pst@make@label\pst@label{\number\count_ten}\l__stex_sproof_pstlabel_postfix_tl
4187 }

```

(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.

```

4188 \keys_define:nn { stex / pstlabel }{
4189   prefix      .tl_set:N   = \l__stex_sproof_pstlabel_prefix_tl,
4190   delimiter   .tl_set:N   = \l__stex_sproof_pstlabel_delimiter_tl,
4191   postfix     .tl_set:N   = \l__stex_sproof_pstlabel_postfix_tl
4192 }
4193 \cs_new_protected:Nn \__stex_sproof_pstlabel_args:n {

```

---

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

```

4194 \tl_set:Nn \l__stex_sproof_pstlabel_prefix_tl {P}
4195 \tl_set:Nn \l__stex_sproof_pstlabel_delimiter_tl {.}
4196 \tl_clear:N \l__stex_sproof_pstlabel_postfix_tl
4197 }
4198 \__stex_sproof_pstlabel_args:n {}
4199 \newcommand\setpstlabelstyle[1]{
4200   \__stex_sproof_pstlabel_args:n {#1}
4201 }
4202 \newcommand\setpstlabelstyledefault{%
4203   \__stex_sproof_pstlabel_args:n{prefix=P,delimiter=.,postfix={}}
4204 }

```

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

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

```

4205 \ExplSyntaxOff
4206 \def\pst@make@label@long#1#2{\@for\@I:=#1\do{\expandafter\expandafter\expandafter\@I\csname
4207 \def\pst@make@label@angles#1#2{\ensuremath{\@for\@I:=#1\do{\rangle}}#2}
4208 \def\pst@make@label@short#1#2{#2}
4209 \def\pst@make@label@empty#1#2{}
4210 \ExplSyntaxOn
4211 \def\pstlabelstyle#1{%
4212   \def\pst@make@label{\use:c{pst@make@label@#1}}%
4213 }%
4214 \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.

```

4215 \def\next@pst@label{%
4216   \global\advance\count\count10 by 1%
4217 }%

```

(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

```

4218 \def\sproof@box{
4219   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
4220 }
4221 \def\spf@proofend{\sproof@box}
4222 \def\sproofend{
4223   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
4224     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
4225   }
4226 }
4227 \def\sProofEndSymbol#1{\def\sproof@box{#1}}

```

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

**spf@\*@kw**

```

4228 \def\spf@proofsketch@kw{Proof Sketch}
4229 \def\spf@proof@kw{Proof}
4230 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

4231 \cs_if_exist:NT \bbl@loaded {
4232   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4233   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4234     \input{proof-ngerman.ldf}
4235   }
4236   \clist_if_in:NnT \l_tmpa_clist {finnish}{
4237     \input{proof-finnish.ldf}
4238   }
4239   \clist_if_in:NnT \l_tmpa_clist {french}{
4240     \input{proof-french.ldf}
4241   }
4242   \clist_if_in:NnT \l_tmpa_clist {russian}{
4243     \input{proof-russian.ldf}
4244   }
4245 }
4246

```

**spfsketch**

```

4247 \newcommand\spfsketch[2][]{
4248   \__stex_sproof_spf_args:n{#1}
4249   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4250     \titleemph{
4251       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
4252         \spf@proofsketch@kw
4253       }{
4254         \l__stex_sproof_spf_type_tl
4255       }
4256     }:
4257   }
4258   {-#2}
4259   %\sref@label@id{this \ifx\spf@type\@empty\spf@proofsketch@kw\else\spf@type\fi}
4260   \sproofend
4261 }

```

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

**spfeq** This is very similar to `\spfsketch`, but uses a computation array<sup>1415</sup>

```

4262 \newenvironment{spfeq}[2][]{
4263   \__stex_sproof_spf_args:n{#1}
4264   %\sref@target
4265   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4266     \titleemph{
4267       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
4268         \spf@proof@kw
4269       }{
4270         \l__stex_sproof_spf_type_tl
4271       }
4272     }:

```

<sup>14</sup>EDNOTE: This should really be more like a tabular with an ensuremath in it. or invoke text on the last column

<sup>15</sup>EDNOTE: document above



```

4273 }
4274 {~#2}
4275 \begin{displaymath}\begin{array}{rcll}
4276 }{
4277 \end{array}\end{displaymath}
4278 }

```

(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.

```

4279 \newenvironment{spf@proof}[2][]{
4280 \__stex_sproof_spf_args:n{#1}
4281 %\sref@target
4282 \count_ten=10
4283 \par\noindent
4284 \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4285 \titleemph{
4286 \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {
4287 \spf@proof@kw
4288 }{
4289 \l__stex_sproof_spf_type_tl
4290 }
4291 }:
4292 }
4293 {~#2}
4294 %\sref@label@id{this \ifx\spf@type\empty\spf@proof@kw\else\spf@type\fi}
4295 \def\pst@label{}
4296 \newcount\pst@count% initialize the labeling mechanism
4297 \begin{description}\begin{pst@with@label}{\l__stex_sproof_pstlabel_prefix_tl}
4298 }{
4299 \end{pst@with@label}\end{description}
4300 }
4301 \newenvironment{sproof}[2][{\begin{spf@proof}[#1][#2]}{\sproofend\end{spf@proof}}
4302 \newenvironment{sProof}[2][{\begin{spf@proof}[#1][#2]}{\end{spf@proof}}

```

**\spfidea**

```

4303 \newcommand\spfidea[2][]{
4304 \__stex_sproof_spf_args:n{#1}
4305 \titleemph{
4306 \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {Proof~Idea}{
4307 \l__stex_sproof_spf_type_tl
4308 }:
4309 }~#2
4310 \sproofend
4311 }

```

(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 16

```

4312 \newenvironment{spfstep}[1][]{
4313   \_stex_sproof_spf_args:n{#1}
4314   \@in@omtexttrue
4315   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4316     \item[\the@pst@label]
4317   }
4318   \tl_if_empty:NF \l__stex_sproof_spf_title_tl {
4319     {(\titleemph{\l__stex_sproof_spf_title_tl})\enspace}
4320   }
4321   %\sref@label@id{\pst@label}
4322   \ignorespacesandpars
4323 }{
4324   \next@pst@label\ignorespacesandpars
4325 }

```

sproofcomment

```

4326 \newenvironment{sproofcomment}[1][]{
4327   \_stex_sproof_spf_args:n{#1}
4328   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4329     \item[\the@pst@label]
4330   }
4331 }{
4332   \next@pst@label
4333 }

```

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.

```

4334 \newenvironment{subproof}[2][]{
4335   \_stex_sproof_spf_args:n{#1}
4336   \def\@test{#2}
4337   \ifx\@test\empty\else
4338     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4339       \item[\the@pst@label]
4340     }{#2}
4341   \fi
4342   \begin{pst@with@label}{\pst@label,\number\count_ten}
4343 }{
4344   \end{pst@with@label}\next@pst@label
4345 }

```

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

```

4346 \newenvironment{spfcases}[2][]{
4347   \def\@test{#1}
4348   \ifx\@test\empty
4349     \begin{subproof}[method=by-cases]{#2}
4350   \else
4351     \begin{subproof}[#1,method=by-cases]{#2}
4352   \fi
4353 }{

```

---

<sup>16</sup>EdNOTE: MK: labeling of steps does not work yet.

```

4354 \end{subproof}
4355 }

```

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

```

4356 \newenvironment{spfcase}[2] []{
4357   \__stex_sproof_spf_args:n{#1}
4358   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4359     \item[\the@pst@label]
4360   }
4361   \def\@test{#2}
4362   \ifx\@test\@empty
4363   \else
4364     {\titleemph{#2}:~}
4365   \fi
4366   \begin{pst@with@label}{\pst@label,\number\count_ten}
4367 }{
4368   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4369     \sproofend
4370   }
4371   \end{pst@with@label}
4372   \next@pst@label
4373 }

```

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

```

4374 \newcommand\spfcasesketch[3] []{
4375   \__stex_sproof_spf_args:n{#1}
4376   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4377     \item[\the@pst@label]
4378   }
4379   \def\@test{#2}
4380   \ifx\@test\@empty
4381   \else
4382     {\titleemph{#2}:~}
4383   \fi#3
4384   \next@pst@label
4385 }%

```

### 34.3 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.

```

4386 \keys_define:nn { stex / just }{
4387   id          .str_set:x:N = \l__stex_sproof_just_id_str,
4388   method      .tl_set:N   = \l__stex_sproof_just_method_tl,
4389   premises    .tl_set:N   = \l__stex_sproof_just_premises_tl,
4390   args        .tl_set:N   = \l__stex_sproof_just_args_tl
4391 }

```

The next three environments and macros are purely semantic, so we ignore the keyval arguments for now and only display the content.<sup>17</sup>

<sup>17</sup>EdNOTE: need to do something about the premise in draft mode.

**justification**

```
4392 \newenvironment{justification}[1] [] {}{}
```

**\premise**

```
4393 \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.

```
4394 \newcommand\justarg[2] [] {#2}
```

```
4395 \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 35

# STEX -Others Implementation

```
4396 <*package>
4397
4398 %%%%%%%%%% others.dtx %%%%%%%%%%
4399
4400 <@@=stex_others>
      Warnings and error messages
4401 % None

\MSC Math subject classifier

4402 \NewDocumentCommand \MSC {m} {
4403   % TODO
4404 }

(End definition for \MSC. This function is documented on page 21.)
      Patching tikzinput, if loaded

4405 \@ifpackageloaded{tikzinput}{
4406   \RequirePackage{stex-tikzinput}
4407 }{}
4408 </package>
```

## Chapter 36

# STEX -Metatheory Implementation

```
4409 <*package>
4410 <@@=stex_modules>
4411
4412 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
4413
4414 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
4415 \begingroup
4416 \stex_module_setup:nn{
4417   ns=\c_stex_metatheory_ns_str,
4418   meta=NONE
4419 }{Metatheory}
4420 \stex_reactivate_macro:N \symdecl
4421 \stex_reactivate_macro:N \notation
4422 \stex_reactivate_macro:N \symdef
4423 \ExplSyntaxOff
4424 \csname stex_suppress_html:n\endcsname{
4425   % is-a (a:A, a \in A, a is an A, etc.)
4426   \symdecl[args=ai]{isa}
4427   \notation[typed]{isa}{#1 \comp{:} #2}{#1 \comp, #2}
4428   \notation[in]{isa}{#1 \comp\in #2}{#1 \comp, #2}
4429   \notation[pred]{isa}{#2\comp(#1 \comp)}{#1 \comp, #2}
4430
4431   % bind (\forall, \Pi, \lambda etc.)
4432   \symdecl[args=Bi]{bind}
4433   \notation[forall]{bind}{\comp\forall #1.\;#2}{#1 \comp, #2}
4434   \notation[\Pi]{bind}{\comp\prod_{#1}#2}{#1 \comp, #2}
4435   \notation[deffun]{bind}{\comp( #1 \comp)\; \to\;}{#1 \comp, #2}
4436
4437   % dummy variable
4438   \symdecl{dummyvar}
4439   \notation[underscore]{dummyvar}{\comp\_}
4440   \notation[dot]{dummyvar}{\comp\cdot}
4441   \notation[dash]{dummyvar}{\comp{\rm --}}
4442
4443   %fromto (function space, Hom-set, implication etc.)
```

```

4444 \symdecl[args=ai]{fromto}
4445 \notation[xarrow]{fromto}{#1 \comp\to #2}{#1 \comp\times #2}
4446 \notation[arrow]{fromto}{#1 \comp\to #2}{#1 \comp\to #2}
4447
4448 % mapto (lambda etc.)
4449 %\symdecl[args=Bi]{mapto}
4450 %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
4451 %\notation[lambda]{mapto}{\comp\lambda #1 \comp. \; #2}{#1 \comp, #2}
4452 %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp. \; #2}{#1 \comp, #2}
4453
4454 % function/operator application
4455 \symdecl[args=ia]{apply}
4456 \notation[prec=0;0x\infpres,parens]{apply}{#1 \comp( #2 \comp)}{#1 \comp, #2}
4457 \notation[prec=0;0x\infpres,lambda]{apply}{#1 \; #2 }{#1 \; #2}
4458
4459 % ‘‘type’’ of all collections (sets, classes, types, kinds)
4460 \symdecl{collection}
4461 \notation[U]{collection}{\comp{\mathcal{U}}}
4462 \notation[set]{collection}{\comp{\textsf{Set}}}
4463
4464 % sequences
4465 \symdecl[args=1]{seqtype}
4466 \notation[kleene]{seqtype}{#1^{\comp\ast}}
4467
4468 \symdef[args=2,li,prec=nobrackets]{sequence-index}{#1_{#2}}
4469 \notation[ui,prec=nobrackets]{sequence-index}{#1^{#2}}
4470
4471 %\symdef[args=3,li]{sequence-from-to}{#1_{#2}\comp{\,\ellipses\,}#1_{#3}}
4472 %\notation[ui]{sequence-from-to}{#1^{#2}\comp{\,\ellipses\,}#1^{#3}}
4473 % ^ superceded by \aseqfromto and \livar/\uivar
4474
4475 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{\,\ellipses\,}}{#1\comp,#2}
4476 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{\,\ellipses\,}#2}{#1\comp,#2}
4477 \symdef[args=aui,prec=nobrackets]{aseqfromtovia}{#1\comp{\,\ellipses\,}#2\comp{\,\ellipses\,}}{#1\comp,#2}
4478
4479 % letin (‘‘let’’, local definitions, variable substitution)
4480 \symdecl[args=bii]{letin}
4481 \notation[let]{letin}{\comp{\rm let}}{\;#1\comp{=}\;#2\; \comp{\rm in}}{\;#3}
4482 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
4483 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
4484
4485 % structures
4486 \symdecl*[args=1]{module-type}
4487 \notation{module-type}{\mathtt{MOD} #1}
4488 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
4489 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{#1 \comp, #2}
4490
4491 }
4492 \ExplSyntaxOn
4493 \stex_add_to_current_module:n{
4494   \let\nappa\apply
4495   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
4496   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
4497   \def\livar{\csname sequence-index\endcsname[li]}

```

```

4498 \def\uivar{\csname sequence-index\endcsname[ui]}
4499 \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
4500 \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
4501 \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
4502 }
4503 \__stex_modules_end_module:
4504 \endgroup
4505 \</package>

```



## Chapter 37

# Tikzinput Implementation

```
4506 <*package>
4507
4508 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
4509
4510 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
4511 \RequirePackage{l3keys2e}
4512
4513 \keys_define:nn { tikzinput } {
4514   image .bool_set:N = \c_tikzinput_image_bool,
4515   image .default:n = false ,
4516   unknown .code:n = {}
4517 }
4518
4519 \ProcessKeysOptions { tikzinput }
4520
4521 \bool_if:NTF \c_tikzinput_image_bool {
4522   \RequirePackage{graphicx}
4523
4524   \providecommand\usetikzlibrary[]{}
4525   \newcommand\tikzinput[2] []{\includegraphics[#1]{#2}}
4526 }{
4527   \RequirePackage{tikz}
4528   \RequirePackage{standalone}
4529
4530   \newcommand \tikzinput [2] [] {
4531     \setkeys{Gin}{#1}
4532     \ifx \Gin@ewidth \Gin@exclamation
4533       \ifx \Gin@eheight \Gin@exclamation
4534         \input { #2 }
4535       \else
4536         \resizebox{!}{ \Gin@eheight }{
4537           \input { #2 }
4538         }
4539       \fi
4540     \else
4541       \ifx \Gin@eheight \Gin@exclamation
4542         \resizebox{ \Gin@ewidth }{!}{
4543           \input { #2 }
```

```

4544     }
4545     \else
4546     \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
4547     \input { #2 }
4548     }
4549     \fi
4550   \fi
4551 }
4552 }
4553
4554 \newcommand \ctikzinput [2] [] {
4555   \begin{center}
4556     \tikzinput [1] {#2}
4557   \end{center}
4558 }
4559
4560 \@ifpackageloaded{stex}{
4561   \RequirePackage{stex-tikzinput}
4562 }{}
4563
4564 </package>
4565 <*stex>
4566 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
4567 \RequirePackage{stex}
4568 \RequirePackage{tikzinput}
4569
4570 \newcommand\mhtikzinput [2] [] {%
4571   \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
4572   \stex_in_repository:nn\Gin@mhrepos{
4573     \tikzinput [1]{\mhpath{##1}{#2}}
4574   }
4575 }
4576 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [1] {#2}\end{center}}
4577 </stex>

```

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

## Chapter 38

# document-structure.sty Implementation

### 38.1 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.

```
4578 \*cls)
4579 \<@@=document_structure>
4580 \ProvidesExplClass{omdoc}{2020/10/19}{1.4}{OMDoc Documents}
4581 \RequirePackage{l3keys2e,expl-keystr-compat}
```

### 38.2 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`

```
4582 \keys_define:nn{ document-structure / pkg }{
4583   class      .str_set_x:N = \c_document_structure_class_str,
4584   minimal    .bool_set:N = \c_document_structure_minimal_bool,
4585   report     .code:n      = {
4586     \ClassWarning{omdoc}{the option 'report' is deprecated, use 'class=report', instead}
4587     \str_set:Nn \c_document_structure_class_str {report}
4588   },
4589   book       .code:n      = {
4590     \ClassWarning{omdoc}{the option 'book' is deprecated, use 'class=book', instead}
4591     \str_set:Nn \c_document_structure_class_str {book}
4592   },
4593   bookpart   .code:n      = {
4594     \ClassWarning{omdoc}{the option 'bookpart' is deprecated, use 'class=book,topsect=chapter}
4595     \str_set:Nn \c_document_structure_class_str {book}
4596     \str_set:Nn \c_document_structure_topsect_str {chapter}
4597   },
```

```

4598 docopt      .str_set_x:N = \c_document_structure_docopt_str,
4599 unknown     .code:n      = {
4600   \PassOptionsToPackage{ \CurrentOption }{ omdoc }
4601 }
4602 }
4603 \ProcessKeysOptions{ document-structure / pkg }
4604 \str_if_empty:NT \c_document_structure_class_str {
4605   \str_set:Nn \c_document_structure_class_str {article}
4606 }
4607 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
4608   {\c_document_structure_class_str}
4609

```

### 38.3 Beefing up the document environment

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

```

4610 \RequirePackage{omdoc}
4611 \bool_if:NF \c_document_structure_minimal_bool {
4612   \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>18</sup>

```

4613 \keys_define:nn { document-structure / document }{
4614   id .str_set_x:N = \c_document_structure_document_id_str
4615 }
4616 \let\__document_structure_orig_document=\document
4617 \renewcommand{\document}[1][]{
4618   \keys_set:nn{ document-structure / document }{ #1 }
4619   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
4620   \__document_structure_orig_document
4621 }

```

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

```

4622 }
4623 \</cls>

```

### 38.4 Implementation: OMDoc Package

```

4624 \*package>
4625 \ProvidesExplPackage{omdoc}{2020/10/19}{1.4}{OMDoc document Structure}
4626 \RequirePackage{expl-keystr-compat,13keys2e}

```

### 38.5 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>18</sup>EdNOTE: faking documentkeys for now. @HANG, please implement

```

4627
4628 \keys_define:nn{ document-structure / pkg }{
4629   class      .str_set_x:N = \c_document_structure_class_str,
4630   topsect    .str_set_x:N = \c_document_structure_topsect_str,
4631   % showignores .bool_set:N = \c_document_structure_showignores_bool,
4632 }
4633 \ProcessKeysOptions{ document-structure / pkg }
4634 \str_if_empty:NT \c_document_structure_class_str {
4635   \str_set:Nn \c_document_structure_class_str {article}
4636 }
4637 \str_if_empty:NT \c_document_structure_topsect_str {
4638   \str_set:Nn \c_document_structure_topsect_str {section}
4639 }

```

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

```

4640 \RequirePackage{xspace}
4641 \RequirePackage{comment}
4642 \@ifpackageloaded{babel}{\RequirePackage[base]{babel}}

```

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

```

4643 \@ifpackageloaded{babel}{
4644   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4645   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4646     \input{omdoc-ngerman.ldf}
4647   }
4648 }{}
4649 %\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.

```

4650 \int_new:N \l_document_structure_section_level_int
4651 \str_case:VnF \c_document_structure_topsect_str {
4652   {part}{
4653     \int_set:Nn \l_document_structure_section_level_int {0}
4654   }
4655   {chapter}{
4656     \int_set:Nn \l_document_structure_section_level_int {1}
4657   }
4658 }{
4659   \str_case:VnF \c_document_structure_class_str {
4660     {book}{
4661       \int_set:Nn \l_document_structure_section_level_int {0}
4662     }
4663     {report}{
4664       \int_set:Nn \l_document_structure_section_level_int {0}
4665     }
4666   }{
4667     \int_set:Nn \l_document_structure_section_level_int {2}
4668   }
4669 }

```

## 38.6 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>19</sup>

EdN:19

```
4670 \def\current@section@level{document}%
4671 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
4672 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
4673 \cs_new_protected:Npn \skipomgroup {
4674   \ifcase\l_document_structure_section_level_int
4675   \or\stepcounter{part}
4676   \or\stepcounter{chapter}
4677   \or\stepcounter{section}
4678   \or\stepcounter{subsection}
4679   \or\stepcounter{subsubsection}
4680   \or\stepcounter{paragraph}
4681   \or\stepcounter{subparagraph}
4682   \fi
4683 }
```

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

`blindomgroup`

```
4684 \newcommand\at@begin@blindomgroup[1]{%
4685 \newenvironment{blindomgroup}
4686 {
4687   \int_incr:N\l_document_structure_section_level_int
4688   \at@begin@blindomgroup\l_document_structure_section_level_int
4689 }{}}
```

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

```
4690 \newcommand\omgroup@nonum[2]{
4691   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
4692   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
4693 }
```

*(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.

```
4694 \newcommand\omgroup@num[2]{
```

<sup>19</sup>EDNOTE: MK: we may have to experiment with the more powerful uppercasing macro from `mfirstuc.sty` once we internationalize.

```

4695 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
4696   \@nameuse{#1}{#2}
4697 }{
4698   \cs_if_exist:NTF\rdfmata@sectioning{
4699     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
4700   }{
4701     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
4702   }
4703 }
4704 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
4705 }

```

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

omgroup

```

4706 \keys_define:nn { document-structure / omgroup }{
4707   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
4708   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
4709   creators     .clist_set:N = \l__document_structure_omgroup_creators_clist,
4710   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
4711   srccite      .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
4712   type         .tl_set:N    = \l__document_structure_omgroup_type_tl,
4713   short        .tl_set:N    = \l__document_structure_omgroup_short_tl,
4714   display      .tl_set:N    = \l__document_structure_omgroup_display_tl,
4715   intro        .tl_set:N    = \l__document_structure_omgroup_intro_tl,
4716   loadmodules  .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
4717 }
4718 \cs_new_protected:Nn \l__document_structure_omgroup_args:n {
4719   \str_clear:N \l__document_structure_omgroup_id_str
4720   \str_clear:N \l__document_structure_omgroup_date_str
4721   \clist_clear:N \l__document_structure_omgroup_creators_clist
4722   \clist_clear:N \l__document_structure_omgroup_contributors_clist
4723   \tl_clear:N \l__document_structure_omgroup_srccite_tl
4724   \tl_clear:N \l__document_structure_omgroup_type_tl
4725   \tl_clear:N \l__document_structure_omgroup_short_tl
4726   \tl_clear:N \l__document_structure_omgroup_display_tl
4727   \tl_clear:N \l__document_structure_omgroup_intro_tl
4728   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
4729   \keys_set:nn { document-structure / omgroup } { #1 }
4730 }

```

we define a switch for numbering lines and a hook for the beginning of groups: The \at@begin@omgroup macro allows customization. It is run at the beginning of the omgroup, i.e. after the section heading.

```

4731 \newif\if@mainmatter\@mainmattertrue
4732 \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.

```

4733 \keys_define:nn { document-structure / sectioning }{
4734   name .str_set_x:N = \l__document_structure_sect_name_str ,
4735   ref   .str_set_x:N = \l__document_structure_sect_ref_str  ,
4736   clear .bool_set:N  = \l__document_structure_sect_clear_bool ,
4737   num   .bool_set:N  = \l__document_structure_sect_num_bool  ,
4738 }

```

```

4739 \cs_new_protected:Nn \__document_structure_sect_args:n {
4740   \str_clear:N \l__document_structure_sect_name_str
4741   \str_clear:N \l__document_structure_sect_ref_str
4742   \bool_set_false:N \l__document_structure_sect_clear_bool
4743   \bool_set_false:N \l__document_structure_sect_num_bool
4744   \keys_set:nn { document-structure / sectioning } { #1 }
4745 }
4746 \newcommand\omdoc@sectioning[3][]{
4747   \__document_structure_sect_args:n {#1}
4748   \let\omdoc@sect@name\l__document_structure_sect_name_str
4749   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
4750   \if@mainmatter% numbering not overridden by frontmatter, etc.
4751     \bool_if:NTF \l__document_structure_sect_num_bool {
4752       \omgroup@num{#2}{#3}
4753     }{
4754       \omgroup@nonum{#2}{#3}
4755     }
4756     \def\current@section@level{\omdoc@sect@name}
4757   \else
4758     \omgroup@nonum{#2}{#3}
4759   \fi
4760 }% 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.

```

4761 \newcommand\omgroup@redefine@addtocontents[1]{%
4762   %\edef\__document_structureimport{#1}%
4763   %\@for\@I:=\__document_structureimport\do{%
4764     %\edef\@path{\csname module@\@I @path\endcsname}%
4765     %\@ifundefined{tf@toc}\relax%
4766     % {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
4767   %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
4768   %\def\addcontentsline##1##2##3{%
4769     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
4770   %\else% hyperref.sty not loaded
4771   %\def\addcontentsline##1##2##3{%
4772     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
4773   %\fi
4774 }% 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.

```

4775 \int_new:N \l_document_structure_omgroup_level_int
4776 \newenvironment{omgroup}[2][]{% keys, title
4777 {
4778   \__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.

```

4779 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
4780   \omgroup@redefine@addtocontents{
4781     %\@ifundefined{module@id}\used@modules%
4782     %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}

```



```

4783     }
4784 }

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

4785 \int_incr:N \l_document_structure_omgroup_level_int
4786 \int_incr:N \l_document_structure_section_level_int
4787 \ifcase\l_document_structure_section_level_int
4788   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
4789   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
4790   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
4791   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
4792   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
4793   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
4794   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
4795 \fi
4796 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
4797 \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
4798 }% for customization
4799 {}

```

and finally, we localize the sections

```

4800 \newcommand\omdoc@part@kw{Part}
4801 \newcommand\omdoc@chapter@kw{Chapter}
4802 \newcommand\omdoc@section@kw{Section}
4803 \newcommand\omdoc@subsection@kw{Subsection}
4804 \newcommand\omdoc@subsubsection@kw{Subsubsubsection}
4805 \newcommand\omdoc@paragraph@kw{paragraph}
4806 \newcommand\omdoc@subparagraph@kw{subparagraph}

```

## 38.7 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`

```

4807 \providecommand\printindex{\IfFileExists{\jobname.ind}{\input{\jobname.ind}}{}}

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

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

4808 \cs_if_exist:NTF\frontmatter{
4809   \let\__document_structure_orig_frontmatter\frontmatter
4810   \let\frontmatter\relax
4811 }{
4812   \tl_set:Nn\__document_structure_orig_frontmatter{
4813     \clearpage
4814     \@mainmatterfalse
4815     \pagenumbering{roman}
4816   }
4817 }
4818 \cs_if_exist:NTF\backmatter{

```

```

4819 \let\__document_structure_orig_backmatter\backmatter
4820 \let\backmatter\relax
4821 }{
4822 \tl_set:Nn\__document_structure_orig_backmatter{
4823 \clearpage
4824 \@mainmatterfalse
4825 \pagenumbering{roman}
4826 }
4827 }

```

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.

```

4828 \newenvironment{frontmatter}{
4829 \__document_structure_orig_frontmatter
4830 }{
4831 \cs_if_exist:NTF\mainmatter{
4832 \mainmatter
4833 }{
4834 \clearpage
4835 \@mainmattertrue
4836 \pagenumbering{arabic}
4837 }
4838 }

```

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

```

4839 \newenvironment{backmatter}{
4840 \__document_structure_orig_backmatter
4841 }{
4842 \cs_if_exist:NTF\mainmatter{
4843 \mainmatter
4844 }{
4845 \clearpage
4846 \@mainmattertrue
4847 \pagenumbering{arabic}
4848 }
4849 }

```

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

```

4850 \@mainmattertrue\pagenumbering{arabic}

```

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

```

4851 \def \c__document_structure_document_str{document}
4852 \newcommand\afterprematurestop{}
4853 \def\prematurestop@endomgroup{
4854 \unless\ifx\@currenvir\c__document_structure_document_str
4855 \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
4856 \expandafter\prematurestop@endomgroup
4857 \fi
4858 }
4859 \providecommand\prematurestop{

```

```

4860 \message{Stopping~sTeX~processing~prematurely}
4861 \prematurestop@endomgroup
4862 \afterprematurestop
4863 \end{document}
4864 }

```

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

## 38.8 Global Variables

**\setSGvar** set a global variable

```

4865 \RequirePackage{etoolbox}
4866 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

**\useSGvar** use a global variable

```

4867 \newrobustcmd\useSGvar[1]{%
4868 \@ifundefined{sTeX@Gvar@#1}
4869 {\PackageError{omdoc}
4870 {The sTeX Global variable #1 is undefined}
4871 {set it with \protect\setSGvar}}
4872 \@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.

```

4873 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
4874 \@ifundefined{sTeX@Gvar@#1}
4875 {\PackageError{omdoc}
4876 {The sTeX Global variable #1 is undefined}
4877 {set it with \protect\setSGvar}}
4878 {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}

```

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

## Chapter 39

# MiKoSlides – Implementation

### 39.1 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.

```
4879 \*cls)
4880 \@@=mikoslides)
4881 \ProvidesExplClass{mikoslides}{2020/12/06}{1.3}{MiKo slides Class}
4882 \RequirePackage{l3keys2e,expl-keystr-compat}
4883
4884 \keys_define:nn{mikoslides / cls}{
4885   class .code:n = {
4886     \PassOptionsToClass{\CurrentOption}{omdoc}
4887     \str_if_eq:nnT{#1}{book}{
4888       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
4889     }
4890     \str_if_eq:nnT{#1}{report}{
4891       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
4892     }
4893   },
4894   notes .bool_set:N = \c__mikoslides_notes_bool ,
4895   slides .code:n = { \bool_set_false:N \c__mikoslides_notes_bool },
4896   unknown .code:n = {
4897     \PassOptionsToClass{\CurrentOption}{omdoc}
4898     \PassOptionsToClass{\CurrentOption}{beamer}
4899     \PassOptionsToPackage{\CurrentOption}{mikoslides}
4900   }
4901 }
4902 \ProcessKeysOptions{ mikoslides / cls }
4903 \bool_if:NTF \c__mikoslides_notes_bool {
4904   \PassOptionsToPackage{notes=true}{mikoslides}
4905 }{
4906   \PassOptionsToPackage{notes=false}{mikoslides}
4907 }
4908 \</cls)
```

now we do the same for the mikoslides package.

```

4909 <*package>
4910 \ProvidesExplPackage{mikoslides}{2020/12/06}{1.3}{MiKo slides Package}
4911 \RequirePackage{l3keys2e,expl-keystr-compat}
4912
4913 \keys_define:nn{mikoslides / pkg}{
4914   topsect          .str_set_x:N = \c__mikoslides_topsect_str,
4915   defaulttopsect   .str_set_x:N = \c__mikoslides_defaulttopsec_str,
4916   notes            .bool_set:N = \c__mikoslides_notes_bool ,
4917   slides            .code:n      = { \bool_set_false:N \c__mikoslides_notes_bool },
4918   sectocframes      .bool_set:N = \c__mikoslides_sectocframes_bool ,
4919   frameimages       .bool_set:N = \c__mikoslides_frameimages_bool ,
4920   fiboxed           .bool_set:N = \c__mikoslides_fiboxed_bool ,
4921   noproblems        .bool_set:N = \c__mikoslides_noproblems_bool,
4922   unknown           .code:n      = {
4923     \PassOptionsToClass{\CurrentOption}{stex}
4924     \PassOptionsToClass{\CurrentOption}{tikzinput}
4925   }
4926 }
4927 \ProcessKeysOptions{ mikoslides / pkg }
4928 \newif\ifnotes
4929 \bool_if:NTF \c__mikoslides_notes_bool {
4930   \notesttrue
4931 }{
4932   \notesfalse
4933 }
4934

```

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

```

4935 \str_if_empty:NTF \c__mikoslides_topsect_str {
4936   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_defaulttopsec_str
4937 }{
4938   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_topsect_str
4939 }
4940 </package>

```

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

```

4941 <*cls>
4942 \bool_if:NTF \c__mikoslides_notes_bool {
4943   \LoadClass{omdoc}
4944 }{
4945   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
4946   \newcounter{Item}
4947   \newcounter{paragraph}
4948   \newcounter{subparagraph}
4949   \newcounter{Hfootnote}
4950   \RequirePackage{omdoc}
4951 }

```

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

```

4952 \RequirePackage{mikoslides}
4953 </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.

```

4954 \*package>
4955 \bool_if:NT \c__mikoslides_notes_bool {
4956   \RequirePackage{a4wide}
4957   \RequirePackage{marginnote}
4958   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
4959   \RequirePackage{mdframed}
4960   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
4961   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
4962 }
4963 \RequirePackage{stex-compatibility}
4964 \RequirePackage{stex-tikzinput}
4965 \RequirePackage{etoolbox}
4966 \RequirePackage{amssymb}
4967 \RequirePackage{amsmath}
4968 \RequirePackage{comment}
4969 \RequirePackage{textcomp}
4970 \RequirePackage{url}
4971 \RequirePackage{graphicx}
4972 \RequirePackage{pgf}

```

## 39.2 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>20</sup>

```

4973 \bool_if:NT \c__mikoslides_notes_bool {
4974   \renewcommand\usetheme[2] [] {\usepackage[#1]{beamernotestheme#2}}
4975 }

```

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

```

4976 \newcounter{slide}
4977 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
4978 \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.

```

4979 \bool_if:NTF \c__mikoslides_notes_bool {
4980   \renewenvironment{note}{\ignorespaces}{\ignorespaces}{}
4981 }{
4982   \excludecomment{note}
4983 }

```

---

<sup>20</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.

```
4984 \bool_if:NT \c__mikoslides_notes_bool {
4985   \newlength{\slideframewidth}
4986   \setlength{\slideframewidth}{1.5pt}
```

**frame** We first define the keys.

```
4987 \cs_new_protected:Nn \__mikoslides_do_yes_param:Nn {
4988   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
4989     \bool_set_true:N #1
4990   }{
4991     \bool_set_false:N #1
4992   }
4993 }
4994 \keys_define:nn{mikoslides / frame}{
4995   label .str_set_x:N = \l__mikoslides_frame_label_str,
4996   allowframebreaks .code:n = {
4997     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowframebreaks_bool { #1 }
4998   },
4999   allowdisplaybreaks .code:n = {
5000     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowdisplaybreaks_bool { #1 }
5001   },
5002   fragile .code:n = {
5003     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_fragile_bool { #1 }
5004   },
5005   shrink .code:n = {
5006     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_shrink_bool { #1 }
5007   },
5008   squeeze .code:n = {
5009     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_squeeze_bool { #1 }
5010   },
5011   t .code:n = {
5012     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_t_bool { #1 }
5013   },
5014 }
5015 \cs_new_protected:Nn \__mikoslides_frame_args:n {
5016   \str_clear:N \l__mikoslides_frame_label_str
5017   \bool_set_true:N \l__mikoslides_frame_allowframebreaks_bool
5018   \bool_set_true:N \l__mikoslides_frame_allowdisplaybreaks_bool
5019   \bool_set_true:N \l__mikoslides_frame_fragile_bool
5020   \bool_set_true:N \l__mikoslides_frame_shrink_bool
5021   \bool_set_true:N \l__mikoslides_frame_squeeze_bool
5022   \bool_set_true:N \l__mikoslides_frame_t_bool
5023   \keys_set:nn { mikoslides / frame }{ #1 }
5024 }
```

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

```
5025 \renewenvironment{frame}[1][]{
5026   \__mikoslides_frame_args:n{#1}
5027   \sffamily
5028   \stepcounter{slide}
5029   \def\@currentlabel{\theslide}
5030   \str_if_empty:NF \l__mikoslides_frame_label_str {
5031     \label{\l__mikoslides_frame_label_str}
```

5032 }

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

```

5033 \def\itemize@level{outer}
5034 \def\itemize@outer{outer}
5035 \def\itemize@inner{inner}
5036 \renewcommand\newpage{\addtocounter{framenum}{1}}
5037 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
5038 \renewenvironment{itemize}{
5039   \ifx\itemize@level\itemize@outer
5040     \def\itemize@label{$\rhd$}
5041   \fi
5042   \ifx\itemize@level\itemize@inner
5043     \def\itemize@label{$\scriptstyle\rhd$}
5044   \fi
5045   \begin{list}
5046   {\itemize@label}
5047   {\setlength{\labelsep}{.3em}
5048    \setlength{\labelwidth}{.5em}
5049    \setlength{\leftmargin}{1.5em}
5050   }
5051   \edef\itemize@level{\itemize@inner}
5052 }{
5053   \end{list}
5054 }

```

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

```

5055 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth]
5056 }{
5057   \medskip\miko@slidelabel\end{mdframed}
5058 }

```

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

`\frametitle`

```

5059 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
5060 }

```

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

EdN:21

`\pause` 21

```

5061 \bool_if:NT \c__mikoslides_notes_bool {
5062   \newcommand\pause{}
5063 }

```

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

`nomtext`

```

5064 \bool_if:NTF \c__mikoslides_notes_bool {
5065   \newenvironment{nomtext}[1][\begin{sparagraph}[#1]}{\end{sparagraph}}
5066 }{
5067   \excludecomment{nomtext}
5068 }

```

---

<sup>21</sup>EdNOTE: MK: fake it in notes mode for now



nomgroup

```
5069 \bool_if:NTF \c__mikoslides_notes_bool {
5070   \newenvironment{nomgroup}[2] [] {\begin{omgroup}[#1]{#2}}{\end{omgroup}}
5071 }{
5072   \excludecomment{nomgroup}
5073 }
```

ndefinition

```
5074 \bool_if:NTF \c__mikoslides_notes_bool {
5075   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}{\end{sdefinition}}
5076 }{
5077   \excludecomment{ndefinition}
5078 }
```

nassertion

```
5079 \bool_if:NTF \c__mikoslides_notes_bool {
5080   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}{\end{sassertion}}
5081 }{
5082   \excludecomment{nassertion}
5083 }
```

nsproof

```
5084 \bool_if:NTF \c__mikoslides_notes_bool {
5085   \newenvironment{nproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
5086 }{
5087   \excludecomment{nproof}
5088 }
```

nexample

```
5089 \bool_if:NTF \c__mikoslides_notes_bool {
5090   \newenvironment{nexample}[1] [] {\begin{example}[#1]}{\end{example}}
5091 }{
5092   \excludecomment{nexample}
5093 }
```

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

```
5094 \def\inputref@preskip{\smallskip}
5095 \def\inputref@postskip{\medskip}
```

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

\inputref\*

```
5096 \let\orig@inputref\inputref
5097 \def\inputref{\@ifstar\ninputref\orig@inputref}
5098 \newcommand\ninputref[2] [] {
5099   \bool_if:NT \c__mikoslides_notes_bool {
5100     \orig@inputref[#1]{#2}
5101   }
5102 }
```

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

### 39.3 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>}`.

```

5103 \newlength{\slidelogoheight}
5104
5105 \bool_if:NTF \c__mikoslides_notes_bool {
5106   \setlength{\slidelogoheight}{.4cm}
5107 }{
5108   \setlength{\slidelogoheight}{1cm}
5109 }
5110 \newsavebox{\slidelogo}
5111 \sbox{\slidelogo}{\text{\TeX}}
5112 \newrobustcmd{\setslidelogo}[1]{
5113   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
5114 }
```

(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.

```

5115 \def\source{Michael Kohlhase}% customize locally
5116 \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.

```

5117 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
5118 \newsavebox{\cclogo}
5119 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
5120 \newif\ifcchref\cchreffalse
5121 \AtBeginDocument{
5122   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
5123 }
5124 \def\licensing{
5125   \ifcchref
5126     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
5127   \else
5128     {\usebox{\cclogo}}
5129   \fi
5130 }
5131 \newrobustcmd{\setlicensing}[2][]{
5132   \def\@url{#1}
5133   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
5134   \ifx\@url\@empty
5135     \def\licensing{{\usebox{\cclogo}}}
5136   \else
5137     \def\licensing{
```

```

5138     \ifcchref
5139     \href{#1}{\usebox{\cclogo}}
5140     \else
5141     {\usebox{\cclogo}}
5142     \fi
5143   }
5144 \fi
5145 }

```

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

EdN:22

`\slidelabel` Now, we set up the slide label for the article mode.<sup>22</sup>

```

5146 \newrobustcmd\miko@slidelabel{
5147   \vbox to \slidelogoheight{
5148     \vss\hbox to \slidewidth
5149     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
5150   }
5151 }

```

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

## 39.4 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.

```

5152 \def\Gin@mhrepos{}
5153 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
5154 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
5155 \newrobustcmd\frameimage[2][{}{
5156   \stepcounter{slide}
5157   \bool_if:NT \c__mikoslides_frameimages_bool {
5158     \def\Gin@ewidth{}\setkeys{Gin}{#1}
5159     \bool_if:NF \c__mikoslides_notes_bool { \vfill }
5160     \begin{center}
5161       \bool_if:NTF \c__mikoslides_fiboxed_bool {
5162         \fbox{
5163           \ifx\Gin@ewidth\@empty
5164             \ifx\Gin@mhrepos\@empty
5165               \mhgraphics[width=\slidewidth,#1]{#2}
5166             \else
5167               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5168             \fi
5169           \else% Gin@ewidth empty
5170             \ifx\Gin@mhrepos\@empty
5171               \mhgraphics[#1]{#2}
5172             \else
5173               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5174             \fi
5175           \fi% Gin@ewidth empty
5176         }
5177       }{
5178         \ifx\Gin@ewidth\@empty

```

---

<sup>22</sup>EdNOTE: see that we can use the themes for the slides some day. This is all fake.

```

5179         \ifx\Gin@mhrepos\empty
5180             \mhgraphics[width=\slidewidth,#1]{#2}
5181         \else
5182             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5183         \fi
5184         \ifx\Gin@mhrepos\empty
5185             \mhgraphics[#1]{#2}
5186         \else
5187             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5188         \fi
5189     \fi% Gin@ewidth empty
5190 }
5191 \end{center}
5192 \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
5193 \bool_if:NF \c__mikoslides_notes_bool { \vfill }
5194 }
5195 } % ifmks@sty@frameimages

```

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

## 39.5 Colors and Highlighting

We first specify sans serif fonts as the default.

```

5196 \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

```

5197 \AddToHook{begindocument}{
5198     \definecolor{green}{rgb}{0,.5,0}
5199     \definecolor{purple}{cmyk}{.3,1,0,.17}
5200 }

```

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`.

```

5201 % \def\STpresent#1{\textcolor{blue}{#1}}
5202 \def\defemph#1{\textcolor{magenta}{#1}}
5203 \def\symrefemph#1{\textcolor{cyan}{#1}}
5204 \def\compemph#1{\textcolor{blue}{#1}}
5205 \def\titleemph#1{\textcolor{blue}{#1}}
5206 \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.

```

5207 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
5208 \def\smalltextwarning{
5209     \pgfuseimage{miko@small@dbend}
5210     \xspace
5211 }
5212 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}

```

```

5213 \newrobustcmd\textwarning{
5214   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
5215   \xspace
5216 }
5217 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}
5218 \newrobustcmd\bigtextwarning{
5219   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
5220   \xspace
5221 }

```

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

```

5222 \newrobustcmd\putgraphicsat[3]{
5223   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
5224 }
5225 \newrobustcmd\putat[2]{
5226   \begin{picture}(0,0)\put(#1){#2}\end{picture}
5227 }

```

## 39.6 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`.

```

5228 \bool_if:NT \c__mikoslides_sectocframes_bool {
5229   \str_if_eq:VnTF \__mikoslidestopsect{part}{
5230     \newcounter{chapter}\counterwithin*{section}{chapter}
5231   }{
5232     \str_if_eq:VnT\__mikoslidestopsect{chapter}{
5233       \newcounter{chapter}\counterwithin*{section}{chapter}
5234     }
5235   }
5236 }

```

`\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
5237 \def\part@prefix{}
5238 \@ifpackageloaded{omdoc}{}{
5239   \str_case:VnF \__mikoslidestopsect {
5240     {part}{
5241       \int_set:Nn \l_document_structure_section_level_int {0}
5242       \def\thesection{\arabic{chapter}.\arabic{section}}
5243       \def\part@prefix{\arabic{chapter}.}
5244     }
5245     {chapter}{
5246       \int_set:Nn \l_document_structure_section_level_int {1}
5247       \def\thesection{\arabic{chapter}.\arabic{section}}
5248       \def\part@prefix{\arabic{chapter}.}
5249     }
5250   }{
5251     \int_set:Nn \l_document_structure_section_level_int {2}
5252     \def\part@prefix{}

```

```

5253 }
5254 }
5255
5256 \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

```

5257 \renewenvironment{omgroup}[2][]{
5258   \__document_structure_omgroup_args:n { #1 }
5259   \int_incr:N \l_document_structure_omgroup_level_int
5260   \int_incr:N \l_document_structure_section_level_int
5261   \bool_if:NT \c__mikoslides_sectocframes_bool {
5262     \stepcounter{slide}
5263     \begin{frame}[noframenumbering]
5264     \vfill\Large\centering
5265     \red{
5266       \ifcase\l_document_structure_section_level_int\or
5267         \stepcounter{part}
5268         \def\__mikoslideslabel{\omdoc@part@kw~\Roman{part}}
5269         \def\currentsectionlevel{\omdoc@part@kw}
5270       \or
5271         \stepcounter{chapter}
5272         \def\__mikoslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
5273         \def\currentsectionlevel{\omdoc@chapter@kw}
5274       \or
5275         \stepcounter{section}
5276         \def\__mikoslideslabel{\part@prefix\arabic{section}}
5277         \def\currentsectionlevel{\omdoc@section@kw}
5278       \or
5279         \stepcounter{subsection}
5280         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
5281         \def\currentsectionlevel{\omdoc@subsection@kw}
5282       \or
5283         \stepcounter{subsubsection}
5284         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
5285         \def\currentsectionlevel{\omdoc@subsubsection@kw}
5286       \or
5287         \stepcounter{paragraph}
5288         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{paragraph}}
5289         \def\currentsectionlevel{\omdoc@paragraph@kw}
5290       \else
5291         \def\__mikoslideslabel{}
5292         \def\currentsectionlevel{\omdoc@paragraph@kw}
5293       \fi% end ifcase
5294       \__mikoslideslabel%\sref@label@id\__mikoslideslabel
5295       \quad #2%
5296     }%
5297     \vfill%
5298     \end{frame}%
5299   }
5300   \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str%

```

```

5301 }{}
5302 }

```

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

```

5303 \def\inserttheorembodyfont{\normalfont}
5304 %\bool_if:NF \c__mikoslides_notes_bool {
5305 % \defbeamertemplate{theorem begin}{miko}
5306 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
5307 % \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
5308 % \inserttheorempunctuation\inserttheorembodyfont\hspace}
5309 % \defbeamertemplate{theorem end}{miko}{}}

```

and we set it as the default one.

```

5310 % \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.

```

5311 % \expandafter\def\csname Parent2\endcsname{}
5312 %}
5313
5314 \AddToHook{begindocument}{ % this does not work for some reasons
5315 \setbeamertemplate{theorems}[ams style]
5316 }
5317 \bool_if:NT \c__mikoslides_notes_bool {
5318 \renewenvironment{columns}[1][]{%
5319 \par\noindent%
5320 \begin{minipage}%
5321 \slidewidth\centering\leavevmode%
5322 }{%
5323 \end{minipage}\par\noindent%
5324 }%
5325 \newsavebox\columnbox%
5326 \renewenvironment<>{column}[2][]{%
5327 \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
5328 }{%
5329 \end{minipage}\end{lrbox}\usebox\columnbox%
5330 }%
5331 }
5332 \bool_if:NTF \c__mikoslides_noproblems_bool {
5333 \newenvironment{problems}{}{}
5334 }{
5335 \excludecomment{problems}
5336 }

```

## 39.7 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.

```

5337 \gdef\printexcursions{}
5338 \newcommand\excursionref[2]{% label, text
5339 \bool_if:NT \c__mikoslides_notes_bool {

```

```

5340 \begin{sparagraph}[title=Excursion]
5341 #2 \sref[fallback=the appendix]{#1}.
5342 \end{sparagraph}
5343 }
5344 }
5345 \newcommand\activate@excursion[2][]{
5346 \gappto\printexcursions{\inputref[#1]{#2}}
5347 }
5348 \newcommand\excursion[4][]{% repos, label, path, text
5349 \bool_if:NT \c__mikoslides_notes_bool {
5350 \activate@excursion[#1]{#3}\excursionref{#2}{#4}
5351 }
5352 }

```

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

\excursiongroup

```

5353 \keys_define:nn{mikoslides / excursiongroup }{
5354 id .str_set_x:N = \l__mikoslides_excursion_id_str,
5355 intro .tl_set:N = \l__mikoslides_excursion_intro_tl,
5356 mhrepos .str_set_x:N = \l__mikoslides_excursion_mhrepos_str
5357 }
5358 \cs_new_protected:Nn \__mikoslides_excursion_args:n {
5359 \tl_clear:N \l__mikoslides_excursion_intro_tl
5360 \str_clear:N \l__mikoslides_excursion_id_str
5361 \str_clear:N \l__mikoslides_excursion_mhrepos_str
5362 \keys_set:nn {mikoslides / excursiongroup }{ #1 }
5363 }
5364 \newcommand\excursiongroup[1][]{
5365 \__mikoslides_excursion_args:n{ #1 }
5366 \ifdefempty\printexcursions{}% only if there are excursions
5367 {\begin{note}
5368 \begin{omgroup}[#1]{Excursions}%
5369 \ifdefempty\l__mikoslides_excursion_intro_tl{{
5370 \inputref[\l__mikoslides_excursion_mhrepos_str]{
5371 \l__mikoslides_excursion_intro_tl
5372 }
5373 }
5374 \printexcursions%
5375 \end{omgroup}
5376 \end{note}}
5377 }
5378 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
5379 \end{package}

```

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



## Chapter 40

# The Implementation

### 40.1 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.

```
5380 <*package>
5381 <@@=problems>
5382 \ProvidesExplPackage{problem}{2019/03/20}{1.3}{Semantic Markup for Problems}
5383 \RequirePackage{l3keys2e,expl-keystr-compatible}
5384
5385 \keys_define:nn { problem / pkg }{
5386   notes      .default:n    = { true },
5387   notes      .bool_set:N   = \c__problems_notes_bool,
5388   gnotes     .default:n    = { true },
5389   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
5390   hints      .default:n    = { true },
5391   hints      .bool_set:N   = \c__problems_hints_bool,
5392   solutions  .default:n    = { true },
5393   solutions  .bool_set:N   = \c__problems_solutions_bool,
5394   pts        .default:n    = { true },
5395   pts        .bool_set:N   = \c__problems_pts_bool,
5396   min        .default:n    = { true },
5397   min        .bool_set:N   = \c__problems_min_bool,
5398   boxed      .default:n    = { true },
5399   boxed      .bool_set:N   = \c__problems_boxed_bool,
5400   unknown    .code:n       = {}
5401 }
5402 \def\solutionstrue{
5403   \bool_set_true:N \c__problems_solutions_bool
5404 }
5405 \def\solutionsfalse{
5406   \bool_set_false:N \c__problems_solutions_bool
5407 }
5408
5409 \ProcessKeysOptions{ problem / pkg }
```

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

```

5410 \RequirePackage{stex-compatibility}
5411 \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.

```

5412 \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.

```

5413 \def\prob@problem@kw{Problem}
5414 \def\prob@solution@kw{Solution}
5415 \def\prob@hint@kw{Hint}
5416 \def\prob@note@kw{Note}
5417 \def\prob@gnote@kw{Grading}
5418 \def\prob@pt@kw{pt}
5419 \def\prob@min@kw{min}

```

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

For the other languages, we set up triggers

```

5420 \@ifpackageloaded{babel}{
5421   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5422   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5423     \input{problem-ngerman.ldf}
5424   }
5425   \clist_if_in:NnT \l_tmpa_clist {finnish}{
5426     \input{problem-finnish.ldf}
5427   }
5428   \clist_if_in:NnT \l_tmpa_clist {french}{
5429     \input{problem-french.ldf}
5430   }
5431   \clist_if_in:NnT \l_tmpa_clist {russian}{
5432     \input{problem-russian.ldf}
5433   }
5434 }{}

```

## 40.2 Problems and Solutions

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

```

5435 \keys_define:nn{ problem / problem }{
5436   id      .str_set:x:N = \l__problems_prob_id_str,
5437   pts     .tl_set:N    = \l__problems_prob_pts_tl,
5438   min     .tl_set:N    = \l__problems_prob_min_tl,
5439   title   .tl_set:N    = \l__problems_prob_title_tl,
5440   refnum  .int_set:N   = \l__problems_prob_refnum_int
5441 }
5442 \cs_new_protected:Nn \__problems_prob_args:n {
5443   \str_clear:N \l__problems_prob_id_str
5444   \tl_clear:N \l__problems_prob_pts_tl
5445   \tl_clear:N \l__problems_prob_min_tl
5446   \tl_clear:N \l__problems_prob_title_tl

```

```

5447 \int_zero_new:N \l__problems_prob_refnum_int
5448 \keys_set:nn { problem / problem }{ #1 }
5449 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
5450   \let\l__problems_inclprob_refnum_int\undefined
5451 }
5452 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

5453 \newcounter{problem}
5454 \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.

```

5455 \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

```

5456 \newcommand\prob@number{
5457   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
5458     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
5459   }{
5460     \int_if_exist:NTF \l__problems_prob_refnum_int {
5461       \prob@label{\int_use:N \l__problems_prob_refnum_int }
5462     }{
5463       \prob@label\theproblem
5464     }
5465   }
5466 }

```

(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.

```

5467 \newcommand\prob@title[3]{%
5468   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
5469     #2 \l__problems_inclprob_title_tl #3
5470   }{
5471     \tl_if_exist:NTF \l__problems_prob_title_tl {
5472       #2 \l__problems_prob_title_tl #3
5473     }{
5474       #1
5475     }
5476   }
5477 }

```

(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.

```

5478 \def\prob@heading{
5479   \prob@problem@kw~\prob@number\prob@title{~}{~}{~}\strut}
5480   %\sref@label@id{\prob@problem@kw~\prob@number}{~}
5481 }

```

(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`

```

5482 \newenvironment{problem}[1][1]{
5483   \__problems_prob_args:n{#1}%\sref@target%
5484   \@in@omtexttrue% we are in a statement (for inline definitions)
5485   \stepcounter{problem}\record@problem
5486   \def\current@section@level{\prob@problem@kw}
5487   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
5488 }%
5489   {\smallskip}
5490   \bool_if:NT \c__problems_boxed_bool {
5491     \surroundwithmdframed{problem}
5492   }

```

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

```

5493 \def\record@problem{
5494   \protected@write\@auxout{}
5495   {
5496     \string\@problem{\prob@number}
5497     {
5498       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
5499         \l__problems_inclprob_pts_tl
5500       }{
5501         \l__problems_prob_pts_tl
5502       }
5503     }%
5504     {
5505       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
5506         \l__problems_inclprob_min_tl
5507       }{
5508         \l__problems_prob_min_tl
5509       }
5510     }
5511   }
5512 }

```

(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).

```

5513 \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.

```

5514 \keys_define:nn { problem / solution }{
5515   id          .str_set_x:N = \l__problems_solution_id_str ,
5516   for         .tl_set:N    = \l__problems_solution_for_tl ,
5517   height      .dim_set:N   = \l__problems_solution_height_dim ,
5518   creators    .clist_set:N = \l__problems_solution_creators_clist ,
5519   contributors .clist_set:N = \l__problems_solution_contributors_clist ,
5520   srccite     .tl_set:N    = \l__problems_solution_srccite_tl
5521 }
5522 \cs_new_protected:Nn \__problems_solution_args:n {
5523   \str_clear:N \l__problems_solution_id_str
5524   \tl_clear:N \l__problems_solution_for_tl
5525   \tl_clear:N \l__problems_solution_srccite_tl
5526   \clist_clear:N \l__problems_solution_creators_clist
5527   \clist_clear:N \l__problems_solution_contributors_clist
5528   \dim_zero:N \l__problems_solution_height_dim
5529   \keys_set:nn { problem / solution }{ #1 }
5530 }

```

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

```

5531 \newcommand\@startsolution[1][ ]{
5532   \__problems_solution_args:n { #1 }
5533   \@in@omtexttrue% we are in a statement.
5534   \bool_if:NF \c__problems_boxed_bool { \hrule }
5535   \smallskip\noindent
5536   {\textbf\prob@solution@kw : \enspace}
5537   \begin{small}
5538   \def\current@section@level{\prob@solution@kw}
5539   \ignorespacesandpars
5540 }

```

**\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.

```

5541 \newcommand\startsolutions{
5542   \specialcomment{solution}{\@startsolution}{
5543     \bool_if:NF \c__problems_boxed_bool {
5544       \hrule\medskip
5545     }
5546     \end{small}%
5547   }
5548   \bool_if:NT \c__problems_boxed_bool {
5549     \surroundwithmdframed{solution}
5550   }
5551 }

```

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

**\stopsolutions**

```

5552 \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.

```

5553 \bool_if:NTF \c__problems_solutions_bool {
5554   \startsolutions
5555 }{
5556   \stopsolutions
5557 }

```

**exnote**

```

5558 \bool_if:NTF \c__problems_notes_bool {
5559   \newenvironment{exnote}[1][]{
5560     \par\smallskip\hrule\smallskip
5561     \noindent\textbf{\prob@note@kw : }\small
5562   }{
5563     \smallskip\hrule
5564   }
5565 }{
5566   \excludecomment{exnote}
5567 }

```

**hint**

```

5568 \bool_if:NTF \c__problems_notes_bool {
5569   \newenvironment{hint}[1][]{
5570     \par\smallskip\hrule\smallskip
5571     \noindent\textbf{\prob@hint@kw :~ }\small
5572   }{
5573     \smallskip\hrule
5574   }
5575   \newenvironment{exhint}[1][]{
5576     \par\smallskip\hrule\smallskip
5577     \noindent\textbf{\prob@hint@kw :~ }\small
5578   }{
5579     \smallskip\hrule
5580   }
5581 }{
5582   \excludecomment{hint}
5583   \excludecomment{exhint}
5584 }

```

**gnote**

```

5585 \bool_if:NTF \c__problems_notes_bool {
5586   \newenvironment{gnote}[1][]{
5587     \par\smallskip\hrule\smallskip
5588     \noindent\textbf{\prob@gnote@kw : }\small
5589   }{
5590     \smallskip\hrule
5591   }
5592 }{
5593   \excludecomment{gnote}
5594 }

```

## 40.3 Multiple Choice Blocks

EdN:23

mcb 23

```
5595 \newenvironment{mcb}{
5596   \begin{enumerate}
5597 }{
5598   \end{enumerate}
5599 }
```

we define the keys for the mcc macro

```
5600 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
5601   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
5602     \bool_set_true:N #1
5603   }{
5604     \bool_set_false:N #1
5605   }
5606 }
5607 \keys_define:nn { problem / mcc }{
5608   id          .str_set_x:N = \l__problems_mcc_id_str ,
5609   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
5610   T           .default:n   = { true } ,
5611   T           .bool_set:N   = \l__problems_mcc_t_bool ,
5612   F           .default:n   = { true } ,
5613   F           .bool_set:N   = \l__problems_mcc_f_bool ,
5614   Ttext       .code:n      = {
5615     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
5616   } ,
5617   Ftext       .code:n      = {
5618     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
5619   }
5620 }
5621 \cs_new_protected:Nn \l__problems_mcc_args:n {
5622   \str_clear:N \l__problems_mcc_id_str
5623   \tl_clear:N \l__problems_mcc_feedback_tl
5624   \bool_set_true:N \l__problems_mcc_t_bool
5625   \bool_set_true:N \l__problems_mcc_f_bool
5626   \bool_set_true:N \l__problems_mcc_Ttext_bool
5627   \bool_set_false:N \l__problems_mcc_Ftext_bool
5628   \keys_set:nn { problem / mcc }{ #1 }
5629 }
```

\mcc

```
5630 \newcommand\mcc[2][]{
5631   \l__problems_mcc_args:n{ #1 }
5632   \item #2
5633   \bool_if:NT \c__problems_solutions_bool {
5634     \
5635     \bool_if:NT \l__problems_mcc_t_bool {
5636       % TODO!
5637       % \ifcsstring{mcc@T}{T}{\mcc@Ttext}%
5638     }
5639     \bool_if:NT \l__problems_mcc_f_bool {
```

---

<sup>23</sup>EdNOTE: MK: maybe import something better here from a dedicated MC package

```

5640      % TODO!
5641      % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
5642    }
5643    \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
5644      !
5645    }{
5646      \l__problems_mcc_feedback_tl
5647    }
5648  }
5649 } %solutions

```

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

## 40.4 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.

```

5650
5651 \keys_define:nn{ problem / inclproblem }{
5652   % id      .str_set_x:N = \l__problems_inclprob_id_str,
5653   pts      .tl_set:N    = \l__problems_inclprob_pts_tl,
5654   min      .tl_set:N    = \l__problems_inclprob_min_tl,
5655   title    .tl_set:N    = \l__problems_inclprob_title_tl,
5656   refnum   .int_set:N    = \l__problems_inclprob_refnum_int,
5657   mhrepos  .str_set_x:N = \l__problems_inclprob_mhrepos_str
5658 }
5659 \cs_new_protected:Nn \__problems_inclprob_args:n {
5660   % \str_clear:N \l__problems_prob_id_str
5661   \tl_clear:N \l__problems_inclprob_pts_tl
5662   \tl_clear:N \l__problems_inclprob_min_tl
5663   \tl_clear:N \l__problems_inclprob_title_tl
5664   \int_zero_new:N \l__problems_inclprob_refnum_int
5665   \str_clear:N \l__problems_inclprob_mhrepos_str
5666   \keys_set:nn { problem / inclproblem }{ #1 }
5667   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
5668     \let\l__problems_inclprob_pts_tl\undefined
5669   }
5670   \tl_if_empty:NT \l__problems_inclprob_min_tl {
5671     \let\l__problems_inclprob_min_tl\undefined
5672   }
5673   \tl_if_empty:NT \l__problems_inclprob_title_tl {
5674     \let\l__problems_inclprob_title_tl\undefined
5675   }
5676   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
5677     \let\l__problems_inclprob_refnum_int\undefined
5678   }
5679 }
5680
5681 \cs_new_protected:Nn \__problems_inclprob_clear: {
5682   % \str_clear:N \l__problems_prob_id_str
5683   \let\l__problems_inclprob_pts_tl\undefined
5684   \let\l__problems_inclprob_min_tl\undefined

```



```

5685 \let\l__problems_inclprob_title_tl\undefined
5686 \let\l__problems_inclprob_refnum_int\undefined
5687 \let\l__problems_inclprob_mhrepos_str\undefined
5688 }
5689
5690 \newcommand\includeproblem[2][]{
5691   \__problems_inclprob_args:n{ #1 }
5692   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
5693     \input{#2}
5694   }{
5695     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
5696       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
5697     }
5698   }
5699   \__problems_inclprob_clear:
5700 }

```

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

## 40.5 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.

```

5701 \AddToHook{enddocument}{
5702   \bool_if:NT \c__problems_pts_bool {
5703     \message{Total:~\arabic{pts}~points}
5704   }
5705   \bool_if:NT \c__problems_min_bool {
5706     \message{Total:~\arabic{min}~minutes}
5707   }
5708 }

```

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

```

5709 \def\pts#1{
5710   \bool_if:NT \c__problems_pts_bool {
5711     \marginpar{#1~\prob@pt@kw}
5712   }
5713 }
5714 \def\min#1{
5715   \bool_if:NT \c__problems_min_bool {
5716     \marginpar{#1~\prob@min@kw}
5717   }
5718 }

```

**\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.

```

5719 \newcounter{pts}
5720 \def\show@pts{
5721   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
5722     \bool_if:NT \c__problems_pts_bool {
5723       \marginpar{\l__problems_inclprob_pts_tl;\prob@pt@kw\smallskip}
5724       \addtocounter{pts}{\l__problems_inclprob_pts_tl}

```

```

5725     }
5726   }{
5727     \tl_if_exist:NT \l__problems_prob_pts_tl {
5728       \bool_if:NT \c__problems_pts_bool {
5729         \marginpar{\l__problems_prob_pts_tl;\prob@pt@kw\smallskip}
5730         \addtocounter{pts}{\l__problems_prob_pts_tl}
5731       }
5732     }
5733   }
5734 }

```

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

\show@min

```

5735 \newcounter{min}
5736 \def\show@min{
5737   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
5738     \bool_if:NT \c__problems_min_bool {
5739       \marginpar{\l__problems_inclprob_pts_tl;min}
5740       \addtocounter{min}{\l__problems_inclprob_min_tl}
5741     }
5742   }{
5743     \tl_if_exist:NT \l__problems_prob_min_tl {
5744       \bool_if:NT \c__problems_min_bool {
5745         \marginpar{\l__problems_prob_min_tl;min}
5746         \addtocounter{min}{\l__problems_prob_min_tl}
5747       }
5748     }
5749   }
5750 }
5751 \</package>

```

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

## Chapter 41

# 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.

### 41.1 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.

```
5752 <@@=hwexam>
5753 <*cls>
5754 \ProvidesExplClass{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
5755 \RequirePackage{l3keys2e,expl-keystr-compatible}
5756 \DeclareOption*{
5757   \PassOptionsToClass{\CurrentOption}{omdoc}
5758   \PassOptionsToPackage{\CurrentOption}{stex}
5759   \PassOptionsToPackage{\CurrentOption}{hwexam}
5760   \PassOptionsToPackage{\CurrentOption}{tikzinput}
5761 }
5762 \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.

```
5763 \LoadClass{omdoc}
5764 \RequirePackage{stex}
5765 \RequirePackage{hwexam}
5766 \RequirePackage{tikzinput}
5767 \RequirePackage{graphicx}
5768 \RequirePackage{a4wide}
5769 \RequirePackage{amssymb}
5770 \RequirePackage{amstext}
5771 \RequirePackage{amsmath}
```

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

```

5772 \newcommand\assig@default@type{\hwexam@assignment@kw}
5773 \def\document@hwexamtype{\assig@default@type}
5774 <@@=document_structure>
5775 \keys_define:nn { document-structure / document }{
5776 id .str_set_x:N = \c_document_structure_document_id_str,
5777 hwexamtype .tl_set:N = \document@hwexamtype
5778 }
5779 <@@=hwexam>
5780 </cls>

```

## Chapter 42

# Implementation: The hwexam Package

### 42.1 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.

```
5781 \*package>
5782 \ProvidesExplPackage{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
5783 \RequirePackage{l3keys2e,expl-keystr-compat}
5784
5785 \newif\iftest\testfalse
5786 \DeclareOption{test}{\testtrue}
5787 \newif\ifmultiple\multiplefalse
5788 \DeclareOption{multiple}{\multipletrue}
5789 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
5790 \ProcessOptions
```

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

```
5791 \RequirePackage{keyval}[1997/11/10]
5792 \RequirePackage{problem}
```

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

```
5793 \newcommand\hwexam@assignment@kw{Assignment}
5794 \newcommand\hwexam@given@kw{Given}
5795 \newcommand\hwexam@due@kw{Due}
5796 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
5797 blank~for~extra~space}%
5798 \newcommand\correction@probs@kw{prob.}%
5799 \newcommand\correction@pts@kw{total}%
5800 \newcommand\correction@reached@kw{reached}%
5801 \newcommand\correction@sum@kw{Sum}%
5802 \newcommand\correction@grade@kw{grade}%
5803 \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

```

5804 \ifpackageloaded{babel}{\RequirePackage[base]{babel}}
5805
5806 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5807 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5808   \input{hwexam-ngerman.ldf}
5809 }
5810 \clist_if_in:NnT \l_tmpa_clist {finnish}{
5811   \input{hwexam-finnish.ldf}
5812 }
5813 \clist_if_in:NnT \l_tmpa_clist {french}{
5814   \input{hwexam-french.ldf}
5815 }
5816 \clist_if_in:NnT \l_tmpa_clist {russian}{
5817   \input{hwexam-russian.ldf}
5818 }

```

## 42.2 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.

```

5819 \newcounter{assignment}
5820 \numberproblemsin{assignment}
5821 \renewcommand\prob@label[1]{\arabic{assignment}.#1}

```

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

```

5822 \keys_define:nn { hwexam / assignment } {
5823   id .str_set:N = \l__hwexam_assign_id_str,
5824   number .int_set:N = \l__hwexam_assign_number_int,
5825   title .tl_set:N = \l__hwexam_assign_title_tl,
5826   type .tl_set:N = \l__hwexam_assign_type_tl,
5827   given .tl_set:N = \l__hwexam_assign_given_tl,
5828   due .tl_set:N = \l__hwexam_assign_due_tl,
5829   loadmodules .code:n = {
5830     \bool_set_true:N \l__hwexam_assign_loadmodules_bool
5831   }
5832 }
5833 \cs_new_protected:Nn \__hwexam_assignment_args:n {
5834   \str_clear:N \l__hwexam_assign_id_str
5835   \int_set:Nn \l__hwexam_assign_number_int {-1}
5836   \tl_clear:N \l__hwexam_assign_title_tl
5837   \tl_clear:N \l__hwexam_assign_type_tl
5838   \tl_clear:N \l__hwexam_assign_given_tl
5839   \tl_clear:N \l__hwexam_assign_due_tl
5840   \bool_set_false:N \l__hwexam_assign_loadmodules_bool
5841   \keys_set:nn { hwexam / assignment }{ #1 }
5842 }

```

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.

```

5843 \newcommand\given@due[2]{
5844 \bool_lazy_all:nF {
5845 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
5846 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
5847 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
5848 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
5849 }{ #1 }
5850
5851 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
5852 \tl_if_empty:NF \l__hwexam_assign_given_tl {
5853 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
5854 }
5855 }{
5856 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
5857 }
5858
5859 \bool_lazy_or:nnF {
5860 \bool_lazy_and_p:nn {
5861 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
5862 }{
5863 \tl_if_empty_p:V \l__hwexam_assign_due_tl
5864 }
5865 }{
5866 \bool_lazy_and_p:nn {
5867 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
5868 }{
5869 \tl_if_empty_p:V \l__hwexam_assign_due_tl
5870 }
5871 }{ ,~ }
5872
5873 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
5874 \tl_if_empty:NF \l__hwexam_assign_due_tl {
5875 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
5876 }
5877 }{
5878 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
5879 }
5880
5881 \bool_lazy_all:nF {
5882 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
5883 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
5884 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
5885 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
5886 }{ #2 }
5887 }

```

`\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.

```

5888 \newcommand\assignment@title[3]{

```

```

5889 \tl_if_empty:NTF \l__hwexam_inclassassign_title_tl {
5890 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
5891 #1
5892 }{
5893 #2\l__hwexam_assign_title_tl#3
5894 }
5895 }{
5896 #2\l__hwexam_inclassassign_title_tl#3
5897 }
5898 }

```

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

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

```

5899 \newcommand\assignment@number{
5900 \int_compare:nNnTF \l__hwexam_inclassassign_number_int = {-1} {
5901 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
5902 \int_use:N \l__hwexam_assign_number_int
5903 }
5904 }{
5905 \int_use:N \l__hwexam_inclassassign_number_int
5906 }
5907 }

```

(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.

```

5908 \newenvironment{assignment}[1][]{
5909 \__hwexam_assignment_args:n { #1 }
5910 %\sref@target
5911 \let\__hwexamnum\l__hwexam_assign_number_int
5912 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
5913 \stepcounter{assignment}
5914 }{
5915 \setcounter{assignment}{\int_use:N\__hwexamnum}
5916 }
5917 \setcounter{problem}{0}
5918 \def\current@section@level{\document@hwexamtype}
5919 %\sref@label@id{\document@hwexamtype \thesection}
5920 \begin{@assignment}
5921 }{
5922 \end{@assignment}
5923 }

```

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

```

5924 \def\__hwexasstitle{
5925 \protect\document@hwexamtype~\arabic{assignment}
5926 \assignment@title{}\;{} \; -- \given@due{}{}
5927 }

```



```

5928 \ifmultiple
5929 \newenvironment{@assignment}{
5930 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
5931 \begin{omgroup}[loadmodules]{\__hwexasstitle}
5932 }{
5933 \begin{omgroup}{\__hwexasstitle}
5934 }
5935 }{
5936 \end{omgroup}
5937 }

```

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

```

5938 \else
5939 \newenvironment{@assignment}{
5940 \begin{center}\bf
5941 \Large\@title\strut\
5942 \document@hwexamtype~\arabic{assignment}\assignment@title{\;}{:\;}{\}
5943 \large\given@due{--\;}{\;}{--}
5944 \end{center}
5945 }{}
5946 \fi% multiple

```

## 42.3 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.

```

5947 \keys_define:nn { hwexam / inclassignment } {
5948 %id .str_set:N = \l__hwexam_assign_id_str,
5949 number .int_set:N = \l__hwexam_inclassign_number_int,
5950 title .tl_set:N = \l__hwexam_inclassign_title_tl,
5951 type .tl_set:N = \l__hwexam_inclassign_type_tl,
5952 given .tl_set:N = \l__hwexam_inclassign_given_tl,
5953 due .tl_set:N = \l__hwexam_inclassign_due_tl,
5954 mhrepos .str_set:N = \l__hwexam_inclassign_mhrepos_str
5955 }
5956 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
5957 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
5958 \tl_clear:N \l__hwexam_inclassign_title_tl
5959 \tl_clear:N \l__hwexam_inclassign_type_tl
5960 \tl_clear:N \l__hwexam_inclassign_given_tl
5961 \tl_clear:N \l__hwexam_inclassign_due_tl
5962 \str_clear:N \l__hwexam_inclassign_mhrepos_str
5963 \keys_set:nn { hwexam / inclassignment }{ #1 }
5964 }
5965 \__hwexam_inclassignment_args:n {}
5966
5967 \newcommand\inputassignment[2][ ]{
5968 \__hwexam_inclassignment_args:n { #1 }
5969 \str_if_empty:NTF \l__hwexam_inclassign_mhrepos_str {
5970 \input{#2}
5971 }{
5972 \stex_in_repository:nn{\l__hwexam_inclassign_mhrepos_str}{

```

```

5973 \input{\mhp\path{\l__hwexam_inclasssign_mhrepos_str}{#2}}
5974 }
5975 }
5976 \__hwexam_inclasssign_args:n {}
5977 }
5978 \newcommand\includeassignment[2][ ]{
5979 \newpage
5980 \inputassignment[#1]{#2}
5981 }

```

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

## 42.4 Typesetting Exams

\quizheading

```

5982 \ExplSyntaxOff
5983 \newcommand\quizheading[1]{%
5984 \def\@tas{#1}%
5985 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
5986 \ifx\@tas\empty\else%
5987 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
5988 \fi%
5989 }
5990 \ExplSyntaxOn

```

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

\testheading

```

5991 \keys_define:nn { hwexam / testheading } {
5992 min .tl_set:N = \l__hwexam_testheading_min_tl,
5993 duration .tl_set:N = \l__hwexam_testheading_duration_tl,
5994 reqpts .tl_set:N = \l__hwexam_testheading_reqpts_tl
5995 }
5996 \cs_new_protected:Nn \__hwexam_testheading_args:n {
5997 \tl_clear:N \l__hwexam_testheading_min_tl
5998 \tl_clear:N \l__hwexam_testheading_duration_tl
5999 \tl_clear:N \l__hwexam_testheading_reqpts_tl
6000 \keys_set:nn { hwexam / testheading }{ #1 }
6001 }
6002 \newenvironment{testheading}[1][ ]{
6003 \__hwexam_testheading_args:n{ #1 }
6004 \noindent\large{Name:~\hfill
6005 Matriculation Number:\hspace*{2cm}\strut}\[1ex]
6006 \begin{center}
6007 \Large\textbf{\@title}\[1ex]
6008 \large\@date\[3ex]
6009 \end{center}
6010 \textbf{You~have~
6011 \tl_if_empty:NTF \l__hwexam_testheading_duration_tl {
6012 {\l__hwexam_testheading_min_tl}~minutes
6013 }{
6014 {\l__hwexam_testheading_duration_tl}
6015 }~

```

```

6016 (sharp)~for~the~test
6017 };\
6018 Write~the~solutions~to~the~sheet.
6019 \par\noindent
6020 \newcount\check@time\check@time=\l__hwexam_testheading_min_tl
6021 \advance\check@time by -\theassignment@totalmin
6022 The~estimated~time~for~solving~this~exam~is~
6023 {\theassignment@totalmin}-minutes,~
6024 leaving~you~{\the\check@time}-minutes~for~revising~
6025 your~exam.
6026
6027 \par\noindent
6028 \newcount\bonus@pts\bonus@pts=\theassignment@totalpts
6029 \advance\bonus@pts by -\l__hwexam_testheading_reqpts_tl
6030 You~can~reach~{\theassignment@totalpts}-points~if~you~
6031 solve~all~problems.~You~will~only~need~
6032 {\l__hwexam_testheading_reqpts_tl}-points~for~a~perfect~score,~
6033 i.e.\ {\the\bonus@pts}-points~are~bonus~points.
6034 \vfill
6035 \begin{center}
6036 {
6037 \Large\em You~have~ample~time,~so~take~it~slow~
6038 and~avoid~rushing~to~mistakes!\}[2ex]
6039 Different~problems~test~different~skills~and~
6040 knowledge,~so~do~not~get~stuck~on~one~problem.
6041 }
6042 \vfill\par\resizebox{\textwidth}{!}{\correction@table}\}[3ex]
6043 \end{center}
6044 }{
6045 \newpage
6046 }

```

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

\testspace

```

6047 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

6048 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

6049 \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.

```

6050 <@=problems>
6051 \renewcommand\@problem[3]{
6052 \stepcounter{assignment@probs}
6053 \def\__problemspts{#2}

```

```

6054 \ifx\__problemspts\@empty\else
6055 \addtocounter{assignment@totalpts}{#2}
6056 \fi
6057 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
6058 \xdef\correction@probs{\correction@probs & #1}%
6059 \xdef\correction@pts{\correction@pts & #2}
6060 \xdef\correction@reached{\correction@reached & }
6061 }
6062 \@@=hwexam>

```

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

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

```

6063 \newcounter{assignment@probs}
6064 \newcounter{assignment@totalpts}
6065 \newcounter{assignment@totalmin}
6066 \def\correction@probs{\correction@probs@kw}%
6067 \def\correction@pts{\correction@pts@kw}%
6068 \def\correction@reached{\correction@reached@kw}%
6069 \def\after@correction@table{}%
6070 \stepcounter{assignment@probs}
6071 \newcommand\correction@table{
6072 \resizebox{\textwidth}{!}{%
6073 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
6074 &\multicolumn{\theassignment@probs}{c|}|%|
6075 {\footnotesize\correction@forgrading@kw} &\\ \hline
6076 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
6077 \correction@pts & \theassignment@totalpts & \\ \hline
6078 \correction@reached & & \[.7cm]\hline
6079 \end{tabular}}
6080 \ifx\after@correction@table\@empty\else\strut\par\noindent\after@correction@table\fi}
6081 \end{package}

```

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

## 42.5 Leftovers

at some point, we may want to reactivate the logos font, then we use

here we define the logos that characterize the assignment

```

\font\bierrfont=../assignments/bierglas
\font\denkerfont=../assignments/denker
\font\uhrfont=../assignments/uhr
\font\warnschildfont=../assignments/achtung

```

```

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

```