

# 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 . . . . .	121
<b>31</b>	<b>STeX-Terms Implementation</b>	<b>130</b>
31.1	Symbol Invocations . . . . .	130
31.2	Terms . . . . .	133
31.3	Notation Components . . . . .	139
<b>32</b>	<b>STeX-Structural Features Implementation</b>	<b>142</b>
32.1	Imports with modification . . . . .	142
32.2	The feature environment . . . . .	148
32.3	Features . . . . .	150
<b>33</b>	<b>STeX-Statements Implementation</b>	<b>156</b>
33.1	Definitions . . . . .	156
33.2	Assertions . . . . .	159
33.3	Examples . . . . .	161
33.4	Logical Paragraphs . . . . .	163

<b>34 The Implementation</b>	<b>166</b>
34.1 Package Options . . . . .	166
34.2 Proofs . . . . .	166
34.3 Justifications . . . . .	172
<b>35 <math>\text{\LaTeX}</math>-Others Implementation</b>	<b>174</b>
<b>36 <math>\text{\LaTeX}</math>-Metatheory Implementation</b>	<b>175</b>
<b>37 Tikzinput Implementation</b>	<b>178</b>
<b>38 document-structure.sty Implementation</b>	<b>180</b>
38.1 The OMDoc Class . . . . .	180
38.2 Class Options . . . . .	180
38.3 Beefing up the <code>document</code> environment . . . . .	181
38.4 Implementation: OMDoc Package . . . . .	181
38.5 Package Options . . . . .	181
38.6 Document Structure . . . . .	183
38.7 Front and Backmatter . . . . .	186
38.8 Global Variables . . . . .	188
<b>39 MiKoSlides – Implementation</b>	<b>189</b>
39.1 Class and Package Options . . . . .	189
39.2 Notes and Slides . . . . .	191
39.3 Header and Footer Lines . . . . .	195
39.4 Frame Images . . . . .	196
39.5 Colors and Highlighting . . . . .	197
39.6 Sectioning . . . . .	198
39.7 Excursions . . . . .	200
<b>40 The Implementation</b>	<b>202</b>
40.1 Package Options . . . . .	202
40.2 Problems and Solutions . . . . .	203
40.3 Multiple Choice Blocks . . . . .	208
40.4 Including Problems . . . . .	209
40.5 Reporting Metadata . . . . .	210
<b>41 Implementation: The hwexam Class</b>	<b>212</b>
41.1 Class Options . . . . .	212
<b>42 Implementation: The hwexam Package</b>	<b>214</b>
42.1 Package Options . . . . .	214
42.2 Assignments . . . . .	215
42.3 Including Assignments . . . . .	218
42.4 Typesetting Exams . . . . .	219
42.5 Leftovers . . . . .	221

**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,
- RuSTeX 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 $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$ Document

Having set everything up, we can write a first  $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$  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  $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$  *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}`),  $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$  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}`.

$\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$  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/smglom/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}
\syndef{universe}{\comp{\mathcal U}}
\syndef[ args=2,op=\circ ]{operation}{\#1 \comp\circ \#2}
\end{module}
\begin{module}{monoid}
\importmodule{magma}
\syndef{unit}{\comp e}
\end{module}
\begin{module}{group}
\importmodule{monoid}
\syndef[ 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{clonemodule}{group}{addition}
\renamedecl[name=universe]{universe}{runiverse}
\renamedecl[name=plus]{operation}{rplus}
\renamedecl[name=zero]{unit}{rzero}
\renamedecl[name=uminus]{inverse}{rminus}
\end{clonemodule}
\notation[plus,op=+,prec=60]{rplus}{#1 \comp+ #2}
\notation[zero]{rzero}{\comp0}
\notation[uminus,op=-]{rminus}{\comp- #1}
\begin{clonemodule}{monoid}{multiplication}
\assign{universe}{runiverse}
\renamedecl[name=times]{operation}{rtimes}
\renamedecl[name=one]{unit}{rone}
\end{clonemodule}
\notation[cdot,op=\cdot,prec=50]{rtimes}{#1 \comp\cdot #2}
\notation[one]{rone}{\compl}

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

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

TODO: explain donotclone

## 5.2 Primitive Symbols (The $\text{\TeX}$ 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` Both print this  $\text{\TeX}$  logo.  
`\stex`

---

### 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 3

```
\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 4

```
\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 5

```
 $\mult*{a}[\comp{\ast}]{b}$  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 6

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

Multiplying again by  $b$  yields...

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

#### Example 7

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

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

<sup>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 8

```
\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 9

```
\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 10

```
\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 11

```
\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 12

```
\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 {&lt;property&gt;} {&lt;resource&gt;} {&lt;content&gt;}</code>
<code>\stex_annotate_invisible:nnn</code>	
<code>\stex_annotate_invisible:n</code>	

---

Annotates the HTML generated by `<content>` with

`property="stex:<property>", resource="<resource>"`.

`\stex_annotate_invisible:n` adds the attributes

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

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

<code>stex_annotate_env</code>	<code>\begin{stex_annotate_env}{&lt;property&gt;}{&lt;resource&gt;}</code> <code>&lt;content&gt;</code> <code>\end{stex_annotate_env}</code> behaves like <code>\stex_annotate:nnn {&lt;property&gt;} {&lt;resource&gt;} {&lt;content&gt;}</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&lt;cs&gt;{&lt;environments&gt;}</code>
<code>\stex_reactivate_macro:N</code>	

---

Makes the macro `<cs>` throw an error, indicating that it is only allowed in the context of `<environments>`.

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

---

<code>\MSC</code>	<code>\MSC{&lt;msc&gt;}</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`.

---

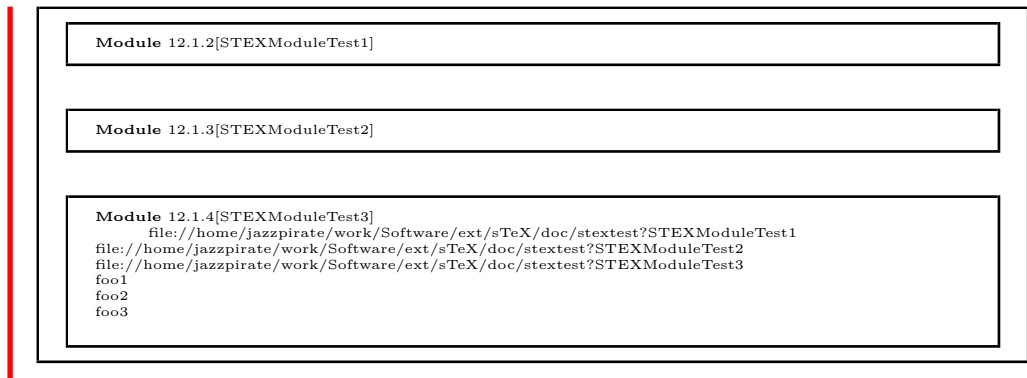
`\stex_invoke_module:n`

---

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.  $\text{\TeX}$  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`,  
`file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest2`  
All symbols: `http://mathhub.info/sTeX?Metatheory?isa`, `http://mathhub.info/sTeX?Metatheory?bind`, `http://mathhub.info/sTeX?Metatheory?collect`,  
`http://mathhub.info/sTeX?Metatheory?fromto`, `http://mathhub.info/sTeX?Metatheory?apply`, `http://mathhub.info/sTeX?Metatheory?collect`,  
`http://mathhub.info/sTeX?Metatheory?seqtype`, `http://mathhub.info/sTeX?Metatheory?sequence-index`, `http://mathhub.info/sTeX?Metatheory?sequence-index`,  
`http://mathhub.info/sTeX?Metatheory?aseqfromto`, `http://mathhub.info/sTeX?Metatheory?aseqfromto`, `http://mathhub.info/sTeX?Metatheory?aseqfromto`,  
`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

---

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

---

Declares a new symbol with semantic macro `\macroname`. Optional arguments are:

- **name**: An (OMDOC) name. By default equal to `⟨macroname⟩`.
- **type**: An (ideally semantic) term. Not used by 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. `\symdecl[args=ii]{plus}` allows for `\plus{2}{2}`.
  - a an *associative* argument; i.e. a sequence of arbitrarily many arguments provided as a comma-separated list, e.g. `\symdecl[args=a]{plus}` allows for `\plus{2,2,2}`.
  - b a *variable* argument. Is treated by 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. `\symdecl[args=bi]{forall}` allows for `\forall{x\in\Nat}{x\geq0}`.

---

---

`\stex_symdecl_do:n`

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

Ultimately stores the symbol  $\langle URI \rangle$  in the property list `\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

# TeX-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 its sister package `statements`.

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

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

Example 1: A very explicit proof, marked up semantically

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` `\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{\@I\angle}}#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

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

`\STRsemantics` 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`{ $\langle vname \rangle$ }{ $\langle text \rangle$ } to set the global variable  $\langle vname \rangle$  to  $\langle text \rangle$  and `\useSGvar`{ $\langle vname \rangle$ } to reference it.

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

<sup>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> |

EdN:11



<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  $\TeX$  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  $\TeX$ 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.
```

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

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

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

### 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>X<sub>M</sub>L, 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>X<sub>M</sub>L-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:nxx{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:n`

```
281 <@@=stex_aftergroup>  
282 \cs_new_protected:Nn \stex_do_aftergroup:n {  
283   \tl_gset:Nn \l__stex_aftergroup_tl { #1 }  
284   \aftergroup\__stex_aftergroup_do:  
285 }  
286 \cs_new_protected:Nn \__stex_aftergroup_do: {  
287   \l__stex_aftergroup_tl  
288 }
```

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

```
289 </package>
```

## Chapter 26

# STEX -MathHub Implementation

```
290 <*package>
291
292 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
293
294 <@@=stex_path>
295
296 Warnings and error messages
297 \msg_new:nnn{stex}{error/norepository}{
298   No~archive~#1~found~in~#2
299 }
300 \msg_new:nnn{stex}{error/notinarchive}{
301   Not~currently~in~an~archive,~but~\detokenize{#1}~
302   needs~one!
303 }
304 \msg_new:nnn{stex}{error/nofile}{
305   \detokenize{#1}~could~not~find~file~#2
306 }
```

### 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
305 \cs_new_protected:Nn \stex_path_from_string:Nn {
306   \str_set:Nx \l_tmpa_str { #2 }
307   \str_if_empty:NTF \l_tmpa_str {
308     \seq_clear:N #1
309   }{
310     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
311     \sys_if_platform_windows:T{
312       \seq_clear:N \l_tmpa_tl
313       \seq_map_inline:Nn #1 {
314         \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
315         \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
316       }
317     }
318   }
```

```

316     }
317     \seq_set_eq:NN #1 \l_tmpa_tl
318   }
319   \stex_path_canonicalize:N #1
320 }
321 }
322 \cs_generate_variant:Nn \stex_path_from_string:Nn
323 { 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
324 \cs_new_protected:Nn \stex_path_to_string:NN {
325   \exp_args:NNe \str_set:Nn #2 { \seq_use:Nn #1 / }
326 }
327
328 \cs_new:Nn \stex_path_to_string:N {
329   \seq_use:Nn #1 /
330 }

```

(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
331 \str_const:Nn \c__stex_path_dot_str {.}
332 \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.

```

333 \cs_new_protected:Nn \stex_path_canonicalize:N {
334   \seq_if_empty:NF #1 {
335     \seq_clear:N \l_tmpa_seq
336     \seq_get_left:NN #1 \l_tmpa_tl
337     \str_if_empty:NT \l_tmpa_tl {
338       \seq_put_right:Nn \l_tmpa_seq {}
339     }
340     \seq_map_inline:Nn #1 {
341       \str_set:Nn \l_tmpa_tl { ##1 }
342       \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_dot_str {} {
343         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
344           \seq_if_empty:NNTF \l_tmpa_seq {
345             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
346               \c__stex_path_up_str
347             }
348           }{
349             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
350             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
351               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
352                 \c__stex_path_up_str
353               }
354             }{
355               \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
356             }

```

```

357     }
358   }{
359     \str_if_empty:NF \l_tmpa_tl {
360       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
361     }
362   }
363 }
364 }
365 \seq_gset_eq:NN #1 \l_tmpa_seq
366 }
367 }

```

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

`\stex_path_if_absolute_p:N`  
`\stex_path_if_absolute:N $\underline{TF}$`

```

368 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
369   \seq_if_empty:NTF #1 {
370     \prg_return_false:
371   }{
372     \seq_get_left:NN #1 \l_tmpa_tl
373     \str_if_empty:NTF \l_tmpa_tl {
374       \prg_return_true:
375     }{
376       \prg_return_false:
377     }
378   }
379 }

```

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

## 26.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

380 \str_new:N\l_stex_kpsewhich_return_str
381 \cs_new_protected:Nn \stex_kpsewhich:n {
382   \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
383   \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
384   \tl_trim_spaces:N \l_stex_kpsewhich_return_str
385 }

```

(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`

```

386 \sys_if_platform_windows:TF{
387   \stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
388 }{
389   \stex_kpsewhich:n{-var-value~PWD}
390 }
391
392 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
393 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
394 \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

395 `<@@=stex_files>`

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

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

`\g__stex_files_stack` keeps track of file changes

396 `\seq_gclear_new:N\g__stex_files_stack`

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`

`\c_stex_mainfile_str`

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

398 `\stex_path_from_string:Nn \c_stex_mainfile_seq`

399 `\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`.

```
400 \seq_gclear_new:N\g_stex_currentfile_seq
401 \AddToHook{file/before}{
402   \stex_path_from_string:Nn\g_stex_currentfile_seq{\CurrentFilePath}
403   \stex_path_if_absolute:NTF\g_stex_currentfile_seq{
404     \exp_args:NNe\seq_put_right:Nn\g_stex_currentfile_seq{\CurrentFile}
405   }{
406     \stex_path_from_string:Nn\g_stex_currentfile_seq{
407       \c_stex_pwd_str/\CurrentFilePath/\CurrentFile
408     }
409   }
410   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
411   \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
412 }
413 \AddToHook{file/after}{
414   \seq_if_empty:NF\g__stex_files_stack{
415     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
416   }
417   \seq_if_empty:NTF\g__stex_files_stack{
418     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
419   }{
420     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
421     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
422   }
423 }
```

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



## 26.4 MathHub Repositories

```

424 <@@=stex_mathhub>

\mathhub
\c_stex_mathhub_seq
\c_stex_mathhub_str
425 \str_if_empty:NTF\mathhub{
426   \stex_kpsewhich:n{-var-value~MATHHUB}
427   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
428
429   \str_if_empty:NTF\c_stex_mathhub_str{
430     \msg_warning:nn{stex}{warning/nomathhub}
431   }{
432     \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
433     \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
434   }
435 }{
436   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
437   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
438     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
439       \c_stex_pwd_str/\mathhub
440     }
441   }
442   \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
443   \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
444 }

```

(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
445 \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
446   \str_set:Nx \l_tmpa_str { #1 }
447   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
448     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
449     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
450     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
451     \__stex_mathhub_find_manifest:N \l_tmpa_seq
452     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
453       \msg_error:nnxx{stex}{error/norepository}{#1}{
454         \stex_path_to_string:N \c_stex_mathhub_str
455       }
456     } {
457       \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
458     }
459   }
460 }

```

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

```

\l__stex_mathhub_manifest_file_seq
461 \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`:

```

462 \cs_new_protected:Nn \_stex_mathhub_find_manifest:N {
463   \seq_set_eq:NN \l_tmpa_seq #1
464   \bool_set_true:N \l_tmpa_bool
465   \bool_while_do:Nn \l_tmpa_bool {
466     \seq_if_empty:NTF \l_tmpa_seq {
467       \bool_set_false:N \l_tmpa_bool
468     }{
469       \file_if_exist:nTF{
470         \stex_path_to_string:N \l_tmpa_seq/MANIFEST.MF
471       }{
472         \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
473         \bool_set_false:N \l_tmpa_bool
474       }{
475         \file_if_exist:nTF{
476           \stex_path_to_string:N \l_tmpa_seq/META-INF/MANIFEST.MF
477         }{
478           \seq_put_right:Nn \l_tmpa_seq{META-INF}
479           \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
480           \bool_set_false:N \l_tmpa_bool
481         }{
482           \file_if_exist:nTF{
483             \stex_path_to_string:N \l_tmpa_seq/meta-inf/MANIFEST.MF
484           }{
485             \seq_put_right:Nn \l_tmpa_seq{meta-inf}
486             \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
487             \bool_set_false:N \l_tmpa_bool
488           }{
489             \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
490           }
491         }
492       }
493     }
494   }
495   \seq_set_eq:NN \l__stex_mathhub_manifest_file_seq \l_tmpa_seq
496 }

```

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

`\c_stex_mathhub_manifest_ior` File variable used for MANIFEST-files

```

497 \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:

```

498 \cs_new_protected:Nn \_stex_mathhub_parse_manifest:n {
499   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
500   \ior_open:Nn \c_stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
501   \ior_map_inline:Nn \c_stex_mathhub_manifest_ior {
502     \str_set:Nn \l_tmpa_str {##1}
503     \exp_args:NNoo \seq_set_split:Nnn
504       \l_tmpb_seq \c_colon_str \l_tmpa_str
505     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {

```

```

506 \exp_args:NNe \str_set:Nn \l_tmpb_tl {
507 \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
508 }
509 \exp_args:No \str_case:nnTF \l_tmpa_tl {
510 {id} {
511 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
512 { id } \l_tmpb_tl
513 }
514 {narration-base} {
515 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
516 { narr } \l_tmpb_tl
517 }
518 {url-base} {
519 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
520 { docurl } \l_tmpb_tl
521 }
522 {source-base} {
523 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
524 { ns } \l_tmpb_tl
525 }
526 {ns} {
527 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
528 { ns } \l_tmpb_tl
529 }
530 {dependencies} {
531 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
532 { deps } \l_tmpb_tl
533 }
534 }{}{}
535 }{}
536 }
537 \ior_close:N \c__stex_mathhub_manifest_ior
538 }

```

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

`\stex_set_current_repository:n`

```

539 \cs_new_protected:Nn \stex_set_current_repository:n {
540 \stex_require_repository:n { #1 }
541 \prop_set_eq:Nc \l_stex_current_repository_prop {
542 c_stex_mathhub_#1_manifest_prop
543 }
544 }

```

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

`\stex_require_repository:n`

```

545 \cs_new_protected:Nn \stex_require_repository:n {
546 \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
547 \stex_debug:nn{mathhub}{Opening~archive:~#1}
548 \__stex_mathhub_do_manifest:n { #1 }
549 \exp_args:Nx \stex_add_to_sms:n {
550 \prop_const_from_keyval:cn { c_stex_mathhub_#1_manifest_prop } {
551 id = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { id } ,
552 ns = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { ns } ,

```

```

553     narr = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { narr } ,
554     deps = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { deps }
555   }
556 }
557 }
558 }

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

559 %\prop_new:N \l_stex_current_repository_prop
560
561 \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
562 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
563   \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
564 } {
565   \__stex_mathhub_parse_manifest:n { main }
566   \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
567   \l_tmpa_str
568   \prop_set_eq:cN { c_stex_mathhub_ \l_tmpa_str _manifest_prop }
569   \c_stex_mathhub_main_manifest_prop
570   \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
571   \stex_debug:nn{mathhub}{Current~repository:~
572   \prop_item:Nn \l_stex_current_repository_prop {id}
573 }
574 }

```

(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.

```

575 \cs_new_protected:Nn \stex_in_repository:nn {
576   \str_set:Nx \l_tmpa_str { #1 }
577   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
578   \str_if_empty:NTF \l_tmpa_str {
579     \prop_if_exist:NTF \l_stex_current_repository_prop {
580       \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
581       \exp_args:Ne \l_tmpa_cs{
582         \prop_item:Nn \l_stex_current_repository_prop { id }
583       }
584     }{
585       \l_tmpa_cs{}
586     }
587   }{
588     \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}
589     \stex_require_repository:n \l_tmpa_str
590     \str_set:Nx \l_tmpa_str { #1 }
591     \exp_args:Nne \use:nn {
592       \stex_set_current_repository:n \l_tmpa_str
593       \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
594     }{
595       \stex_debug:nn{mathhub}{switching~back~to:~
596       \prop_if_exist:NTF \l_stex_current_repository_prop {
597         \prop_item:Nn \l_stex_current_repository_prop { id }::~

```

```

598         \meaning\l_stex_current_repository_prop
599     }{
600         no~repository
601     }
602 }
603 \prop_if_exist:NTF \l_stex_current_repository_prop {
604     \stex_set_current_repository:n {
605         \prop_item:Nn \l_stex_current_repository_prop { id }
606     }
607 }{
608     \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
609 }
610 }
611 }
612 }

```

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

```

\inputref
\stex_inputref:nn
\mhinput\stex_mhinput:nn
613 \newif \ifinputref \inputreffalse
614
615 \cs_new_protected:Nn \stex_mhinput:nn {
616     \stex_in_repository:nn {#1} {
617         \ifinputref
618             \input{ \c_stex_mathhub_str / ##1 / source / #2 }
619         \else
620             \inputreftrue
621             \input{ \c_stex_mathhub_str / ##1 / source / #2 }
622             \inputreffalse
623         \fi
624     }
625 }
626 \NewDocumentCommand \mhinput { 0{} m}{
627     \stex_mhinput:nn{ #1 }{ #2 }
628 }
629
630 \cs_new_protected:Nn \stex_inputref:nn {
631     \stex_in_repository:nn {#1} {
632         \bool_lazy_any:nTF {
633             {\rustex_if_p:} {\latexml_if_p:}
634         } {
635             \str_clear:N \l_tmpa_str
636             \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
637                 \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
638             }
639             \stex_annotate_invisible:nnn{inputref}{
640                 \l_tmpa_str / #2
641             }{}
642         }{
643             \begingroup
644             \inputreftrue
645             \input{ \c_stex_mathhub_str / ##1 / source / #2 }
646             \endgroup
647         }

```

```

648 }
649 }
650
651 \NewDocumentCommand \inputref { 0{} m}{
652   \stex_inputref:nn{ #1 }{ #2 }
653 }
654
655 \cs_new_protected:Nn \stex_mhbibresource:nn {
656   \stex_in_repository:nn {#1} {
657     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
658   }
659 }
660 \newcommand\addmhbibresource[2][]{
661   \stex_mhbibresource:nn{ #1 }{ #2 }
662 }

```

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

#### `\mhpath`

```

663 \def \mhpath #1 #2 {
664   \exp_args:Ne \str_if_eq:nnTF{#1}{-}{
665     \c_stex_mathhub_str /
666     \prop_item:Nn \l_stex_current_repository_prop { id }
667     / source / #2
668   }{
669     \c_stex_mathhub_str / #1 / source / #2
670   }
671 }

```

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

#### `\libinput`

```

672 \cs_new_protected:Npn \libinput #1 {
673   \prop_if_exist:NF \l_stex_current_repository_prop {
674     \msg_error:nnn{stex}{error/notinarchive}\libinput
675   }
676   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
677     \msg_error:nnn{stex}{error/notinarchive}\libinput
678   }
679   \bool_set_false:N \l_tmpa_bool
680   \tl_clear:N \l_tmpa_tl
681   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
682   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
683   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str
684   \seq_pop_left:NNT \l_tmpb_seq \l_tmpb_str {
685     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
686     \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
687       / meta-inf / lib / #1.tex}{
688       \bool_set_true:N \l_tmpa_bool
689       \tl_put_right:Nx \l_tmpa_tl {
690         \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
691           / meta-inf / lib / #1.tex}
692       }
693     }{}

```

```

694 }
695 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
696 / \l_tmpa_str / lib / #1.tex
697 }{
698   \bool_set_true:N \l_tmpa_bool
699   \tl_put_right:Nx \l_tmpa_tl {
700     \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
701       / \l_tmpa_str / lib / #1.tex}
702   }
703 }{}
704 \bool_if:NF \l_tmpa_bool {
705   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
706 }
707 \l_tmpa_tl
708 }

```

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

```

709 </package>

```

## Chapter 27

# STEX -References Implementation

```
710 <*package>
711
712 %%%%%%%%%% references.dtx %%%%%%%%%%
713
714 %\RequirePackage{hyperref}
715 %\RequirePackage{cleveref}
716 <@@=stex_refs>
717
718 Warnings and error messages
719
720 \iow_new:N \c__stex_refs_refs_iow
721 \AddToHook{begindocument}{
722   \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
723 }
724 \AddToHook{enddocument}{
725   \iow_close:N \c__stex_refs_refs_iow
726 }
727
728 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
729
730 \NewDocumentCommand \STEXreftitle { m } {
731   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
732 }
733 }
```

### 27.1 Document URIs and URLs

```
731 \seq_new:N \g__stex_refs_all_refs_seq
732
733 \str_new:N \l_stex_current_docns_str
734
735 \cs_new_protected:Nn \stex_get_document_uri: {
736   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
737   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
738   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
739   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
740 }
```



```

740 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
741
742 \str_clear:N \l_tmpa_str
743 \prop_if_exist:NT \l_stex_current_repository_prop {
744   \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
745     \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
746   }
747 }
748
749 \str_if_empty:NTF \l_tmpa_str {
750   \str_set:Nx \l_stex_current_docns_str {
751     file:/\stex_path_to_string:N \l_tmpa_seq
752   }
753 }{
754   \bool_set_true:N \l_tmpa_bool
755   \bool_while_do:Nn \l_tmpa_bool {
756     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
757     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
758       {source} { \bool_set_false:N \l_tmpa_bool }
759     }{}{
760       \seq_if_empty:NT \l_tmpa_seq {
761         \bool_set_false:N \l_tmpa_bool
762       }
763     }
764   }
765
766   \seq_if_empty:NTF \l_tmpa_seq {
767     \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
768   }{
769     \str_set:Nx \l_stex_current_docns_str {
770       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
771     }
772   }
773 }
774 }
775
776 \str_new:N \l_stex_current_docurl_str
777 \cs_new_protected:Nn \stex_get_document_url: {
778   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
779   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
780   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
781   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
782   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
783
784   \str_clear:N \l_tmpa_str
785   \prop_if_exist:NT \l_stex_current_repository_prop {
786     \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
787       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
788         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
789       }
790     }
791   }
792
793   \str_if_empty:NTF \l_tmpa_str {
794     \str_set:Nx \l_stex_current_docurl_str {

```

```

794     file:/\stex_path_to_string:N \l_tmpa_seq
795   }
796 }{
797   \bool_set_true:N \l_tmpa_bool
798   \bool_while_do:Nn \l_tmpa_bool {
799     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
800     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
801       {source} { \bool_set_false:N \l_tmpa_bool }
802     }{}{
803       \seq_if_empty:NT \l_tmpa_seq {
804         \bool_set_false:N \l_tmpa_bool
805       }
806     }
807   }
808
809   \seq_if_empty:NTF \l_tmpa_seq {
810     \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
811   }{
812     \str_set:Nx \l_stex_current_docurl_str {
813       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
814     }
815   }
816 }
817 }

```

## 27.2 Setting Reference Targets

```

818 \str_const:Nn \c__stex_refs_url_str{URL}
819 \str_const:Nn \c__stex_refs_ref_str{REF}
820 % @currentlabel -> number
821 % @currentlabelname -> title
822 % @currentHref -> name.number <- id of some kind
823 % \theH# -> \arabic{section}
824 % \the# -> number
825 % \hyper@makecurrent{#}
826 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
827   \stex_get_document_uri:
828   \str_set:Nx \l_tmpa_str { #1 }
829   \str_if_empty:NT \l_tmpa_str {
830     \int_zero:N \l_tmpa_int
831     \bool_set_true:N \l_tmpa_bool
832     \bool_while_do:Nn \l_tmpa_bool {
833       \cs_if_exist:cTF {
834         sref_\l_stex_current_docns_str?? REF_\int_use:N \l_tmpa_int _type
835       }{
836         \int_incr:N \l_tmpa_int
837       }{
838         \str_set:Nx \l_tmpa_str { REF_\int_use:N \l_tmpa_int }
839         \bool_set_false:N \l_tmpa_bool
840       }
841     }
842   }
843   \str_set:Nx \l_tmpa_str {
844     \l_stex_current_docns_str??\l_tmpa_str

```

```

845 }
846 \seq_gput_right:No \g__stex_refs_all_refs_seq \l_tmpa_str
847 \stex_if_smsmode:TF {
848   \stex_get_document_url:
849   \str_gset_eq:cN {sref_url_\l_tmpa_str _str}\l_stex_current_docurl_str
850   \str_gset_eq:cN {sref_\l_tmpa_str _type}\c__stex_refs_url_str
851 }{
852   \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter{\@currentlabel\iffalse}{
853   \exp_args:Nx\label{sref_\l_tmpa_str}
854
855   \exp_args:NNNx\immediate\write\@auxout{\stexauxadddocref{\l_tmpa_str}}
856   \str_gset:cx {sref_\l_tmpa_str _type}\c__stex_refs_ref_str
857 }
858 }
859 \cs_new_protected:Npn \stexauxadddocref #1 {
860   \str_set:Nx \l_tmpa_str {#1}
861   \str_gset_eq:cN{sref_\l_tmpa_str _type}\c__stex_refs_ref_str
862   \seq_gput_right:Nx \g__stex_refs_all_refs_seq {\l_tmpa_str}
863 }
864 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
865   \str_gset_eq:cN {sref_sym_#1_uri} \l_stex_current_docns_str
866 }

```

## 27.3 Using References

```

867 \str_new:N \l__stex_refs_indocument_str
868 \keys_define:nn { stex / sref } {
869   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
870   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
871   pre           .tl_set:N = \l__stex_refs_pre_tl ,
872   post          .tl_set:N = \l__stex_refs_post_tl ,
873   %indoc        .str_set_x:N = \l__stex_refs_repo_str ,
874 }
875
876 \bool_new:N \c__stex_refs_hyperref_bool
877 \bool_set_false:N \c__stex_refs_hyperref_bool
878 \AddToHook{begindocument}{
879   \@ifpackageloaded{hyperref}{
880     \bool_set_true:N \c__stex_refs_hyperref_bool
881   }{}
882 }
883
884
885 \cs_new_protected:Nn \__stex_refs_args:n {
886   \tl_clear:N \l__stex_refs_linktext_tl
887   \tl_clear:N \l__stex_refs_fallback_tl
888   \tl_clear:N \l__stex_refs_pre_tl
889   \tl_clear:N \l__stex_refs_post_tl
890   \str_clear:N \l__stex_refs_repo_str
891   \keys_set:nn { stex / sref } { #1 }
892 }
893
894 \NewDocumentCommand \sref { 0{} m }{
895   \__stex_refs_args:n { #1 }

```

```

896 \str_if_empty:NTF \l__stex_refs_indocument_str {
897   \str_set:Nn \l_tmpa_str { #2 }
898   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
899   \tl_set:Nn \l_tmpa_tl {
900     \l__stex_refs_fallback_tl
901   }
902   \seq_map_inline:Nn \g__stex_refs_all_refs_seq {
903     \str_set:Nn \l_tmpb_str { ##1 }
904     \str_if_eq:eeT { \l_tmpa_str } {
905       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int }{-1 }
906     } {
907       \seq_map_break:n {
908         \tl_set:Nn \l_tmpa_tl {
909           % doc uri in \l_tmpb_str
910           \str_set:Nx \l_tmpa_str {\use:c{sref_\l_tmpb_str _type}}
911           \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
912             % reference
913             \cs_if_exist:cTF{autoref}{
914               \l__stex_refs_pre_tl\autoref{sref_\l_tmpb_str}\l__stex_refs_post_tl
915             }{
916               \l__stex_refs_pre_tl\ref{sref_\l_tmpb_str}\l__stex_refs_post_tl
917             }
918           }{
919             % URL
920             \if_bool:N \c__stex_refs_hyperref_bool {
921               \exp_args:Nx \href{\use:c{sref_url_\l_tmpb_str _str}}{\l__stex_refs_fallback
922             }{
923               \l__stex_refs_fallback_tl
924             }
925           }
926         }
927       }
928     }
929   }
930   \l_tmpa_tl
931 }{
932   % TODO
933 }
934 }
935
936 </package>

```

## Chapter 28

# STEX -Modules Implementation

```
937 <*package>
938
939 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
940
941 <@@=stex_modules>
    Warnings and error messages
942 \msg_new:nnn{stex}{error/unknownmodule}{
943   No~module~#1~found
944 }
945 \msg_new:nnn{stex}{error/syntax}{
946   Syntax~error:~#1
947 }
948 \msg_new:nnn{stex}{error/siglanguage}{
949   Module~#1~declares~signature~#2,~but~does~not~
950   declare~its~language
951 }
952
953 \msg_new:nnn{stex}{error/concllictingmodules}{
954   Comflicting~imports~for~module~#1
955 }

\l_stex_current_module_str The current module:
956 \str_new:N \l_stex_current_module_str
    (End definition for \l_stex_current_module_str. This variable is documented on page 26.)

\l_stex_all_modules_seq Stores all available modules
957 \seq_new:N \l_stex_all_modules_seq
    (End definition for \l_stex_all_modules_seq. This variable is documented on page 26.)

\stex_if_in_module_p:
\stex_if_in_module:TF
958 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
959   \str_if_empty:NTF \l_stex_current_module_str
960     \prg_return_false: \prg_return_true:
961 }
```

(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`

```
962 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
963   \prop_if_exist:cTF { c_stex_module_#1_prop }
964   \prg_return_true: \prg_return_false:
965 }
```

(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:

```
966 \cs_new_protected:Nn \stex_add_to_current_module:n {
967   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
968 }
969 \cs_new_protected:Npn \STEXexport {
970   \beginngroup
971   \newlinechar=-1\relax
972   \endlinechar=-1\relax
973   %\catcode'\ = 9\relax
974   \expandafter\endgroup\STEXexport:n
975 }
976 \cs_new_protected:Nn \STEXexport:n {
977   \ignorespaces #1
978   \stex_add_to_current_module:n { \ignorespaces #1 }
979   \stex_smsmode_set_codes:
980 }
981 \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`

```
982 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
983   \str_set:Nx \l_tmpa_str { #1 }
984   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
985 }
986
987 %\cs_new_protected:Nn \stex_add_field_to_current_module:n {
988 % \str_set:Nx \l_tmpa_str { #1 }
989 % \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _fields} { \l_tmpa_str }
990 %}
```

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

`\stex_collect_imports:n`

```
991 \cs_new_protected:Nn \stex_collect_imports:n {
992   \seq_clear:N \l_stex_collect_imports_seq
993   \__stex_modules_collect_imports:n {#1}
994 }
995 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
996   \seq_map_inline:cn {c_stex_module_#1_imports} {
997     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
998       \__stex_modules_collect_imports:n { ##1 }
999     }
1000 }
```

```

1000 }
1001 \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1002   \seq_put_right:Nn \l_stex_collect_imports_seq { #1 }
1003 }
1004 }

```

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

`\stex_add_import_to_current_module:n`

```

1005 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1006   \str_set:Nx \l_tmpa_str { #1 }
1007   \exp_args:Nno
1008   \seq_if_in:cnF{c_stex_module\_l_stex_current_module_str_imports}\l_tmpa_str{
1009     \seq_gput_right:co{c_stex_module\_l_stex_current_module_str_imports}\l_tmpa_str
1010   }
1011 }

```

(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).

```

1012 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
1013   \str_set:Nx \l_tmpa_str { #1 }
1014   \seq_set_eq:NN \l_tmpa_seq #2
1015   % split off file extension
1016   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1017   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1018   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1019   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1020
1021   \bool_set_true:N \l_tmpa_bool
1022   \bool_while_do:Nn \l_tmpa_bool {
1023     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1024     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1025       {source} { \bool_set_false:N \l_tmpa_bool }
1026     }{}{
1027       \seq_if_empty:NT \l_tmpa_seq {
1028         \bool_set_false:N \l_tmpa_bool
1029       }
1030     }
1031   }
1032
1033   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
1034   \str_if_empty:NTF \l_stex_modules_subpath_str {
1035     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
1036   }{
1037     \str_set:Nx \l_stex_modules_ns_str {
1038       \l_tmpa_str/\l_stex_modules_subpath_str
1039     }
1040   }
1041 }

```

(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
1042 \str_new:N \l_stex_modules_ns_str
1043 \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.

```

1044 \cs_new_protected:Nn \stex_modules_current_namespace: {
1045   \str_clear:N \l_stex_modules_subpath_str
1046   \prop_if_exist:NTF \l_stex_current_repository_prop {
1047     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1048     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1049   }{
1050     % split off file extension
1051     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1052     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1053     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1054     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1055     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1056     \str_set:Nx \l_stex_modules_ns_str {
1057       file:/\stex_path_to_string:N \l_tmpa_seq
1058     }
1059   }
1060 }

```

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

## 28.1 The module environment

module arguments:

```

1061 \keys_define:nn { stex / module } {
1062   title      .str_set_x:N = \l_stex_module_title_str ,
1063   ns         .str_set_x:N = \l_stex_module_ns_str ,
1064   lang       .str_set_x:N = \l_stex_module_lang_str ,
1065   sig        .str_set_x:N = \l_stex_module_sig_str ,
1066   creators   .str_set_x:N = \l_stex_module_creators_str ,
1067   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1068   meta       .str_set_x:N = \l_stex_module_meta_str ,
1069   srccite    .str_set_x:N = \l_stex_module_srccite_str
1070 }
1071
1072 \cs_new_protected:Nn \__stex_modules_args:n {
1073   \str_clear:N \l_stex_module_title_str
1074   \str_clear:N \l_stex_module_ns_str
1075   \str_clear:N \l_stex_module_lang_str
1076   \str_clear:N \l_stex_module_sig_str
1077   \str_clear:N \l_stex_module_creators_str
1078   \str_clear:N \l_stex_module_contributors_str
1079   \str_clear:N \l_stex_module_meta_str
1080   \str_clear:N \l_stex_module_srccite_str
1081   \keys_set:nn { stex / module } { #1 }

```



```

1082 }
1083
1084 % module parameters here? In the body?
1085

```

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

```

1086 \cs_new_protected:Nn \stex_module_setup:nn {
1087   \str_set:Nx \l_stex_module_name_str { #2 }
1088   \__stex_modules_args:n { #1 }

```

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

Are we in a nested module?

```

1089 \stex_if_in_module:TF {
1090   % Nested module
1091   \prop_get:cnN {c_stex_module\_l_stex_current_module_str _prop}
1092     { ns } \l_stex_module_ns_str
1093   \str_set:Nx \l_stex_module_name_str {
1094     \prop_item:cn {c_stex_module\_l_stex_current_module_str _prop}
1095       { name } / \l_stex_module_name_str
1096   }
1097 }{
1098   % not nested:
1099   \str_if_empty:NT \l_stex_module_ns_str {
1100     \stex_modules_current_namespace:
1101     \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1102     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1103       / {\l_stex_module_ns_str}
1104     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1105     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1106       \str_set:Nx \l_stex_module_ns_str {
1107         \stex_path_to_string:N \l_tmpa_seq
1108       }
1109     }
1110   }
1111 }

```

Next, we determine the language of the module:

```

1112 \str_if_empty:NT \l_stex_module_lang_str {
1113   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1114   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1115   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1116   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1117   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1118     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1119       inferred~from~file~name}
1120     \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1121   }
1122 }
1123
1124 \str_if_empty:NF \l_stex_module_lang_str {
1125   \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1126     \l_tmpa_str {
1127     \ltx@ifpackageloaded{babel}{
1128       \exp_args:Nx \selectlanguage { \l_tmpa_str }

```

```

1129     }{}
1130   } {
1131     \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
1132   }
1133 }

```

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

```

1134 \str_if_empty:NTF \l_stex_module_sig_str {
1135   \exp_args:Nnx \prop_gset_from_keyval:cn {
1136     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1137   } {
1138     name      = \l_stex_module_name_str ,
1139     ns        = \l_stex_module_ns_str ,
1140     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1141     lang      = \l_stex_module_lang_str ,
1142     sig       = \l_stex_module_sig_str ,
1143     meta      = \l_stex_module_meta_str
1144   }
1145   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1146   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _fields}
1147   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1148   \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1149   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1150 \str_if_empty:NT \l_stex_module_meta_str {
1151   \str_set:Nx \l_stex_module_meta_str {
1152     \c_stex_metatheory_ns_str ? Metatheory
1153   }
1154 }
1155 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1156   \bool_set_true:N \l_stex_in_meta_bool
1157   \exp_args:Nx \stex_add_to_current_module:n {
1158     \bool_set_true:N \l_stex_in_meta_bool
1159     \stex_activate_module:n {\l_stex_module_meta_str}
1160     \bool_set_false:N \l_stex_in_meta_bool
1161   }
1162   \stex_activate_module:n {\l_stex_module_meta_str}
1163   \bool_set_false:N \l_stex_in_meta_bool
1164 }
1165 }{
1166   \str_if_empty:NT \l_stex_module_lang_str {
1167     \msg_error:nnxx{stex}{error/siglanguage}{
1168       \l_stex_module_ns_str?\l_stex_module_name_str
1169     }{\l_stex_module_sig_str}
1170   }
1171
1172   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1173   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1174   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1175   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1176   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1177   \str_set:Nx \l_tmpa_str {

```

```

1178     \stex_path_to_string:N \l_tmpa_seq /
1179     \l_tmpa_str . \l_stex_module_sig_str .tex
1180 }
1181 \IfFileExists \l_tmpa_str {
1182     \exp_args:No \stex_in_smsmode:nn { \l_tmpa_str } {
1183         \seq_clear:N \l_stex_all_modules_seq
1184         %\prop_clear:N \l_stex_current_module_prop
1185         \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1186         \input { \l_tmpa_str }
1187     }
1188 }{
1189     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1190 }
1191 \stex_activate_module:n {
1192     \l_stex_module_ns_str ? \l_stex_module_name_str
1193 }
1194 %\prop_set_eq:Nc \l_stex_current_module_prop {
1195 % c_stex_module_
1196 % \l_stex_module_ns_str ?
1197 % \l_stex_module_name_str
1198 % _prop
1199 %}
1200 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1201 }
1202 }

```

(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}

1203 \cs_new_protected:Nn \__stex_modules_begin_module:nn {
1204     \stex_reactivate_macro:N \STEXexport
1205     \stex_reactivate_macro:N \importmodule
1206     \stex_reactivate_macro:N \symdecl
1207     \stex_reactivate_macro:N \notation
1208     \stex_reactivate_macro:N \symdef
1209     \stex_module_setup:nn{#1}{#2}
1210
1211     \stex_debug:nn{modules}{
1212         New~module:\\
1213         Namespace:~\l_stex_module_ns_str\\
1214         Name:~\l_stex_module_name_str\\
1215         Language:~\l_stex_module_lang_str\\
1216         Signature:~\l_stex_module_sig_str\\
1217         Metatheory:~\l_stex_module_meta_str\\
1218         File:~\stex_path_to_string:N \g_stex_currentfile_seq
1219     }
1220
1221     \seq_put_right:Nx \l_stex_all_modules_seq {
1222         \l_stex_module_ns_str ? \l_stex_module_name_str
1223     }
1224
1225     % \seq_gput_right:Nx \g_stex_modules_in_file_seq

```

```

1226 %      { \l_stex_module_ns_str ? \l_stex_module_name_str }
1227
1228 \stex_if_smsmode:TF {
1229   \stex_smsmode_set_codes:
1230 } {
1231   \begin{stex_annotate_env} {theory} {
1232     \l_stex_module_ns_str ? \l_stex_module_name_str
1233   }
1234
1235   \stex_annotate_invisible:nnn{header}{} {
1236     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1237     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1238     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1239       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1240     }
1241   }
1242 }
1243 % TODO: Inherit metatheory for nested modules?
1244 }
1245 \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}`

```

1246 \cs_new_protected:Nn \_stex_modules_end_module: {
1247 %   \str_set:Nx \l_tmpa_str {
1248 %     c_stex_module_
1249 %     \prop_item:Nn \l_stex_current_module_prop { ns } ?
1250 %     \prop_item:Nn \l_stex_current_module_prop { name }
1251 %     _prop
1252 %   }
1253 %^^A \prop_new:c { \l_tmpa_str }
1254 %   \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
1255   \stex_debug:nn{modules}{Closing module~\prop_item:cn {c_stex_module_\l_stex_current_module_
1256 }

```

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

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

```

1257 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1258 \NewDocumentEnvironment { @module } { 0 } { m } {
1259   \par
1260   \_stex_modules_begin_module:nn{#1}{#2}
1261 } {
1262   \_stex_modules_end_module:
1263   \stex_if_smsmode:TF {
1264 %     \exp_args:Nx \stex_add_to_sms:n {
1265 %       \prop_gset_from_keyval:cn {
1266 %         c_stex_module_
1267 %         \prop_item:Nn \l_stex_current_module_prop { ns } ?
1268 %         \prop_item:Nn \l_stex_current_module_prop { name }
1269 %         _prop
1270 %       } {
1271 %         name          = \prop_item:cn { \l_tmpa_str } { name } ,

```

```

1272 %      ns      = \prop_item:cn { \l_tmpa_str } { ns } ,
1273 %      file     = \prop_item:cn { \l_tmpa_str } { file } ,
1274 %      lang     = \prop_item:cn { \l_tmpa_str } { lang } ,
1275 %      sig      = \prop_item:cn { \l_tmpa_str } { sig } ,
1276 %      meta     = \prop_item:cn { \l_tmpa_str } { meta }
1277 %    }
1278 %  }
1279 }{
1280   \end{stex_annotate_env}
1281 }
1282 }

```

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

```

1283 \cs_if_exist:NTF \thesection {
1284   \newcounter{module}[section]
1285 }{
1286   \newcounter{module}
1287 }
1288
1289 \bool_if:NT \c_stex_showmods_bool {
1290   \latexml_if:F { \RequirePackage{mdframed} }
1291 }
1292
1293 \cs_new_protected:Nn \stex_modules_heading: {
1294   \stepcounter{module}
1295   \par
1296   \bool_if:NT \c_stex_showmods_bool {
1297     \noindent{\textbf{Module} ~
1298       \cs_if_exist:NT \thesection {\thesection.}
1299       \themodule ~ [\l_stex_module_name_str]
1300     }
1301     \str_if_empty:NTF \l_stex_module_title_str {
1302       }{
1303         \quad(\l_stex_module_title_str)\hfill
1304       }\par
1305     }
1306     \edef\@currentlabel{Module~\thesection.\themodule~[\l_stex_module_name_str]}
1307     % TODO
1308     \stex_ref_new_doc_target:n \l_stex_module_name_str
1309   }

```

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

Finally:

```

1310 \NewDocumentEnvironment { module } { 0{} m } {
1311   \bool_if:NT \c_stex_showmods_bool {
1312     \begin{mdframed}
1313   }
1314   \begin{@module}[#1]{#2}
1315   \stex_modules_heading:
1316 }{
1317   \end{@module}
1318   \bool_if:NT \c_stex_showmods_bool {
1319     \end{mdframed}
1320   }

```

1321 }

## 28.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n 1322 \NewDocumentCommand \STEXModule { m } {
1323   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1324   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1325   \tl_set:Nn \l_tmpa_tl {
1326     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1327   }
1328   \seq_map_inline:Nn \l_stex_all_modules_seq {
1329     \str_set:Nn \l_tmpb_str { ##1 }
1330     \str_if_eq:eeT { \l_tmpa_str } {
1331       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1332     } {
1333       \seq_map_break:n {
1334         \tl_set:Nn \l_tmpa_tl {
1335           \stex_invoke_module:n { ##1 }
1336         }
1337       }
1338     }
1339   }
1340   \l_tmpa_tl
1341 }
1342
1343 \cs_new_protected:Nn \stex_invoke_module:n {
1344   \stex_debug:nn{modules}{Invoking~module~#1}
1345   \peek_charcode_remove:NTF ! {
1346     \__stex_modules_invoke_uri:nN { #1 }
1347   } {
1348     \peek_charcode_remove:NTF ? {
1349       \__stex_modules_invoke_symbol:nn { #1 }
1350     } {
1351       \msg_error:nnx{stex}{error/syntax}{
1352         ?~or~!~expected~after~
1353         \c_backslash_str STEXModule{#1}
1354       }
1355     }
1356   }
1357 }
1358
1359 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1360   \str_set:Nn #2 { #1 }
1361 }
1362
1363 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1364   \stex_invoke_symbol:n{#1?#2}
1365 }

```

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

`\stex_activate_module:n`

```
1366 \bool_new:N \l_stex_in_meta_bool
1367 \bool_set_false:N \l_stex_in_meta_bool
1368 \cs_new_protected:Nn \stex_activate_module:n {
1369   \stex_debug:nn{modules}{Activating~module~#1}
1370   \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1371     \msg_error:nnn{stex}{error/conclictingmodules}{ #1 }
1372   }
1373   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1374     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1375     \use:c{ c_stex_module_#1_code }
1376   }
1377 }
```

*(End definition for \stex\_activate\_module:n. This function is documented on page 30.)*

```
1378 </package>
```

## Chapter 29

# sTeX -Module Inheritance Implementation

```
1379 <*package>
1380
1381 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1382
```

### 29.1 SMS Mode

```
1383 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1384 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1385 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1386 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1387
1388 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1389   \makeatletter
1390   \makeatother
1391   \ExplSyntaxOn
1392   \ExplSyntaxOff
1393 }
1394
1395 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1396   \symdef
1397   \importmodule
1398   \notation
1399   \symdecl
1400   \STEXexport
1401 }
1402
1403 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1404   \tl_to_str:n {
1405     module,
1406     @module
```



```
1407 }
1408 }
```

(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
1409 \bool_new:N \g__stex_smsmode_bool
1410 \bool_set_false:N \g__stex_smsmode_bool
1411 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1412   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1413 }
```

(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
1414 \bool_new:N \g__stex_smsmode_catcode_bool
1415 \bool_set_false:N \g__stex_smsmode_catcode_bool
1416 \prg_new_conditional:Nnn \_stex_smsmode_if_catcodes: { p, T, F, TF } {
1417   \bool_if:NTF \g__stex_smsmode_catcode_bool
1418   \prg_return_true: \prg_return_false:
1419 }
```

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

```
\stex_smsmode_set_codes:
1420 \cs_new_protected:Nn \stex_smsmode_set_codes: {
1421   \stex_if_smsmode:T {
1422     \_stex_smsmode_if_catcodes:F {
1423       \bool_gset_true:N \g__stex_smsmode_catcode_bool
1424       \exp_after:wN \char_gset_active_eq:NN
1425       \c_backslash_str \_stex_smsmode_cs:
1426       \tex_global:D \char_set_catcode_active:N \
1427       \tex_global:D \char_set_catcode_other:N $
1428       \tex_global:D \char_set_catcode_other:N ^
1429       \tex_global:D \char_set_catcode_other:N _
1430       \tex_global:D \char_set_catcode_other:N &
1431       \tex_global:D \char_set_catcode_other:N ##
1432     }
1433   }
1434 } \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.
1435 \cs_new_protected:Nn \_stex_smsmode_unset_codes: {
1436   \_stex_smsmode_if_catcodes:T {
1437     \bool_gset_false:N \g__stex_smsmode_catcode_bool
1438     \exp_after:wN \tex_global:D \exp_after:wN
1439     \char_set_catcode_escape:N \c_backslash_str
1440     \tex_global:D \char_set_catcode_math_toggle:N $
1441     \tex_global:D \char_set_catcode_math_superscript:N ^
1442     \tex_global:D \char_set_catcode_math_subscript:N _
1443     \tex_global:D \char_set_catcode_alignment:N &
1444     \tex_global:D \char_set_catcode_parameter:N ##
1445   }
1446 } \iffalse $ \fi % to make syntax highlighting work again
```

(End definition for `\_stex_smsmode_unset_codes:`.)

`\stex_in_smsmode:nn`

```

1447 \cs_new_protected:Nn \stex_in_smsmode:nn {
1448   \vbox_set:Nn \l_tmpa_box {
1449     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1450     \bool_gset_true:N \g__stex_smsmode_bool
1451     \stex_smsmode_set_codes:
1452     #2
1453     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1454     \stex_if_smsmode:F {
1455       \__stex_smsmode_unset_codes:
1456     }
1457   }
1458   \box_clear:N \l_tmpa_box
1459 }

```

(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:

```

1460 \cs_new_protected:Nn \_stex_smsmode_cs: {
1461   \str_clear:N \l_tmpa_str
1462   \peek_analysis_map_inline:n {
1463     % #1: token (one expansion)
1464     % #2: charcode
1465     % #3 catcode
1466     \token_if_eq_charcode:NNTF ##3 B {
1467       % token is a letter
1468       \exp_args:NNNo \str_put_right:Nn \l_tmpa_str { ##1 }
1469     } {
1470       \str_if_empty:NTF \l_tmpa_str {
1471         % we don't allow (or need) single non-letter CSs
1472         % for now
1473         \peek_analysis_map_break:
1474       }{
1475         \str_if_eq:onTF \l_tmpa_str { begin } {
1476           \peek_analysis_map_break:n {
1477             \exp_after:wN \_stex_smsmode_checkbegin:n ##1
1478           }
1479         } {
1480           \str_if_eq:onTF \l_tmpa_str { end } {
1481             \peek_analysis_map_break:n {
1482               \exp_after:wN \_stex_smsmode_checkend:n ##1
1483             }
1484           } {
1485             \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
1486             \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1487             \g_stex_smsmode_allowedmacros_tl
1488             { \use:c{\l_tmpa_str} } {
1489               \stex_debug:nn{modules}{Executing-1:~\l_tmpa_str}
1490               \peek_analysis_map_break:n {
1491                 \exp_after:wN \l_tmpa_tl ##1
1492               }
1493             }
1494           }
1495         }
1496       }
1497     }
1498   }
1499 }

```

```

1493     } {
1494         \exp_args:NNo \exp_args:NNo \tl_if_in:NnTF
1495         \g_stex_smsmode_allowedmacros_escape_tl
1496         { \use:c{\l_tmpa_str} } {
1497             \__stex_smsmode_unset_codes:
1498             \stex_debug:nn{modules}{Executing~2:~\l_tmpa_str}
1499             % TODO \__stex_smsmode_rescan_cs:
1500             \int_compare:nNnTF {##2} = {92} {
1501                 \peek_analysis_map_break:n {
1502                     \__stex_smsmode_unset_codes:
1503                     \__stex_smsmode_rescan_cs:
1504                 }
1505             } {
1506                 \peek_analysis_map_break:n {
1507                     \exp_after:wN \l_tmpa_tl ##1
1508                 }
1509             }
1510         } {
1511             \int_compare:nNnTF {##2} = {92} {
1512                 \peek_analysis_map_break:n { \__stex_smsmode_cs: }
1513             } {
1514                 \peek_analysis_map_break:n { \exp_after:wN\relax ##1 }
1515             }
1516         }
1517     }
1518 }
1519 }
1520 }
1521 }
1522 }
1523 }

```

(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:

```

1524 \cs_new_protected:Nn \__stex_smsmode_rescan_cs: {
1525     \str_clear:N \l_tmpb_str
1526     \peek_analysis_map_inline:n {
1527         \token_if_eq_charcode:NNTF ##3 B {
1528             % token is a letter
1529             \exp_args:NNo \str_put_right:Nn \l_tmpb_str { ##1 }
1530         } {
1531             \peek_analysis_map_break:n {
1532                 \exp_after:wN \use:c \exp_after:wN {
1533                     \exp_after:wN \l_tmpa_str\exp_after:wN
1534                 } \use:c { \l_tmpb_str \exp_after:wN } ##1
1535             }
1536         }
1537     }
1538 }

```

(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.

```

1539 \cs_new_protected:Nn \__stex_smsmode_checkbegin:n {
1540   \str_set:Nn \l_tmpa_str { #1 }
1541   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1542     \__stex_smsmode_unset_codes:
1543     \begin{#1}
1544   }
1545 }
```

(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.

```

1552 \cs_new_protected:Nn \__stex_smsmode_checkend:n {
1553   \str_set:Nn \l_tmpa_str { #1 }
1554   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1555     \end{#1}
1556   }
1557 }
```

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

## 29.2 Inheritance

1552 `<@@=stex_importmodule>`

`\stex_import_module_uri:nn`

```

1553 \cs_new_protected:Nn \stex_import_module_uri:nn {
1554   \str_set:Nx \l_stex_import_archive_str { #1 }
1555   \str_set:Nn \l_stex_import_path_str { #2 }
1556
1557   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1558   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1559   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1560
1561   \stex_modules_current_namespace:
1562   \bool_lazy_all:nTF {
1563     {\str_if_empty_p:N \l_stex_import_archive_str}
1564     {\str_if_empty_p:N \l_stex_import_path_str}
1565     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1566   }{
1567     \str_set_eq:NN \l_stex_import_path_str \l_stex_modules_subpath_str
1568     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1569   }{
1570     \str_if_empty:NT \l_stex_import_archive_str {
1571       \prop_if_exist:NT \l_stex_current_repository_prop {
1572         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1573       }
1574     }
1575     \str_if_empty:NTF \l_stex_import_archive_str {
1576       \str_if_empty:NF \l_stex_import_path_str {
1577         \str_set:Nx \l_stex_import_ns_str {
1578           \l_stex_module_ns_str / \l_stex_import_path_str
1579         }
1580       }
1581     }
```

```

1581   }{
1582     \stex_require_repository:n \l_stex_import_archive_str
1583     \prop_get:cnN { c_stex_mathhub\_l_stex_import_archive_str _manifest_prop } { ns }
1584     \l_stex_import_ns_str
1585     \str_if_empty:NF \l_stex_import_path_str {
1586       \str_set:Nx \l_stex_import_ns_str {
1587         \l_stex_import_ns_str / \l_stex_import_path_str
1588       }
1589     }
1590   }
1591 }
1592 }

```

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

<code>\l_stex_import_name_str</code>	Store the return values of <code>\stex_import_module_uri:nn</code> .
<code>\l_stex_import_archive_str</code>	1593 <code>\str_new:N \l_stex_import_name_str</code>
<code>\l_stex_import_path_str</code>	1594 <code>\str_new:N \l_stex_import_archive_str</code>
<code>\l_stex_import_ns_str</code>	1595 <code>\str_new:N \l_stex_import_path_str</code>
	1596 <code>\str_new:N \l_stex_import_ns_str</code>

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

```

\stex_import_require_module:nnnn    {<ns>} {<archive-ID>} {<path>} {<name>}
1597 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1598   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1599
1600     % archive
1601     \str_set:Nx \l_tmpa_str { #2 }
1602     \str_if_empty:NTF \l_tmpa_str {
1603       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1604     } {
1605       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1606       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1607       \seq_put_right:Nn \l_tmpa_seq { source }
1608     }
1609
1610     % path
1611     \str_set:Nx \l_tmpb_str { #3 }
1612     \str_if_empty:NTF \l_tmpb_str {
1613       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1614
1615       \ltx@ifpackageloaded{babel} {
1616         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1617           { \language } \l_tmpb_str {
1618           \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1619         }
1620       } {
1621         \str_clear:N \l_tmpb_str
1622       }
1623
1624       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1625       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1626         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }

```

```

1627 }{
1628   \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1629   \IfFileExists{ \l_tmpa_str.tex }{
1630     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1631   }{
1632     % try english as default
1633     \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1634     \IfFileExists{ \l_tmpa_str.en.tex }{
1635       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1636     }{
1637       \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1638     }
1639   }
1640 }
1641
1642 } {
1643   \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1644   \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1645
1646   \ltx@ifpackageloaded{babel} {
1647     \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1648       { \language } \l_tmpb_str {
1649       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1650     }
1651   } {
1652     \str_clear:N \l_tmpb_str
1653   }
1654
1655   \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1656
1657   \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1658   \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1659     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1660   }{
1661     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1662     \IfFileExists{ \l_tmpa_str/#4.tex }{
1663       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1664     }{
1665       % try english as default
1666       \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1667       \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1668         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1669       }{
1670         \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1671         \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1672           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1673         }{
1674           \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1675           \IfFileExists{ \l_tmpa_str.tex }{
1676             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1677           }{
1678             % try english as default
1679             \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1680             \IfFileExists{ \l_tmpa_str.en.tex }{

```

```

1681         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1682     }{
1683         \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1684     }
1685 }
1686 }
1687 }
1688 }
1689 }
1690 }
1691
1692 \exp_args:No \stex_in_smsmode:nn { \g__stex_importmodule_file_str } {
1693     \seq_clear:N \l_stex_all_modules_seq
1694     \str_clear:N \l_stex_current_module_str
1695     \str_set:Nx \l_tmpb_str { #2 }
1696     \str_if_empty:NF \l_tmpb_str {
1697         \stex_set_current_repository:n { #2 }
1698     }
1699     \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1700     \input { \g__stex_importmodule_file_str }
1701 }
1702
1703 \stex_if_module_exists:nF { #1 ? #4 } {
1704     \msg_error:nnx{stex}{error/unknownmodule}{
1705         #1?#4~(in~file~\g__stex_importmodule_file_str)
1706     }
1707 }
1708 }
1709 \stex_activate_module:n { #1 ? #4 }
1710 }

```

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

## `\importmodule`

```

1711 \NewDocumentCommand \importmodule { 0{} m } {
1712     \stex_import_module_uri:nn { #1 } { #2 }
1713     \stex_debug:nn{modules}{Importing~module:~
1714         \l_stex_import_ns_str ? \l_stex_import_name_str
1715     }
1716     \stex_if_smsmode:F {
1717         \stex_import_require_module:nnnn
1718         { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1719         { \l_stex_import_path_str } { \l_stex_import_name_str }
1720         \stex_annotate_invisible:nnn
1721         {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1722     }
1723     \exp_args:Nx \stex_add_to_current_module:n {
1724         \stex_import_require_module:nnnn
1725         { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1726         { \l_stex_import_path_str } { \l_stex_import_name_str }
1727     }
1728     \exp_args:Nx \stex_add_import_to_current_module:n {
1729         \l_stex_import_ns_str ? \l_stex_import_name_str
1730     }

```

```

1731 \stex_smsmode_set_codes:
1732 }
1733 \stex_deactivate_macro:Nn \importmodule {module-environments}

```

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

## **\usemodule**

```

1734 \NewDocumentCommand \usemodule { 0{} m } {
1735   \stex_if_smsmode:F {
1736     \stex_import_module_uri:nn { #1 } { #2 }
1737     \stex_import_require_module:nnnn
1738     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1739     { \l_stex_import_path_str } { \l_stex_import_name_str }
1740     \stex_annotate_invisible:nnn
1741     {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1742   }
1743   \stex_smsmode_set_codes:
1744 }

```

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

```

1745 \endpackage

```



## Chapter 30

# STEX -Symbols Implementation

```
1746 <*package>
1747
1748 %%%%%%%%%%%%% symbols.dtx %%%%%%%%%%%%%
1749
```

Warnings and error messages

```
1750
```

### 30.1 Symbol Declarations

```
1751 <@@=stex_symdecl>
```

`\l_stex_all_symbols_seq` Stores all available symbols

```
1752 \seq_new:N \l_stex_all_symbols_seq
```

(End definition for `\l_stex_all_symbols_seq`. This variable is documented on page 36.)

`\STEXsymbol`

```
1753 \NewDocumentCommand \STEXsymbol { m } {
1754   \stex_get_symbol:n { #1 }
1755   \exp_args:No
1756   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1757 }
```

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

symdecl arguments:

```
1758 \keys_define:nn { stex / symdecl } {
1759   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1760   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1761   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1762   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
1763   align     .str_set:N    = \l_stex_symdecl_align_str , % TODO(?)
1764   gfc       .str_set:N    = \l_stex_symdecl_gfc_str , % TODO(?)
1765   specializes .str_set:N  = \l_stex_symdecl_specializes_str , % TODO(?)
1766   def       .tl_set:N    = \l_stex_symdecl_definiens_tl
1767 }
```

```

1768
1769 \bool_new:N \l_stex_symdecl_make_macro_bool
1770
1771 \cs_new_protected:Nn \__stex_symdecl_args:n {
1772   \str_clear:N \l_stex_symdecl_name_str
1773   \str_clear:N \l_stex_symdecl_args_str
1774   \bool_set_false:N \l_stex_symdecl_local_bool
1775   \tl_clear:N \l_stex_symdecl_type_tl
1776   \tl_clear:N \l_stex_symdecl_definiens_tl
1777
1778   \keys_set:nn { stex / symdecl } { #1 }
1779 }

```

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

```

1780
1781 \NewDocumentCommand \symdecl { s O{} m } {
1782   \__stex_symdecl_args:n { #2 }
1783   \IfBooleanTF #1 {
1784     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1785   } {
1786     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1787   }
1788   \stex_symdecl_do:n { #3 }
1789   \stex_smsmode_set_codes:
1790 }
1791 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

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

```

1792 \cs_new_protected:Nn \stex_symdecl_do:n {
1793   \stex_if_in_module:F {
1794     % TODO throw error? some default namespace?
1795   }
1796
1797   \str_if_empty:NT \l_stex_symdecl_name_str {
1798     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1799   }
1800
1801   \prop_if_exist:cT { l_stex_symdecl_
1802     \l_stex_current_module_str ?
1803     \l_stex_symdecl_name_str
1804     _prop
1805   }{
1806     % TODO throw error (beware of circular dependencies)
1807   }
1808
1809   \prop_clear:N \l_tmpa_prop
1810   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
1811   \seq_clear:N \l_tmpa_seq
1812   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1813   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1814

```

```

1815 \exp_args:No \stex_add_constant_to_current_module:n {
1816   \l_stex_symdecl_name_str
1817 }
1818
1819 % arity/args
1820 \int_zero:N \l_tmpb_int
1821
1822 \bool_set_true:N \l_tmpa_bool
1823 \str_map_inline:Nn \l_stex_symdecl_args_str {
1824   \token_case_meaning:NnF ##1 {
1825     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1826     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1827     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1828     {\tl_to_str:n a} {
1829       \bool_set_false:N \l_tmpa_bool
1830       \int_incr:N \l_tmpb_int
1831     }
1832     {\tl_to_str:n B} {
1833       \bool_set_false:N \l_tmpa_bool
1834       \int_incr:N \l_tmpb_int
1835     }
1836   }{
1837     \msg_set:nnn{stex}{error/wrongargs}{
1838       args~value~in~symbol~declaration~for~
1839       \l_stex_current_module_str ?
1840       \l_stex_symdecl_name_str ~
1841       needs~to~be~
1842       i,~a,~b~or~B,~but~##1~given
1843     }
1844     \msg_error:nn{stex}{error/wrongargs}
1845   }
1846 }
1847 \bool_if:NTF \l_tmpa_bool {
1848   % possibly numeric
1849   \str_if_empty:NTF \l_stex_symdecl_args_str {
1850     \prop_put:Nnn \l_tmpa_prop { args } {}
1851     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1852   }{
1853     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1854     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1855     \str_clear:N \l_tmpa_str
1856     \int_step_inline:nn \l_tmpa_int {
1857       \str_put_right:Nn \l_tmpa_str i
1858     }
1859     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1860   }
1861 } {
1862   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1863   \prop_put:Nnx \l_tmpa_prop { arity }
1864     { \str_count:N \l_stex_symdecl_args_str }
1865 }
1866 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
1867
1868

```

```

1869 % semantic macro
1870
1871 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1872   \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1873     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1874   } }
1875
1876   \bool_if:NF \l_stex_symdecl_local_bool {
1877     \exp_args:Nx \stex_add_to_current_module:n {
1878       \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1879         \l_stex_current_module_str ? \l_stex_symdecl_name_str
1880       } }
1881     }
1882   }
1883 }
1884
1885 % add to all symbols
1886
1887 \bool_if:NF \l_stex_symdecl_local_bool {
1888   \exp_args:Nx \stex_add_to_current_module:n {
1889     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1890       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1891     }
1892   }
1893 %   \exp_args:Nx \stex_add_field_to_current_module:n {
1894 %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1895 %   }
1896 }
1897
1898 \stex_debug:nn{symbols}{New~symbol:~
1899   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
1900   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
1901   Args:~\prop_item:Nn \l_tmpa_prop { args }
1902 }
1903
1904 % circular dependencies require this:
1905
1906 \prop_if_exist:cF {
1907   l_stex_symdecl_
1908   \l_stex_current_module_str ? \l_stex_symdecl_name_str
1909   _prop
1910 } {
1911   \prop_set_eq:cN {
1912     l_stex_symdecl_
1913     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1914     _prop
1915   } \l_tmpa_prop
1916 }
1917
1918 \seq_clear:c {
1919   l_stex_symdecl_
1920   \l_stex_current_module_str ? \l_stex_symdecl_name_str
1921   _notations
1922 }

```

```

1923
1924 \bool_if:NF \l_stex_symdecl_local_bool {
1925   \exp_args:Nx
1926   \stex_add_to_current_module:n {
1927     \seq_clear:c {
1928       l_stex_symdecl_
1929       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1930       _notations
1931     }
1932     \prop_set_from_keyval:cn {
1933       l_stex_symdecl_
1934       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1935       _prop
1936     } {
1937       name      = \prop_item:Nn \l_tmpa_prop { name }      ,
1938       module    = \prop_item:Nn \l_tmpa_prop { module }    ,
1939       type      = \prop_item:Nn \l_tmpa_prop { type }      ,
1940       args      = \prop_item:Nn \l_tmpa_prop { args }      ,
1941       arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
1942       assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
1943     }
1944   }
1945 }
1946
1947 \stex_if_smsmode:TF {
1948   \bool_if:NF \l_stex_symdecl_local_bool {
1949     % \exp_args:Nx \stex_add_to_sms:n {
1950     %   \prop_set_from_keyval:cn {
1951     %     l_stex_symdecl_
1952     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1953     %     _prop
1954     %   } {
1955     %     name      = \prop_item:Nn \l_tmpa_prop { name }      ,
1956     %     module    = \prop_item:Nn \l_tmpa_prop { module }    ,
1957     %     local     = \prop_item:Nn \l_tmpa_prop { local }     ,
1958     %     type      = \prop_item:Nn \l_tmpa_prop { type }      ,
1959     %     args      = \prop_item:Nn \l_tmpa_prop { args }      ,
1960     %     arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
1961     %     assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
1962     %   }
1963     %   \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1964     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1965     %   }
1966     % }
1967   }
1968 }{
1969   \exp_args:NNx \seq_put_right:Nn \l_stex_all_symbols_seq {
1970     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1971   }
1972   \stex_if_do_html:T {
1973     \stex_annotate_invisible:nnn {symdecl} {
1974       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1975     } {
1976       \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}{}}{${\l_st

```

```

1977     \stex_annotate_invisible:nnn{args}{}{
1978     \prop_item:Nn \l_tmpa_prop { args }
1979   }
1980   \stex_annotate_invisible:nnn{macroname}{#1}{}
1981   \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
1982     \stex_annotate_invisible:nnn{definiens}{}
1983     { $\l_stex_symdecl_definiens_tl$ }
1984   }
1985 }
1986 }
1987 }
1988 }

```

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

`\stex_get_symbol:n`

```

1989 \str_new:N \l_stex_get_symbol_uri_str
1990
1991 \cs_new_protected:Nn \stex_get_symbol:n {
1992   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
1993     \__stex_symdecl_get_symbol_from_cs:n { #1 }
1994   }{
1995     % argument is a string
1996     % is it a command name?
1997     \cs_if_exist:cTF { #1 }{
1998       \cs_set_eq:Nc \l_tmpa_tl { #1 }
1999       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2000       \str_if_empty:NNTF \l_tmpa_str {
2001         \exp_args:Nx \cs_if_eq:NNTF {
2002           \tl_head:N \l_tmpa_tl
2003         } \stex_invoke_symbol:n {
2004           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
2005         }{
2006           \__stex_symdecl_get_symbol_from_string:n { #1 }
2007         }
2008       } {
2009         \__stex_symdecl_get_symbol_from_string:n { #1 }
2010       }
2011     }{
2012       % argument is not a command name
2013       \__stex_symdecl_get_symbol_from_string:n { #1 }
2014       % \l_stex_all_symbols_seq
2015     }
2016   }
2017 }
2018
2019 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2020   \str_set:Nn \l_tmpa_str { #1 }
2021   \bool_set_false:N \l_tmpa_bool
2022   \stex_if_in_module:T {
2023     \exp_args:Nno \seq_if_in:cnT {c_stex_module_\l_stex_current_module_str _constants} { \l_
2024       \bool_set_true:N \l_tmpa_bool
2025       \str_set:Nx \l_stex_get_symbol_uri_str {
2026         \l_stex_current_module_str ? #1

```

```

2027     }
2028   }
2029 }
2030 \bool_if:NF \l_tmpa_bool {
2031   \tl_set:Nn \l_tmpa_tl {
2032     \msg_set:nnn{stex}{error/unknownsymbol}{
2033       No~symbol~#1~found!
2034     }
2035     \msg_error:nn{stex}{error/unknownsymbol}
2036   }
2037   \str_set:Nn \l_tmpa_str { #1 }
2038   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2039   \seq_map_inline:Nn \l_stex_all_symbols_seq {
2040     \str_set:Nn \l_tmpb_str { ##1 }
2041     \str_if_eq:eeT { \l_tmpa_str } {
2042       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
2043     } {
2044       \seq_map_break:n {
2045         \tl_set:Nn \l_tmpa_tl {
2046           \str_set:Nn \l_stex_get_symbol_uri_str {
2047             ##1
2048           }
2049         }
2050       }
2051     }
2052   }
2053   \l_tmpa_tl
2054 }
2055 }
2056
2057 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
2058   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2059   { \tl_tail:N \l_tmpa_tl }
2060   \tl_if_single:NTF \l_tmpa_tl {
2061     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2062       \exp_after:wN \str_set:Nn \exp_after:wN
2063       \l_stex_get_symbol_uri_str \l_tmpa_tl
2064     }{
2065       % TODO
2066       % tail is not a single group
2067     }
2068   }{
2069     % TODO
2070     % tail is not a single group
2071   }
2072 }

```

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

## 30.2 Notations

```

2073 <@@=stex_notation>

```

notation arguments:

```

2074 \keys_define:nn { stex / notation } {
2075   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2076   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2077   prec .str_set_x:N = \l__stex_notation_prec_str ,
2078   op .tl_set:N = \l__stex_notation_op_tl ,
2079   primary .bool_set:N = \l__stex_notation_primary_bool ,
2080   primary .default:n = {true} ,
2081   unknown .code:n = \str_set:Nx
2082     \l__stex_notation_variant_str \l_keys_key_str
2083 }
2084
2085 \cs_new_protected:Nn \stex_notation_args:n {
2086   \str_clear:N \l__stex_notation_lang_str
2087   \str_clear:N \l__stex_notation_variant_str
2088   \str_clear:N \l__stex_notation_prec_str
2089   \tl_clear:N \l__stex_notation_op_tl
2090   \bool_set_false:N \l__stex_notation_primary_bool
2091
2092   \keys_set:nn { stex / notation } { #1 }
2093 }

```

**\notation**

```

2094 \NewDocumentCommand \notation { 0{ } m } {
2095   \stex_notation_args:n { #1 }
2096   \tl_clear:N \l_stex_symdecl_definiens_tl
2097   \stex_get_symbol:n { #2 }
2098   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
2099 }
2100 \stex_deactivate_macro:Nn \notation {module~environments}

```

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

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

```

2101 \cs_new_protected:Nn \stex_notation_do:nn {
2102   \let\l_stex_current_symbol_str\relax
2103   \prop_set_eq:Nc \l_tmpa_prop {
2104     \l_stex_symdecl_#1_prop
2105   }
2106
2107   \prop_clear:N \l_tmpb_prop
2108   \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
2109   \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
2110   \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
2111
2112   % precedences
2113   \seq_clear:N \l_tmpb_seq
2114   \exp_args:NNno
2115   \str_if_empty:NTF \l__stex_notation_prec_str {
2116     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2117     \int_compare:nNnTF \l_tmpa_str = 0 {
2118       \exp_args:NNnx
2119       \prop_put:Nno \l_tmpb_prop { opprec }
2120       { \neginfprec }
2121     }{
2122       \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }

```



```

2123     }
2124   } {
2125     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2126       \exp_args:NNnx
2127       \prop_put:Nno \l_tmpb_prop { opprec }
2128       { \neginfprec }
2129       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2130       \int_step_inline:nn { \l_tmpa_str } {
2131         \exp_args:NNx
2132         \seq_put_right:Nn \l_tmpb_seq { \infprec }
2133       }
2134     }{
2135       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2136       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2137         \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
2138         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2139           \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2140             \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2141           \seq_map_inline:Nn \l_tmpa_seq {
2142             \seq_put_right:Nn \l_tmpb_seq { ##1 }
2143           }
2144         }
2145         \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2146       }{
2147         \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2148         \int_compare:nNnTF \l_tmpa_str = 0 {
2149           \exp_args:NNnx
2150           \prop_put:Nno \l_tmpb_prop { opprec }
2151           { \infprec }
2152         }{
2153           \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
2154         }
2155       }
2156     }
2157   }
2158
2159   \seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
2160   \int_step_inline:nn { \l_tmpa_str } {
2161     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpb_str {
2162       \exp_args:NNx
2163       \seq_put_right:Nn \l_tmpb_seq {
2164         \prop_item:Nn \l_tmpb_prop { opprec }
2165       }
2166     }
2167   }
2168
2169   \prop_put:Nno \l_tmpb_prop { argprec } \l_tmpb_seq
2170   \tl_clear:N \l_tmpa_tl
2171
2172   \int_compare:nNnTF \l_tmpa_str = 0 {
2173     \exp_args:NNe
2174     \cs_set:Npn \l__stex_notation_macrocode_cs {
2175       \stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2176       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }

```

```

2177         { \prop_item:Nn \l_tmpb_prop { opprec } }
2178         { \exp_not:n { #2 } }
2179     }
2180     \__stex_notation_final:
2181 }{
2182     \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
2183     \str_if_in:NnTF \l_tmpb_str b {
2184         \exp_args:Nne \use:nn
2185         {
2186             \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2187             \cs_set:Npn \l_tmpa_str } { {
2188                 \stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2189                 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2190                 { \prop_item:Nn \l_tmpb_prop { opprec } }
2191                 { \exp_not:n { #2 } }
2192             } }
2193         }{
2194             \str_if_in:NnTF \l_tmpb_str B {
2195                 \exp_args:Nne \use:nn
2196                 {
2197                     \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2198                     \cs_set:Npn \l_tmpa_str } { {
2199                         \stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2200                         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2201                         { \prop_item:Nn \l_tmpb_prop { opprec } }
2202                         { \exp_not:n { #2 } }
2203                     } }
2204                 }{
2205                     \exp_args:Nne \use:nn
2206                     {
2207                         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2208                         \cs_set:Npn \l_tmpa_str } { {
2209                             \stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2210                             { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2211                             { \prop_item:Nn \l_tmpb_prop { opprec } }
2212                             { \exp_not:n { #2 } }
2213                         } }
2214                     }
2215                 }
2216
2217         \int_zero:N \l_tmpa_int
2218         \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2219         \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2220         \__stex_notation_arguments:
2221     }
2222 }

```

(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

```

2223 \cs_new_protected:Nn \__stex_notation_arguments: {
2224     \int_incr:N \l_tmpa_int
2225     \str_if_empty:NnTF \l_tmpa_str {
2226         \__stex_notation_final:

```

```

2227 }{
2228   \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2229   \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2230   \str_if_eq:VnTF \l_tmpb_str a {
2231     \__stex_notation_argument_assoc:n
2232   }{
2233     \str_if_eq:VnTF \l_tmpb_str B {
2234       \__stex_notation_argument_assoc:n
2235     }{
2236       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2237       \tl_put_right:Nx \l_tmpa_tl {
2238         { \stex_term_math_arg:nnn
2239           { \int_use:N \l_tmpa_int }
2240           { \l_tmpb_str }
2241           { ####\int_use:N \l_tmpa_int }
2242         }
2243       }
2244       \__stex_notation_arguments:
2245     }
2246   }
2247 }
2248 }

```

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

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

```

2249 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
2250   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2251   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
2252   \tl_put_right:Nx \l_tmpa_tl {
2253     { \stex_term_math_assoc_arg:nnnn
2254       { \int_use:N \l_tmpa_int }
2255       { \l_tmpb_str }
2256       \exp_args:No \exp_not:n
2257       {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
2258       { ####\int_use:N \l_tmpa_int }
2259     }
2260   }
2261   \__stex_notation_arguments:
2262 }

```

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

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

```

2263 \cs_new_protected:Nn \__stex_notation_final: {
2264   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
2265   \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
2266   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2267   \exp_args:Nne \use:nn
2268   {
2269     \cs_generate_from_arg_count:cNnn {
2270       stex_notation_ \l_tmpa_str \c_hash_str
2271       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2272       _cs

```

```

2273     }
2274     \cs_set:Npn \l_tmpb_str } { {
2275         \exp_after:wN \exp_after:wN \exp_after:wN
2276         \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2277         { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2278     } }
2279
2280 \tl_if_empty:NF \l__stex_notation_op_tl {
2281     \cs_set:cpx {
2282         stex_op_notation_ \l_tmpa_str \c_hash_str
2283         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2284         _cs
2285     } {
2286         \stex_term_oms:nnn {
2287             \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2288             \l__stex_notation_lang_str
2289         }{
2290             \l_tmpa_str
2291         }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2292     }
2293 }
2294
2295 \exp_args:Ne
2296 \stex_add_to_current_module:n {
2297     \cs_generate_from_arg_count:cNnn {
2298         stex_notation_ \l_tmpa_str \c_hash_str
2299         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2300         _cs
2301     } \cs_set:Npn {\l_tmpb_str} {
2302         \exp_after:wN \exp_after:wN \exp_after:wN
2303         \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2304         { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2305     }
2306 \tl_if_empty:NF \l__stex_notation_op_tl {
2307     \cs_set:cpn {
2308         stex_op_notation_ \l_tmpa_str \c_hash_str
2309         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2310         _cs
2311     } {
2312         \stex_term_oms:nnn {
2313             \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2314             \l__stex_notation_lang_str
2315         }{
2316             \l_tmpa_str
2317         }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2318     }
2319 }
2320 }
2321
2322 \seq_put_right:cx {
2323     l_stex_symdecl_
2324     \prop_item:Nn \l_tmpb_prop { symbol }
2325     _notations
2326 } {

```

```

2327 \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2328 }
2329
2330 \stex_debug:nn{symbols}{
2331   Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2332   ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
2333   Operator~precedence:~
2334   \prop_item:Nn \l_tmpb_prop { opprec }^^J
2335   Argument~precedences:~
2336   \seq_use:Nn \l_tmpa_seq {,~}^^J
2337   Notation: \cs_meaning:c {
2338     stex_notation_ \l_tmpa_str \c_hash_str
2339     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2340     _cs
2341   }
2342 }
2343
2344 \prop_set_eq:cN {
2345   l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2346   \c_hash_str \l__stex_notation_lang_str _prop
2347 } \l_tmpb_prop
2348
2349 \exp_args:Ne
2350 \stex_add_to_current_module:n {
2351   \seq_put_right:cn {
2352     l_stex_symdecl_
2353     \prop_item:Nn \l_tmpb_prop { symbol }
2354     _notations
2355   } {
2356     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2357   }
2358   \prop_set_from_keyval:cn {
2359     l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2360     \c_hash_str \l__stex_notation_lang_str _prop
2361   } {
2362     symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2363     language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2364     variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,
2365     opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2366     argprecs    = \prop_item:Nn \l_tmpb_prop { argprecs }    ,
2367   }
2368 }
2369
2370 \stex_if_smsmode:TF {
2371   \stex_smsmode_set_codes:
2372   % \exp_args:Nx \stex_add_to_sms:n {
2373   %   \prop_set_from_keyval:cn {
2374   %     l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2375   %     \c_hash_str \l__stex_notation_lang_str _prop
2376   %   } {
2377   %     symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2378   %     language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2379   %     variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,
2380   %     opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,

```

```

2381 %      argprec = \prop_item:Nn \l_tmpb_prop { argprec } ,
2382 %    }
2383 %  }
2384 }{
2385
2386 % HTML annotations
2387 \stex_if_do_html:T {
2388   \stex_annotate_invisible:nnn { notation }
2389   { \prop_item:Nn \l_tmpb_prop { symbol } } {
2390     \stex_annotate_invisible:nnn { notationfragment }
2391     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
2392     \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2393     \stex_annotate_invisible:nnn { precedence }
2394     { \prop_item:Nn \l_tmpb_prop { opprec };
2395       \seq_use:Nn \l_tmpa_seq { x }
2396     }{}
2397
2398     \int_zero:N \l_tmpa_int
2399     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2400     \tl_clear:N \l_tmpa_tl
2401     \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
2402       \int_incr:N \l_tmpa_int
2403       \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2404       \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2405       \str_if_eq:VnTF \l_tmpb_str a {
2406         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2407           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2408           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2409         } }
2410       }{
2411         \str_if_eq:VnTF \l_tmpb_str B {
2412           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2413             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2414             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2415           } }
2416         }{
2417           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2418             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2419           } }
2420         }
2421       }
2422     }
2423     \stex_annotate_invisible:nnn { notationcomp }{}{
2424       \str_set:Nx \l_stex_current_symbol_str {\prop_item:Nn \l_tmpb_prop { symbol }}
2425       $ \exp_args:Nno \use:nn { \use:c {
2426         stex_notation_ \l_stex_current_symbol_str
2427         \c_hash_str \l__stex_notation_variant_str
2428         \c_hash_str \l__stex_notation_lang_str _cs
2429       } } { \l_tmpa_tl } $
2430     }
2431   }
2432 }
2433 }
2434 }

```

(End definition for `\_stex_notation_final:`)

**`\symdef`**

```

2435 \keys_define:nn { stex / symdef } {
2436   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2437   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2438   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2439   type      .tl_set:N = \l_stex_symdecl_type_tl ,
2440   def       .tl_set:N = \l_stex_symdecl_definiens_tl ,
2441   op        .tl_set:N = \l__stex_notation_op_tl ,
2442   lang      .str_set_x:N = \l__stex_notation_lang_str ,
2443   variant   .str_set_x:N = \l__stex_notation_variant_str ,
2444   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2445   unknown   .code:n = \str_set:Nx
2446             \l__stex_notation_variant_str \l_keys_key_str
2447 }
2448
2449 \cs_new_protected:Nn \_stex_notation_symdef_args:n {
2450   \str_clear:N \l_stex_symdecl_name_str
2451   \str_clear:N \l_stex_symdecl_args_str
2452   \bool_set_false:N \l_stex_symdecl_local_bool
2453   \tl_clear:N \l_stex_symdecl_type_tl
2454   \tl_clear:N \l_stex_symdecl_definiens_tl
2455   \str_clear:N \l__stex_notation_lang_str
2456   \str_clear:N \l__stex_notation_variant_str
2457   \str_clear:N \l__stex_notation_prec_str
2458   \tl_clear:N \l__stex_notation_op_tl
2459
2460   \keys_set:nn { stex / symdef } { #1 }
2461 }
2462
2463 \NewDocumentCommand \symdef { 0{} m } {
2464   \_stex_notation_symdef_args:n { #1 }
2465   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2466   \stex_symdecl_do:n { #2 }
2467   \exp_args:Nx \stex_notation_do:nn {
2468     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2469   }
2470 }
2471 \stex_deactivate_macro:Nn \symdef {module~environments}
2472
2473 (End definition for \symdef. This function is documented on page 37.)
2474 \endpackage

```

## Chapter 31

# STEX -Terms Implementation

```
2473 <*package>
2474
2475 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2476
2477 <@@=stex_terms>
2478
2479 Warnings and error messages
2480 \msg_new:nnn{stex}{error/nonotation}{
2481   Symbol~#1~invoked,~but~has~no~notation~#2!
2482 }
2483 \msg_new:nnn{stex}{error/notationarg}{
2484   Error~in~parsing~notation~#1
2485 }
2486 \msg_new:nnn{stex}{error/noop}{
2487   Symbol~#1~has~no~operator~notation~for~notation~#2
2488 }
```

### 31.1 Symbol Invocations

Arguments:

```
2488 \keys_define:nn { stex / terms } {
2489   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2490   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2491   unknown .code:n = \str_set:Nx
2492     \l__stex_terms_variant_str \l_keys_key_str
2493 }
2494
2495 \cs_new_protected:Nn \__stex_terms_args:n {
2496   \str_clear:N \l__stex_terms_lang_str
2497   \str_clear:N \l__stex_terms_variant_str
2498   \str_clear:N \l__stex_terms_prec_str
2499   \tl_clear:N \l__stex_terms_op_tl
2500
2501   \keys_set:nn { stex / terms } { #1 }
```



2502 }

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

```
2503 \cs_new_protected:Nn \stex_invoke_symbol:n {
2504   \if_mode_math:
2505     \exp_after:wN \__stex_terms_invoke_math:n
2506   \else:
2507     \exp_after:wN \__stex_terms_invoke_text:n
2508   \fi: { #1 }
2509 }
```

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

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

```
2510 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2511   \peek_charcode_remove:NTF ! {
2512     \peek_charcode:NTF [ {
2513       \__stex_terms_invoke_op:nw { #1 }
2514     }{
2515       \peek_charcode_remove:NTF ! {
2516         \peek_charcode:NTF [ {
2517           \__stex_terms_invoke_op_custom:nw
2518         }{
2519           % TODO throw error
2520         }
2521       }{
2522         \__stex_terms_invoke_op:nw { #1 } []
2523       }
2524     }{
2525       \peek_charcode_remove:NTF * {
2526         \__stex_terms_invoke_text:n { #1 }
2527       }{
2528         \peek_charcode:NTF [ {
2529           \__stex_terms_invoke_math:nw { #1 }
2530         }{
2531           \__stex_terms_invoke_math:nw { #1 } []
2532         }
2533       }
2534     }
2535   }
2536 }
```

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

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

```
2537 \cs_new_protected:Npn \__stex_terms_invoke_op_custom:nw #1 [#2] {
2538   \stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
2539     \stex_highlight_term:nn{#1}{#2}
2540   }
2541 }
```

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

\\_stex\_terms\_invoke\_op:nw

```

2542 \cs_new_protected:Npn \_stex_terms_invoke_op:nw #1 [#2] {
2543   \_stex_terms_args:n { #2 }
2544   \cs_if_exist:cTF {
2545     stex_op_notation_ #1 \c_hash_str
2546     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2547   }{
2548     \csname stex_op_notation_ #1 \c_hash_str
2549       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2550     \endcsname
2551   }{
2552     \msg_error:nnxx{stex}{error/noop}{#1}{\l__stex_terms_variant_str \c_hash_str \l__stex_te
2553   }
2554 }

```

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

\\_stex\_terms\_invoke\_math:nw

```

2555 \cs_new_protected:Npn \_stex_terms_invoke_math:nw #1 [#2] {
2556   \_stex_terms_args:n { #2 }
2557   \seq_if_empty:cTF {
2558     l_stex_symdecl_ #1 _notations
2559   } {
2560     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
2561   } {
2562     \seq_if_in:cxTF {
2563       l_stex_symdecl_ #1 _notations
2564     }
2565     { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
2566       \str_set:Nn \l_stex_current_symbol_str { #1 }
2567       \use:c{
2568         stex_notation_ #1 \c_hash_str
2569         \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2570         _cs
2571       }
2572     }{
2573       \str_if_empty:NTF \l__stex_terms_variant_str {
2574         \str_if_empty:NTF \l__stex_terms_lang_str {
2575           \seq_get_left:cN {
2576             l_stex_symdecl_ #1 _notations
2577           } \l_tmpa_str
2578           \str_set:Nn \l_stex_current_symbol_str { #1 }
2579           \use:c{
2580             stex_notation_ #1 \c_hash_str \l_tmpa_str
2581             _cs
2582           }
2583         }{
2584           \msg_error:nnxx{stex}{error/nonotation}{#1}{
2585             ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2586           }
2587         }
2588       }{
2589         \msg_error:nnxx{stex}{error/nonotation}{#1}{
2590           ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str

```

```

2591     }
2592   }
2593 }
2594 }
2595 }

```

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

`\_stex_terms_invoke_text:n`

```

2596 \cs_new_protected:Nn \_stex_terms_invoke_text:n {
2597   \peek_charcode_remove:NTF ! {
2598     \stex_term_custom:nn { #1 } { }
2599   }{
2600     \prop_set_eq:Nc \l_tmpa_prop {
2601       l_stex_symdecl_ #1 _prop
2602     }
2603     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2604     \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2605   }
2606 }

```

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

## 31.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
2607 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2608 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2609 \int_new:N \l__stex_terms_downprec
2610 \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
2611 \tl_set:Nn \l__stex_terms_left_bracket_str (
2612 \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

```

2613 \cs_new_protected:Nn \_stex_terms_maybe_brackets:nn {
2614   \bool_if:NTF \l__stex_terms_brackets_done_bool {
2615     \bool_set_false:N \l__stex_terms_brackets_done_bool
2616     #2
2617   } {
2618     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
2619       \bool_if:NTF \l_stex_inarray_bool { #2 }{
2620         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
2621         \dobrackets { #2 }
2622       }

```

```

2623     }{ #2 }
2624   }
2625 }

```

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

### `\dobrackets`

```

2626 \bool_new:N \l__stex_terms_brackets_done_bool
2627 %\RequirePackage{scalerel}
2628 \cs_new_protected:Npn \dobrackets #1 {
2629   %\ThisStyle{\if D\m@switch
2630   %   \exp_args:Nnx \use:nn
2631   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
2632   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
2633   % \else
2634   \exp_args:Nnx \use:nn
2635   {
2636     \bool_set_true:N \l__stex_terms_brackets_done_bool
2637     \int_set:Nn \l__stex_terms_downprec \infprec
2638     \l__stex_terms_left_bracket_str
2639     #1
2640   }
2641   {
2642     \bool_set_false:N \l__stex_terms_brackets_done_bool
2643     \l__stex_terms_right_bracket_str
2644     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2645   }
2646   %\fi}
2647 }

```

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

### `\withbrackets`

```

2648 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2649   \exp_args:Nnx \use:nn
2650   {
2651     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
2652     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
2653     #3
2654   }
2655   {
2656     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
2657     {\l__stex_terms_left_bracket_str}
2658     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
2659     {\l__stex_terms_right_bracket_str}
2660   }
2661 }

```

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

### `\STEXinvisible`

```

2662 \cs_new_protected:Npn \STEXinvisible #1 {
2663   \stex_annotate_invisible:n { #1 }
2664 }

```

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

OMDoc terms:

`\_stex_term_math_oms:nnnn`

```

2665 \cs_new_protected:Nn \_stex_term_oms:nnn {
2666   \stex_annotate:nnn{ OMID }{ #2 }{
2667     \stex_highlight_term:nn { #1 } { #3 }
2668   }
2669 }
2670
2671 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2672   \__stex_terms_maybe_brackets:nn { #3 }{
2673     \stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2674   }
2675 }

```

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

`\_stex_term_math_oma:nnnn`

```

2676 \cs_new_protected:Nn \_stex_term_oma:nnn {
2677   \stex_annotate:nnn{ OMA }{ #2 }{
2678     \stex_highlight_term:nn { #1 } { #3 }
2679   }
2680 }
2681
2682 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2683   \__stex_terms_maybe_brackets:nn { #3 }{
2684     \stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2685   }
2686 }

```

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

`\_stex_term_math_omb:nnnn`

```

2687 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2688   \stex_annotate:nnn{ OMBIND }{ #2 }{
2689     \stex_highlight_term:nn { #1 } { #3 }
2690   }
2691 }
2692
2693 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2694   \__stex_terms_maybe_brackets:nn { #3 }{
2695     \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2696   }
2697 }

```

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

`\_stex_term_math_arg:nnn`

```

2698 \cs_new_protected:Nn \_stex_term_arg:nn {
2699   \stex_unhighlight_term:n {
2700     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2701   }
2702 }

```

```

2703 \cs_new_protected:Nn \stex_term_math_arg:nnn {
2704   \exp_args:Nnx \use:nn
2705     { \int_set:Nn \l__stex_terms_downprec { #2 }
2706       \stex_term_arg:nn { #1 }{ #3 }
2707     }
2708     { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2709   }

```

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

`\stex_term_math_assoc_arg:nnnn`

```

2710 \cs_new_protected:Nn \stex_term_math_assoc_arg:nnnn {
2711   \clist_set:Nn \l_tmpa_clist{ #4 }
2712   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
2713     \tl_set:Nn \l_tmpa_tl { #4 }
2714   }{
2715     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
2716     \clist_reverse:N \l_tmpa_clist
2717     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
2718
2719     \clist_map_inline:Nn \l_tmpa_clist {
2720       \exp_args:NNNo \exp_args:Nno \tl_set:No \l_tmpa_tl {
2721         \exp_args:Nno
2722           \l_tmpa_cs { ##1 } \l_tmpa_tl
2723       }
2724     }
2725
2726   }
2727   \exp_args:Nnno
2728   \stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2729 }

```

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

`\stex_term_custom:nn`

```

2730 \cs_new_protected:Nn \stex_term_custom:nn {
2731   \str_set:Nn \l__stex_terms_custom_uri { #1 }
2732   \str_set:Nn \l_tmpa_str { #2 }
2733   \tl_clear:N \l_tmpa_tl
2734   \int_zero:N \l_tmpa_int
2735   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2736   \__stex_terms_custom_loop:
2737 }

```

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

`\__stex_terms_custom_loop:`

```

2738 \cs_new_protected:Nn \__stex_terms_custom_loop: {
2739   \bool_set_false:N \l_tmpa_bool
2740   \bool_while_do:nn {
2741     \str_if_eq_p:ee X {
2742       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2743     }
2744   }{
2745     \int_incr:N \l_tmpa_int

```

```

2746 }
2747
2748 \peek_charcode:NTF [ {
2749   % notation/text component
2750   \__stex_terms_custom_component:w
2751 } {
2752   \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2753     % all arguments read => finish
2754     \__stex_terms_custom_final:
2755   } {
2756     % arguments missing
2757     \peek_charcode_remove:NTF * {
2758       % invisible, specific argument position or both
2759       \peek_charcode:NTF [ {
2760         % visible specific argument position
2761         \__stex_terms_custom_arg:wn
2762       } {
2763         % invisible
2764         \peek_charcode_remove:NTF * {
2765           % invisible specific argument position
2766           \__stex_terms_custom_arg_inv:wn
2767         } {
2768           % invisible next argument
2769           \__stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2770         }
2771       }
2772     } {
2773       % next normal argument
2774       \__stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
2775     }
2776   }
2777 }
2778 }

```

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

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

```

2779 \cs_new_protected:Npn \__stex_terms_custom_arg_inv:wn [ #1 ] #2 {
2780   \bool_set_true:N \l_tmpa_bool
2781   \__stex_terms_custom_arg:wn [ #1 ] { #2 }
2782 }

```

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

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

```

2783 \cs_new_protected:Npn \__stex_terms_custom_arg:wn [ #1 ] #2 {
2784   \str_set:Nx \l_tmpb_str {
2785     \str_item:Nn \l_tmpa_str { #1 }
2786   }
2787   \str_case:VnTF \l_tmpb_str {
2788     { X } {
2789       \msg_error:nnx{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2790     }
2791     { i } { \__stex_terms_custom_set_X:n { #1 } }
2792     { b } { \__stex_terms_custom_set_X:n { #1 } }

```

```

2793 { a } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
2794 { B } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
2795 }{}{
2796 \msg_error:nnx{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2797 }
2798
2799 \bool_if:nTF \l_tmpa_bool {
2800 \tl_put_right:Nx \l_tmpa_tl {
2801 \stex_annotate_invisible:n {
2802 \stex_term_arg:nn { \int_eval:n { #1 } }
2803 \exp_not:n { { #2 } }
2804 }
2805 }
2806 } {
2807 \tl_put_right:Nx \l_tmpa_tl {
2808 \stex_term_arg:nn { \int_eval:n { #1 } }
2809 \exp_not:n { { #2 } }
2810 }
2811 }
2812
2813 \__stex_terms_custom_loop:
2814 }

```

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

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

```

2815 \cs_new_protected:Nn \__stex_terms_custom_set_X:n {
2816 \str_set:Nx \l_tmpa_str {
2817 \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
2818 X
2819 \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
2820 }
2821 }

```

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

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

```

2822 \cs_new_protected:Npn \__stex_terms_custom_component:w [ #1 ] {
2823 \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
2824 \__stex_terms_custom_loop:
2825 }

```

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

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

```

2826 \cs_new_protected:Nn \__stex_terms_custom_final: {
2827 \int_compare:nNnTF \l_tmpb_int = 0 {
2828 \exp_args:Nnno \stex_term_oms:nnn
2829 }{
2830 \str_if_in:NnTF \l_tmpa_str {b} {
2831 \exp_args:Nnno \stex_term_ombind:nnn
2832 } {
2833 \exp_args:Nnno \stex_term_oma:nnn
2834 }
2835 }

```



```

2836 { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
2837 }

```

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

**\symref**

**\symname**

```

2838 \NewDocumentCommand \symref { m m }{
2839   \let\compemph_uri_prev:\compemph@uri
2840   \let\compemph@uri\symrefemph@uri
2841   \STEXsymbol{#1}! [#2]
2842   \let\compemph@uri\compemph_uri_prev:
2843 }
2844
2845 \keys_define:nn { stex / symname } {
2846   post      .str_set_x:N    = \l_stex_symname_post_str
2847 }
2848
2849 \cs_new_protected:Nn \stex_symname_args:n {
2850   \str_clear:N \l_stex_symname_post_str
2851   \keys_set:nn { stex / symname } { #1 }
2852 }
2853
2854 \NewDocumentCommand \symname { O{} m }{
2855   \stex_symname_args:n { #1 }
2856   \stex_get_symbol:n { #2 }
2857   \str_set:Nx \l_tmpa_str {
2858     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
2859   }
2860   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {-}
2861
2862   \let\compemph_uri_prev:\compemph@uri
2863   \let\compemph@uri\symrefemph@uri
2864   \exp_args:NNx \use:nn
2865   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }! [
2866     \l_tmpa_str \l_stex_symname_post_str
2867   ] }
2868   \let\compemph@uri\compemph_uri_prev:
2869 }

```

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

## 31.3 Notation Components

```

2870 <@@=stex_notationcomps>

```

**\stex\_highlight\_term:nn**

```

2871
2872 \str_new:N \l_stex_current_symbol_str
2873 \cs_new_protected:Nn \stex_highlight_term:nn {
2874   \exp_args:Nnx
2875   \use:nn {
2876     \str_set:Nx \l_stex_current_symbol_str { #1 }
2877     #2
2878   } {

```

```

2879 \str_set:Nx \exp_not:N \l_stex_current_symbol_str
2880 { \l_stex_current_symbol_str }
2881 }
2882 }
2883
2884 \cs_new_protected:Nn \stex_unhighlight_term:n {
2885 % \latexml_if:TF {
2886 % #1
2887 % } {
2888 % \rustex_if:TF {
2889 % #1
2890 % } {
2891 % #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
2892 % }
2893 % }
2894 }

```

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

```

\comp
\compemph@uri
\compemph
\defemph
\defemph@uri
\symrefemph
\symrefemph@uri
2895 \cs_new_protected:Npn \comp #1 {
2896 \str_if_empty:NF \l_stex_current_symbol_str {
2897 \rustex_if:TF {
2898 \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
2899 }{
2900 \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
2901 }
2902 }
2903 }
2904
2905 \cs_new_protected:Npn \compemph@uri #1 #2 {
2906 \compemph{ #1 }
2907 }
2908
2909
2910 \cs_new_protected:Npn \compemph #1 {
2911 #1
2912 }
2913
2914 \cs_new_protected:Npn \defemph@uri #1 #2 {
2915 \defemph{#1}
2916 }
2917
2918 \cs_new_protected:Npn \defemph #1 {
2919 \textbf{#1}
2920 }
2921
2922 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
2923 \symrefemph{#1}
2924 }
2925
2926 \cs_new_protected:Npn \symrefemph #1 {
2927 \textbf{#1}
2928 }

```

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

## `\ellipses`

```
2929 \NewDocumentCommand \ellipses {} { \ldots }
```

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

```

\parray
\prmatrix 2930 \bool_new:N \l_stex_inarray_bool
\parrayline 2931 \bool_set_false:N \l_stex_inarray_bool
\parraylineh 2932 \NewDocumentCommand \parray { m m } {
\parraycell 2933 \begin{group}
2934 \bool_set_true:N \l_stex_inarray_bool
2935 \begin{array}{#1}
2936 #2
2937 \end{array}
2938 \end{group}
2939 }
2940
2941 \NewDocumentCommand \prmatrix { m } {
2942 \begin{group}
2943 \bool_set_true:N \l_stex_inarray_bool
2944 \begin{matrix}
2945 #1
2946 \end{matrix}
2947 \end{group}
2948 }
2949
2950 \def \maybepline {
2951 \bool_if:NT \l_stex_inarray_bool {\hline}
2952 }
2953
2954 \def \parrayline #1 #2 {
2955 #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
2956 }
2957
2958 \def \pmrow #1 { \parrayline{}{ #1 } }
2959
2960 \def \parraylineh #1 #2 {
2961 #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
2962 }
2963
2964 \def \parraycell #1 {
2965 #1 \bool_if:NT \l_stex_inarray_bool {&}
2966 }

```

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

```
2967 \end{package}
```

## Chapter 32

# STEX -Structural Features Implementation

```
2968 <*package>
2969
2970 %%%%%%%%%%% features.dtx %%%%%%%%%%%
2971
2972 <@@=stex_features>
2973
2974 Warnings and error messages
```

### 32.1 Imports with modification

```
2974 \cs_new_protected:Nn \stex_get_symbol_in_clonemodule:n {
2975   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2976     \__stex_features_get_symbol_from_cs:n { #1 }
2977   }{
2978     % argument is a string
2979     % is it a command name?
2980     \cs_if_exist:cTF { #1 }{
2981       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2982       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2983       \str_if_empty:NNTF \l_tmpa_str {
2984         \exp_args:Nx \cs_if_eq:NNTF {
2985           \tl_head:N \l_tmpa_tl
2986         } \stex_invoke_symbol:n {
2987           \exp_args:No \__stex_features_get_symbol_from_cs:n { \use:c { #1 } }
2988         }{
2989           \__stex_features_get_symbol_from_string:n { #1 }
2990         }
2991       } {
2992         \__stex_features_get_symbol_from_string:n { #1 }
2993       }
2994     }{
2995       % argument is not a command name
```

```

2996     \_stex_features_get_symbol_from_string:n { #1 }
2997     % \l_stex_all_symbols_seq
2998   }
2999 }
3000 }
3001
3002 \cs_new_protected:Nn \_stex_features_get_symbol_from_string:n {
3003   \str_set:Nn \l_tmpa_str { #1 }
3004   \bool_set_false:N \l_tmpa_bool
3005   \bool_if:NF \l_tmpa_bool {
3006     \tl_set:Nn \l_tmpa_tl {
3007       \msg_set:nnn{stex}{error/unknownsymbol}{
3008         No~symbol~#1~found!
3009       }
3010       \msg_error:nn{stex}{error/unknownsymbol}
3011     }
3012     \str_set:Nn \l_tmpa_str { #1 }
3013     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3014     \seq_map_inline:Nn \_stex_features_clonemodule_fields_seq {
3015       \str_set:Nn \l_tmpb_str { ##1 }
3016       \str_if_eq:eeT { \l_tmpa_str } {
3017         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3018       } {
3019         \seq_map_break:n {
3020           \tl_set:Nn \l_tmpa_tl {
3021             \str_set:Nn \l_stex_get_symbol_uri_str {
3022               ##1
3023             }
3024             \_stex_features_get_symbol_check:
3025           }
3026         }
3027       }
3028     }
3029     \l_tmpa_tl
3030   }
3031 }
3032
3033 \cs_new_protected:Nn \_stex_features_get_symbol_from_cs:n {
3034   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3035     { \tl_tail:N \l_tmpa_tl }
3036   \tl_if_single:NTF \l_tmpa_tl {
3037     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3038       \exp_after:wN \str_set:Nn \exp_after:wN
3039         \l_stex_get_symbol_uri_str \l_tmpa_tl
3040       \_stex_features_get_symbol_check:
3041     }{
3042       % TODO
3043       % tail is not a single group
3044     }
3045   }{
3046     % TODO
3047     % tail is not a single group
3048   }
3049 }

```

```

3050
3051 \cs_new_protected:Nn \__stex_features_get_symbol_check: {
3052   \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq {?} \l_stex_get_symbol_uri_str
3053   \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 3 {
3054     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
3055     \str_set:Nx \l_tmpa_str {\seq_use:Nn \l_tmpa_seq ?}
3056     \seq_if_in:NoF \l__stex_features_clonemodule_modules_seq {
3057       % TODO error
3058     }
3059   }{
3060     % TODO error
3061   }
3062 }
3063
3064 \NewDocumentEnvironment {clonemodule} { 0{} m m}{
3065   \stex_import_module_uri:nn { #1 } { #2 }
3066   \stex_deactivate_macro:Nn \symdecl {module~environments}
3067   \stex_deactivate_macro:Nn \symdef {module~environments}
3068   \stex_deactivate_macro:Nn \notation {module~environments}
3069   \stex_reactivate_macro:N \assign
3070   \stex_reactivate_macro:N \renamedec1
3071   \stex_reactivate_macro:N \donotclone
3072   \str_set:Nx \l_stex_current_clonemodule_name_str {#3}
3073   %\let\notation\notation_in_clonemodules:
3074   %\stex_module_setup:nn {}{ #3 }
3075   \stex_import_require_module:nnnn
3076     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3077     { \l_stex_import_path_str } { \l_stex_import_name_str }
3078   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3079   \seq_set_eq:NN \l__stex_features_clonemodule_modules_seq \l_stex_collect_imports_seq
3080   \seq_clear:N \l__stex_features_clonemodule_fields_seq
3081   \seq_map_inline:Nn \l__stex_features_clonemodule_modules_seq {
3082     \seq_map_inline:cn {c_stex_module_##1_constants}{
3083       \exp_args:NNx \seq_put_right:Nn \l__stex_features_clonemodule_fields_seq {
3084         ##1 ? #####1
3085       }
3086     }
3087   }
3088   \seq_clear:N \l_tmpa_seq
3089   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_clonemodule_prop {
3090     name      = \l_stex_current_clonemodule_name_str ,
3091     module    = \l_stex_current_module_str ,
3092     from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3093     includes  = \l_tmpa_seq ,
3094     fields    = \l_tmpa_seq
3095   }
3096   \stex_debug:nn{clonemodule}{cloning~module~{\l_stex_import_ns_str ?\l_stex_import_name_str
3097     as~\l_stex_current_module_str?\l_stex_current_clonemodule_name_str}
3098   \stex_debug:nn{clonemodule}{fields:\seq_use:Nn \l__stex_features_clonemodule_fields_seq {,
3099   % todo
3100
3101   \stex_if_smsmode:TF {
3102     \stex_smsmode_set_codes:
3103   } {

```

```

3104     \begin{stex_annotate_env} {structure} {
3105         \l_stex_current_module_str?\l_stex_current_clonemodule_name_str
3106     }
3107     \stex_annotate_invisible:nnn{from}{\l_stex_import_ns_str ?\l_stex_import_name_str}{ }
3108 }
3109 \bool_set_eq:NN \l__stex_features_oldhtml_bool \l_stex_html_do_output_bool
3110 \bool_set_false:N \l_stex_html_do_output_bool
3111 }{
3112     \bool_set_eq:NN \l_stex_html_do_output_bool \l__stex_features_oldhtml_bool
3113     \tl_clear:N \l_tmpa_tl
3114     \prop_get:NnN \l_stex_current_clonemodule_prop {fields} \l_tmpa_seq
3115     \seq_map_inline:Nn \l__stex_features_clonemodule_modules_seq {
3116         \seq_map_inline:cn {c_stex_module_###1_constants}{\stex_annotate:nnn{assignment} {###1?###1
3117             \str_if_exist:cTF {l__stex_features_clonemodule_###1?####1_name_str} {
3118                 \tl_put_right:Nx \l_tmpa_tl {
3119                     \prop_set_from_keyval:cn {
3120                         l_stex_symdecl\l_stex_current_module_str ? \use:c{l__stex_features_clonemodule_
3121                     }{
3122                         \exp_after:wN \prop_to_keyval:N \csname
3123                         l_stex_symdecl\l_stex_current_module_str ? \use:c{l__stex_features_clonemodule_
3124                         \endcsname
3125                     }
3126                     \seq_clear:c {
3127                         l_stex_symdecl_
3128                         \l_stex_current_module_str ? \use:c{l__stex_features_clonemodule_###1?####1_name_
3129                         _notations
3130                     }
3131                 }
3132             }
3133             \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_features_clonemodule_###1?####1_name_
3134             \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{l__stex_features_
3135             \str_if_exist:cT {l__stex_features_clonemodule_###1?####1_macroname_str} {
3136                 \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_clonemodule_###1?###
3137                 \tl_put_right:Nx \l_tmpa_tl {
3138                     \tl_set:cx {\use:c{l__stex_features_clonemodule_###1?####1_macroname_str}}{
3139                     \stex_invoke_symbol:n {
3140                         \l_stex_current_module_str ? \use:c{l__stex_features_clonemodule_###1?####1_n
3141                     }
3142                 }
3143             }
3144         }{
3145             \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ###1?####1 _prop}
3146             \prop_put:Nnx \l_tmpa_prop { name }{ \l_stex_current_clonemodule_name_str / ####1 }
3147             \prop_put:Nnx \l_tmpa_prop { module }{ \l_stex_current_module_str }
3148             \tl_put_right:Nx \l_tmpa_tl {
3149                 \prop_set_from_keyval:cn {
3150                     l_stex_symdecl\l_stex_current_module_str ? \l_stex_current_clonemodule_name_str
3151                 }{
3152                     \prop_to_keyval:N \l_tmpa_prop
3153                 }
3154                 \seq_clear:c {
3155                     l_stex_symdecl_
3156                     \l_stex_current_module_str ? \l_stex_current_clonemodule_name_str / ####1
3157                     _notations

```

```

3158     }
3159   }
3160   \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \l_stex_current_clonemod
3161   \str_if_exist:cT {l__stex_features_clonemodule_##1?####1_macroname_str} {
3162     \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_clonemodule_##1?##
3163     \tl_put_right:Nx \l_tmpa_tl {
3164       \tl_set:cx {\use:c{l__stex_features_clonemodule_##1?####1_macroname_str}}{
3165       \stex_invoke_symbol:n {
3166         \l_stex_current_module_str ? \l_stex_current_clonemodule_name_str / ####1
3167     }
3168   }
3169 }
3170 }
3171 }
3172 \tl_if_exist:cT {l__stex_features_clonemodule_##1?####1_def_tl}{
3173   \stex_annotate_invisible:nnn{definiens}{\use:c{l__stex_features_clonemodule_##1?##
3174 }
3175 % todo notations
3176 }}
3177 }
3178 \prop_put:Nno \l_stex_current_clonemodule_prop {fields} \l_tmpa_seq
3179 \tl_put_left:Nx \l_tmpa_tl {
3180   \prop_set_from_keyval:cn {
3181     l_stex_clonemodule_ \l_stex_current_module_str?\l_stex_current_clonemodule_name_str _p
3182   }{
3183     \prop_to_keyval:N \l_stex_current_clonemodule_prop
3184   }
3185 }
3186 \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl
3187 \stex_debug:nn{clonemodule}{result:\meaning \l_tmpa_tl}
3188 \exp_args:Nx \stex_do_aftergroup:n { \stex_if_smsmode:TF {
3189   \exp_args:No \exp_not:n \l_tmpa_tl
3190   }{ \stex_do_aftergroup:n {
3191     \exp_args:No \exp_not:n \l_tmpa_tl
3192   } }
3193 }
3194 \stex_if_smsmode:F {
3195   \end{stex_annotate_env}
3196 }
3197 }
3198
3199 \NewDocumentCommand \donotclone { 0{} m}{
3200   \stex_import_module_uri:nn { #1 } { #2 }
3201   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3202   \seq_map_inline:Nn \l_stex_collect_imports_seq {
3203     \seq_remove_all:Nn \l__stex_features_clonemodule_modules_seq { ##1 }
3204     \seq_map_inline:cn {c_stex_module_##1_constants}{
3205       \seq_remove_all:Nn \l__stex_features_clonemodule_fields_seq { ##1 ? ####1 }
3206       \bool_lazy_any_p:nT {
3207         { \cs_if_exist_p:c {l__stex_features_clonemodule_##1?####1_name_str}}
3208         { \cs_if_exist_p:c {l__stex_features_clonemodule_##1?####1_macroname_str}}
3209         { \cs_if_exist_p:c {l__stex_features_clonemodule_##1?####1_def_tl}}
3210       }{
3211         % TODO throw error

```



```

3212     }
3213   }
3214 }
3215
3216 \prop_get:NnN \l_stex_current_clonemodule_prop { includes } \l_tmpa_seq
3217 \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_ns_str ?\l_stex_import_name_str }
3218 \prop_put:Nnx \l_stex_current_clonemodule_prop {includes} \l_tmpa_seq
3219 }
3220
3221 \NewDocumentCommand \assign { m m }{
3222   \stex_get_symbol_in_clonemodule:n {#1}
3223   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
3224   \tl_set:cn {l__stex_features_clonemodule_###1?###1_def_tl}{#2}
3225 }
3226
3227 \keys_define:nn { stex / renamedec1 } {
3228   name          .str_set_x:N = \l_stex_renamedec1_name_str
3229 }
3230 \cs_new_protected:Nn \__stex_features_renamedec1_args:n {
3231   \str_clear:N \l_stex_renamedec1_name_str
3232 }
3233 \keys_set:nn { stex / renamedec1 } { #1 }
3234 }
3235
3236 \NewDocumentCommand \renamedec1 { O{} m m }{
3237   \__stex_features_renamedec1_args:n { #1 }
3238   \stex_get_symbol_in_clonemodule:n {#2}
3239   \stex_debug:nn{renamedec1}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
3240   \str_set:cx {l__stex_features_clonemodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
3241   \str_if_empty:NTF \l_stex_renamedec1_name_str {
3242     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3243       \l_stex_get_symbol_uri_str
3244     } }
3245   } {
3246     \str_set:cx {l__stex_features_clonemodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex_
3247     \stex_debug:nn{renamedec1}{@~\l_stex_current_module_str ? \l_stex_renamedec1_name_str}
3248     \prop_set_eq:cc {l_stex_symdecl_
3249       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3250     _prop
3251     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
3252     \seq_set_eq:cc {l_stex_symdecl_
3253       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3254     _notations
3255     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
3256     \prop_put:cnx {l_stex_symdecl_
3257       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3258     _prop
3259     }{ name }{ \l_stex_renamedec1_name_str }
3260     \prop_put:cnx {l_stex_symdecl_
3261       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3262     _prop
3263     }{ module }{ \l_stex_current_module_str }
3264     \exp_args:NNx \seq_put_left:Nn \__stex_features_clonemodule_fields_seq {
3265       \l_stex_current_module_str ? \l_stex_renamedec1_name_str

```

```

3266     }
3267     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3268       \l_stex_current_module_str ? \l_stex_renameddecl_name_str
3269     } }
3270   }
3271 }
3272 %\NewDocumentCommand \notation_in_clonemodules: { 0{ } m } {
3273 % \stex_notation_args:n { #1 }
3274 % \tl_clear:N \l_stex_symdecl_definiens_tl
3275 % \stex_get_symbol_in_clonemodule:n { #2 }
3276 % \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
3277 % % todo
3278 %}
3279 \stex_deactivate_macro:Nn \assign {clonemodules}
3280 \stex_deactivate_macro:Nn \renameddecl {clonemodules}
3281 \stex_deactivate_macro:Nn \donotclone {clonemodules}
3282
3283
3284 \seq_new:N \l_stex_implicit_morphisms_seq
3285 \NewDocumentCommand \implicitmorphism { 0{ } m m }{
3286   \stex_import_module_uri:nn { #1 } { #2 }
3287   \stex_debug:nn{implicits}{
3288     Implicit~morphism:~
3289     \l_stex_module_ns_str ? \l__stex_features_name_str
3290   }
3291   \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
3292     \l_stex_module_ns_str ? \l__stex_features_name_str
3293   }{
3294     \msg_error:nnn{stex}{error/conflictingmodules}{
3295       \l_stex_module_ns_str ? \l__stex_features_name_str
3296     }
3297   }
3298
3299   % TODO
3300
3301
3302
3303   \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
3304     \l_stex_module_ns_str ? \l__stex_features_name_str
3305   }
3306 }
3307

```

## 32.2 The feature environment

structural@feature

```

3308
3309 \NewDocumentEnvironment{structural@feature}{ m m m }{
3310   \stex_if_in_module:F {
3311     \msg_set:nnn{stex}{error/nomodule}{
3312       Structural~Feature~has~to~occur~in~a~module:\\
3313       Feature~#2~of~type~#1\\
3314       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq

```

```

3315     }
3316     \msg_error:nn{stex}{error/nomodule}
3317 }
3318
3319 \str_set:Nx \l_stex_module_name_str {
3320     \prop_item:Nn \l_stex_current_module_prop
3321     { name } / #2 - feature
3322 }
3323
3324 \str_set:Nx \l_stex_module_ns_str {
3325     \prop_item:Nn \l_stex_current_module_prop
3326     { ns }
3327 }
3328
3329
3330 \str_clear:N \l_tmpa_str
3331 \seq_clear:N \l_tmpa_seq
3332 \tl_clear:N \l_tmpa_tl
3333 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
3334     origname = #2,
3335     name      = \l_stex_module_name_str ,
3336     ns        = \l_stex_module_ns_str ,
3337     imports   = \exp_not:o { \l_tmpa_seq } ,
3338     constants = \exp_not:o { \l_tmpa_seq } ,
3339     content   = \exp_not:o { \l_tmpa_tl } ,
3340     file      = \exp_not:o { \g_stex_currentfile_seq } ,
3341     lang      = \l_stex_module_lang_str ,
3342     sig       = \l_tmpa_str ,
3343     meta      = \l_tmpa_str ,
3344     feature   = #1 ,
3345 }
3346
3347 \stex_if_smsmode:TF {
3348     \stex_smsmode_set_codes:
3349 } {
3350     \begin{stex_annotate_env}{ feature:#1 }{}
3351     \stex_annotate_invisible:nnn{header}{}{ #3 }
3352 }
3353 }{
3354     \str_set:Nx \l_tmpa_str {
3355         c_stex_feature_
3356         \prop_item:Nn \l_stex_current_module_prop { ns } ?
3357         \prop_item:Nn \l_stex_current_module_prop { name }
3358         _prop
3359     }
3360     \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
3361     \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
3362     \stex_if_smsmode:TF {
3363         \exp_args:Nx \stex_add_to_sms:n {
3364             \prop_gset_from_keyval:cn {
3365                 c_stex_feature_
3366                 \prop_item:Nn \l_stex_current_module_prop { ns } ?
3367                 \prop_item:Nn \l_stex_current_module_prop { name }
3368                 _prop

```

```

3369     } {
3370         origname = #2,
3371         name      = \prop_item:cn { \l_tmpa_str } { name } ,
3372         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
3373         imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
3374         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
3375         content   = \prop_item:cn { \l_tmpa_str } { content } ,
3376         file      = \prop_item:cn { \l_tmpa_str } { file } ,
3377         lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
3378         sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
3379         meta      = \prop_item:cn { \l_tmpa_str } { meta } ,
3380         feature   = \prop_item:cn { \l_tmpa_str } { feature }
3381     }
3382 }
3383 } {
3384     \end{stex_annotate_env}
3385 }
3386 }
3387

```

## 32.3 Features

structure

```

3388
3389 \prop_new:N \l_stex_all_structures_prop
3390
3391 \keys_define:nn { stex / features / structure } {
3392     name .str_set_x:N = \l__stex_features_structure_name_str ,
3393 }
3394
3395 \cs_new_protected:Nn \__stex_features_structure_args:n {
3396     \str_clear:N \l__stex_features_structure_name_str
3397     \keys_set:nn { stex / features / structure } { #1 }
3398 }
3399
3400 %\stex_new_feature:nnnn { structure } { 0{ } m } {
3401 % \__stex_features_structure_args:n { ##1 }
3402 % \str_if_empty:NT \l__stex_features_structure_name_str {
3403 %     \str_set:Nx \l__stex_features_structure_name_str { ##2 }
3404 % }
3405 %} {
3406 %
3407 %}
3408
3409 \NewDocumentEnvironment{mathstructure}{ 0{ } m }{
3410     \__stex_features_structure_args:n { #1 }
3411     \str_if_empty:NT \l__stex_features_structure_name_str {
3412         \str_set:Nx \l__stex_features_structure_name_str { #2 }
3413     }
3414     \exp_args:Nnnx
3415     \begin{structural@feature}{ structure }
3416         { \l__stex_features_structure_name_str }{}
3417     \seq_clear:N \l_tmpa_seq

```

```

3418 \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
3419
3420 }{
3421 \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
3422 \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
3423 \str_set:Nx \l_tmpa_str {
3424 \prop_item:Nn \l_stex_current_module_prop { ns } ?
3425 \prop_item:Nn \l_stex_current_module_prop { name }
3426 }
3427 \seq_map_inline:Nn \l_tmpa_seq {
3428 \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
3429 }
3430 \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
3431 \exp_args:Nnx
3432 \AddToHookNext { env / mathstructure / after }{
3433 \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}}{
3434 \_stex_term_math_oms:nnnn { \l_tmpa_str }{}{0}{}
3435 }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
3436 \STEXexport {
3437 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3438 {\prop_item:Nn \l_stex_current_module_prop { origname }}
3439 {\l_tmpa_str}
3440 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3441 {#2}{\l_tmpa_str}
3442 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3443 % \prop_item:Nn \l_stex_current_module_prop { origname },
3444 % \l_tmpa_str
3445 % }
3446 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3447 % #2,\l_tmpa_str
3448 % }
3449 % \tl_set:cx { #2 } {
3450 % \stex_invoke_structure:n { \l_tmpa_str }
3451 % }
3452 % }
3453
3454 \end{structural@feature}
3455 % \g_stex_last_feature_prop
3456 }

```

\instantiate

```

3457 \seq_new:N \l__stex_features_structure_field_seq
3458 \str_new:N \l__stex_features_structure_field_str
3459 \str_new:N \l__stex_features_structure_def_tl
3460 \prop_new:N \l__stex_features_structure_prop
3461 \NewDocumentCommand \instantiate { m O{} m }{
3462 \stex_smsmode_set_codes:
3463 \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
3464 \prop_set_eq:Nc \l__stex_features_structure_prop {
3465 c_stex_feature_\l_tmpa_str _prop
3466 }
3467 \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
3468 \seq_map_inline:Nn \l__stex_features_structure_field_seq {
3469 \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }

```

```

3470 \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3471   \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
3472   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
3473     {} \l_tmpa_tl
3474   \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3475     \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
3476     \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3477     \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3478   }{
3479     \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
3480     \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3481     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
3482       \l_tmpa_tl
3483     \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3484       \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
3485       \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3486     }{
3487       \tl_clear:N \l_tmpb_tl
3488     }
3489   }
3490 }{
3491   \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
3492   \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3493     \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
3494     \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
3495     \tl_clear:N \l_tmpa_tl
3496   }{
3497     % TODO throw error
3498   }
3499 }
3500 % \l_tmpa_str: name
3501 % \l_tmpa_tl: definiens
3502 % \l_tmpb_tl: notation
3503 \tl_if_empty:NT \l__stex_features_structure_field_str {
3504   % TODO throw error
3505 }
3506 \str_clear:N \l_tmpb_str
3507
3508 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3509 \seq_map_inline:Nn \l_tmpa_seq {
3510   \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
3511   \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
3512   \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
3513     \seq_map_break:n {
3514       \str_set:Nn \l_tmpb_str { ####1 }
3515     }
3516   }
3517 }
3518 \prop_get:cnN { l_stex_symdecl_ \l_tmpb_str _prop } {args}
3519   \l_tmpb_str
3520
3521 \tl_if_empty:NNTF \l_tmpb_tl {
3522   \tl_if_empty:NF \l_tmpa_tl {
3523     \exp_args:Nx \use:n {

```

```

3524         \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fe
3525     }
3526 }
3527 }{
3528     \tl_if_empty:NTF \l_tmpa_tl {
3529         \exp_args:Nx \use:n {
3530             \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\
3531         }
3532     }
3533     }{
3534         \exp_args:Nx \use:n {
3535             \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
3536             \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
3537         }
3538     }
3539 }
3540 % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
3541 % \prop_item:Nn \l_stex_current_module_prop {name} ?
3542 % #3/\l__stex_features_structure_field_str
3543 % \par
3544 % \expandafter\present\csname
3545 %     l_stex_symdecl_
3546 %     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3547 %     \prop_item:Nn \l_stex_current_module_prop {name} ?
3548 %     #3/\l__stex_features_structure_field_str
3549 %     _prop
3550 % \endcsname
3551 }
3552
3553 \tl_clear:N \l__stex_features_structure_def_tl
3554
3555 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3556 \seq_map_inline:Nn \l_tmpa_seq {
3557     \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3558     \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3559     \exp_args:Nx \use:n {
3560         \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
3561
3562         }
3563     }
3564 }
3565
3566 \prop_if_exist:cF {
3567     l_stex_symdecl_
3568     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3569     \prop_item:Nn \l_stex_current_module_prop {name} ?
3570     #3/\l_tmpa_str
3571     _prop
3572 }{
3573     \prop_get:cnN { l_stex_symdecl_ ##1 _prop } {args}
3574     \l_tmpb_str
3575     \exp_args:Nx \use:n {
3576         \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
3577     }
3578 }

```

```

3578 }
3579
3580 \symdecl*[type={\STEXsymbol{module-type}}{
3581   \stex_term_math_oms:nnnn {
3582     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3583     \prop_item:Nn \l__stex_features_structure_prop {name}
3584     }{}{0}{}
3585   }{}{#3}
3586 }
3587 % TODO: -> sms file
3588
3589 \tl_set:cx{ #3 }{
3590   \stex_invoke_structure:nnn {
3591     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3592     \prop_item:Nn \l_stex_current_module_prop {name} ? #3
3593   } {
3594     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3595     \prop_item:Nn \l__stex_features_structure_prop {name}
3596   }
3597 }
3598
3599 }

```

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

\stex\_invoke\_structure:nnn

```

3600 % #1: URI of the instance
3601 % #2: URI of the instantiated module
3602 \cs_new_protected:Nn \stex_invoke_structure:nnn {
3603   \tl_if_empty:nTF{ #3 }{
3604     \prop_set_eq:Nc \l__stex_features_structure_prop {
3605       c_stex_feature_ #2 _prop
3606     }
3607     \tl_clear:N \l_tmpa_tl
3608     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3609     \seq_map_inline:Nn \l_tmpa_seq {
3610       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3611       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3612       \cs_if_exist:cT {
3613         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
3614       }{
3615         \tl_if_empty:NF \l_tmpa_tl {
3616           \tl_put_right:Nn \l_tmpa_tl {,}
3617         }
3618         \tl_put_right:Nx \l_tmpa_tl {
3619           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
3620         }
3621       }
3622     }
3623     \exp_args:No \mathstrut \l_tmpa_tl
3624   }{
3625     \stex_invoke_symbol:n{#1/#3}
3626   }
3627 }

```



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

3628 `\endpackage`

## Chapter 33

# STEX -Statements Implementation

```
3629 <*package>
3630
3631 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3632
3633 \protected\def\ignorespacesandpars{
3634   \begingroup\catcode13=10\relax
3635   \@ifnextchar\par{
3636     \endgroup\expandafter\ignorespacesandpars\@gobble
3637   }{
3638     \endgroup
3639   }
3640 }
3641
3642 <@@=stex_statements>
3643
3644   Warnings and error messages
```

\titleemph

```
3644 \def\titleemph#1{\textbf{#1}}
```

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

### 33.1 Definitions

definiendum

```
3645 \keys_define:nn {stex / definiendum }{
3646   post      .tl_set:N      = \l__stex_statements_definiendum_post_tl,
3647   root      .str_set_x:N    = \l__stex_statements_definiendum_root_str,
3648   gfa       .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
3649 }
3650 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
3651   \str_clear:N \l__stex_statements_definiendum_root_str
3652   \tl_clear:N \l__stex_statements_definiendum_post_tl
3653   \str_clear:N \l__stex_statements_definiendum_gfa_str
```

```

3654 \keys_set:nn { stex / definiendum } { #1 }
3655 }
3656 \NewDocumentCommand \definiendum { 0{} m m } {
3657   \__stex_statements_definiendum_args:n { #1 }
3658   \stex_get_symbol:n { #2 }
3659   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3660   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
3661     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
3662       \tl_set:Nn \l_tmpa_tl { #3 }
3663     } {
3664       \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
3665       \tl_set:Nn \l_tmpa_tl {
3666         \l__stex_statements_definiendum_root_str\l__stex_statements_definiendum_post_tl
3667       }
3668     }
3669   } {
3670     \tl_set:Nn \l_tmpa_tl { #3 }
3671   }
3672
3673   % TODO root
3674   \rustex_if:TF {
3675     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
3676   } {
3677     \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
3678   }
3679 }
3680 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

#### definame

```

3681 \NewDocumentCommand \definame { 0{} m } {
3682   \__stex_statements_definiendum_args:n { #1 }
3683   % TODO: root
3684   \stex_get_symbol:n { #2 }
3685   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3686   \str_set:Nx \l_tmpa_str {
3687     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3688   }
3689   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3690   \rustex_if:TF {
3691     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3692       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3693     }
3694   } {
3695     \defemph@uri {
3696       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3697     } { \l_stex_get_symbol_uri_str }
3698   }
3699 }
3700 \stex_deactivate_macro:Nn \definame {definition~environments}

```

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

sdefinition

```

3701
3702 \keys_define:nn {stex / sdefinition }{
3703   type      .str_set_x:N = \sdefinitiontype,
3704   id        .str_set_x:N = \sdefinitionid,
3705   title     .tl_set:N    = \sdefinitiontitle
3706 }
3707 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
3708   \str_clear:N \sdefinitiontype
3709   \str_clear:N \sdefinitionid
3710   \tl_clear:N \sdefinitiontitle
3711   \keys_set:nn { stex / sdefinition }{ #1 }
3712 }
3713
3714 \NewDocumentEnvironment{sdefinition}{0{}}{
3715   \__stex_statements_sdefinition_args:n{ #1 }
3716   \stex_reactivate_macro:N \definiendum
3717   \stex_reactivate_macro:N \definame
3718   \stex_smsmode_set_codes:
3719   \clist_set:No \l_tmpa_clist \sdefinitiontype
3720   \tl_clear:N \l_tmpa_tl
3721   \clist_map_inline:Nn \l_tmpa_clist {
3722     \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
3723       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
3724     }
3725   }
3726   \tl_if_empty:NTF \l_tmpa_tl {
3727     \__stex_statements_sdefinition_start:
3728   }{
3729     \l_tmpa_tl
3730   }
3731   \stex_ref_new_doc_target:n \sdefinitionid
3732   \stex_if_smsmode:F {
3733     \exp_args:Nnnx
3734     \begin{stex_annotate_env}{definition}{}
3735     \str_if_empty:NF \sdefinitiontype {
3736       \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{}
3737     }
3738   }
3739 }{
3740   \stex_if_smsmode:F {
3741     \end{stex_annotate_env}
3742   }
3743   \clist_set:No \l_tmpa_clist \sdefinitiontype
3744   \tl_clear:N \l_tmpa_tl
3745   \clist_map_inline:Nn \l_tmpa_clist {
3746     \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
3747       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
3748     }
3749   }
3750   \tl_if_empty:NTF \l_tmpa_tl {
3751     \__stex_statements_sdefinition_end:
3752   }{
3753     \l_tmpa_tl

```

```

3754 }
3755 }

```

`\stexpatchdefinition`

```

3756 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
3757   \par\noindent\titllemph{Definition\tl_if_empty:NF \sdefinitiontitle {
3758     ~(\sdefinitiontitle)
3759   }~}
3760 }
3761 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\par\medskip}
3762
3763 \newcommand\stexpatchdefinition[3] [] {
3764   \str_set:Nx \l_tmpa_str{ #1 }
3765   \str_if_empty:NTF \l_tmpa_str {
3766     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
3767     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
3768   }{
3769     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2
3770     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
3771   }
3772 }

```

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

`\inlinedef inline:`

```

3773 \NewDocumentCommand \inlinedef { m } {
3774   \begingroup
3775   \stex_reactivate_macro:N \definiendum
3776   \stex_reactivate_macro:N \definame
3777   \stex_ref_new_doc_target:n{}
3778   #1
3779   \endgroup
3780 }

```

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

## 33.2 Assertions

`sassertion`

```

3781
3782 \keys_define:nn {stex / sassertion }{
3783   type      .str_set_x:N = \sassertiontype,
3784   id        .str_set_x:N = \sassertionid,
3785   title     .tl_set:N     = \sassertiontitle ,
3786   name      .str_set_x:N = \sassertionname
3787 }
3788 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
3789   \str_clear:N \sassertiontype
3790   \str_clear:N \sassertionid
3791   \str_clear:N \sassertionname
3792   \tl_clear:N \sassertiontitle
3793   \keys_set:nn { stex / sassertion }{ #1 }
3794 }

```

```

3795
3796 \tl_new:N \g__stex_statements_aftergroup_tl
3797
3798 \NewDocumentEnvironment{sassertion}{0{}}{
3799   \__stex_statements_sassertion_args:n{ #1 }
3800   \stex_smsmode_set_codes:
3801   \clist_set:No \l_tmpa_clist \sassertiontype
3802   \tl_clear:N \l_tmpa_tl
3803   \clist_map_inline:Nn \l_tmpa_clist {
3804     \tl_if_exist:cT {\__stex_statements_sassertion_##1_start:}{
3805       \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_statements_sassertion_##1_start:}}
3806     }
3807   }
3808   \tl_if_empty:NTF \l_tmpa_tl {
3809     \__stex_statements_sassertion_start:
3810   }{
3811     \l_tmpa_tl
3812   }
3813   \stex_ref_new_doc_target:n \sassertionid
3814   \stex_if_smsmode:F {
3815     \exp_args:Nnnx
3816     \begin{stex_annotate_env}{assertion}{}
3817     \str_if_empty:NF \sassertiontype {
3818       \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
3819     }
3820   }
3821 }{
3822   \stex_if_smsmode:F {
3823     \end{stex_annotate_env}
3824   }
3825   \clist_set:No \l_tmpa_clist \sassertiontype
3826   \tl_clear:N \l_tmpa_tl
3827   \clist_map_inline:Nn \l_tmpa_clist {
3828     \tl_if_exist:cT {\__stex_statements_sassertion_##1_end:}{
3829       \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_statements_sassertion_##1_end:}}
3830     }
3831   }
3832   \tl_if_empty:NTF \l_tmpa_tl {
3833     \__stex_statements_sassertion_end:
3834   }{
3835     \l_tmpa_tl
3836   }
3837   \str_if_empty:NF \sassertionname {
3838     \tl_gset:Nx \g__stex_statements_aftergroup_tl {
3839       \symdecl*{\sassertionname}
3840     }
3841     \aftergroup\g__stex_statements_aftergroup_tl
3842   }
3843 }

```

\stexpatchassertion

```

3844
3845 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
3846   \par\noindent\titllemph{Assertion~\tl_if_empty:NF \sassertiontitle {

```

```

3847     (\sassertiontitle)
3848   }~}
3849 }
3850 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
3851
3852 \newcommand\stexpatchassertion[3] [] {
3853   \str_set:Nx \l_tmpa_str{ #1 }
3854   \str_if_empty:NTF \l_tmpa_str {
3855     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
3856     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
3857   }{
3858     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
3859     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
3860   }
3861 }

```

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

`\inlineass` inline:

```

3862 \NewDocumentCommand \inlineass { m } {
3863   \begingroup
3864   \stex_ref_new_doc_target:n{
3865     #1
3866   }
3867 }

```

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

## 33.3 Examples

`sexample`

```

3868
3869 \keys_define:nn {stex / sexample }{
3870   type      .str_set_x:N = \exampletype,
3871   id        .str_set_x:N = \sexampleid,
3872   title     .tl_set:N = \sexampletile,
3873   for       .clist_set:N = \sexamplefor,
3874 }
3875 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
3876   \str_clear:N \sexampletype
3877   \str_clear:N \sexampleid
3878   \tl_clear:N \sexampletile
3879   \clist_clear:N \sexamplefor
3880   \keys_set:nn { stex / sexample }{ #1 }
3881 }
3882
3883 \NewDocumentEnvironment{sexample}{0{}}{
3884   \__stex_statements_sexample_args:n{ #1 }
3885   \stex_smsmode_set_codes:
3886   \clist_set:N \l_tmpa_clist \sexampletype
3887   \tl_clear:N \l_tmpa_tl
3888   \clist_map_inline:Nn \l_tmpa_clist {
3889     \tl_if_exist:cT {\__stex_statements_sexample_##1_start:}{

```

```

3890     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
3891   }
3892 }
3893 \tl_if_empty:NTF \l_tmpa_tl {
3894   \__stex_statements_sexample_start:
3895 }{
3896   \l_tmpa_tl
3897 }
3898 \stex_ref_new_doc_target:n \sexampleid
3899 \stex_if_smsmode:F {
3900   \seq_clear:N \l_tmpa_seq
3901   \clist_map_inline:Nn \sexamplefor {
3902     \str_if_eq:nnF{ ##1 }{{}{
3903       \stex_get_symbol:n { ##1 }
3904       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
3905         \l_stex_get_symbol_uri_str
3906       }
3907     }
3908   }
3909   \exp_args:Nnnx
3910   \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
3911   \str_if_empty:NF \sexamplotype {
3912     \stex_annotate_invisible:nnn{type}{\sexampletype}{}
3913   }
3914 }
3915 }{
3916   \stex_if_smsmode:F {
3917     \end{stex_annotate_env}
3918   }
3919   \clist_set:Nn \l_tmpa_clist \sexamplotype
3920   \tl_clear:N \l_tmpa_tl
3921   \clist_map_inline:Nn \l_tmpa_clist {
3922     \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
3923       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
3924     }
3925   }
3926   \tl_if_empty:NTF \l_tmpa_tl {
3927     \__stex_statements_sexample_end:
3928   }{
3929     \l_tmpa_tl
3930   }
3931 }

```

\stexpatchexample

```

3932
3933 \cs_new_protected:Nn \__stex_statements_sexample_start: {
3934   \par\noindent\titleemph{Example~\tl_if_empty:NF \sexamplotype {
3935     (\sexamplotype)
3936   }~}
3937 }
3938 \cs_new_protected:Nn \__stex_statements_sexample_end: {\par\medskip}
3939
3940 \newcommand\stexpatchexample[3]{} {
3941   \str_set:Nx \l_tmpa_str{ #1 }

```



```

3942 \str_if_empty:NTF \l_tmpa_str {
3943   \tl_set:Nn \__stex_statements_sexample_start: { #2 }
3944   \tl_set:Nn \__stex_statements_sexample_end: { #3 }
3945 }{
3946   \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
3947   \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
3948 }
3949 }

```

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

`\inlineex` inline:

```

3950 \NewDocumentCommand \inlineex { m } {
3951   \begingroup
3952   \stex_ref_new_doc_target:n{
3953     #1
3954   \endgroup
3955 }

```

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

## 33.4 Logical Paragraphs

`sparagraph`

```

3956 \keys_define:nn { stex / spparagraph } {
3957   id      .str_set_x:N = \sparagraphid ,
3958   title   .tl_set:N    = \l_stex_sparagraph_title_tl ,
3959   type    .str_set_x:N = \sparagraphtype ,
3960   for     .str_set_x:N = \sparagraphfor ,
3961   from    .tl_set_x:N   = \sparagraphfrom ,
3962   start   .tl_set:N     = \l_stex_sparagraph_start_tl ,
3963   name    .str_set:N    = \sparagraphname
3964 }
3965
3966 \cs_new_protected:Nn \stex_sparagraph_args:n {
3967   \tl_clear:N \l_stex_sparagraph_title_tl
3968   \tl_clear:N \sparagraphfrom
3969   \tl_clear:N \l_stex_sparagraph_start_tl
3970   \str_clear:N \sparagraphid
3971   \str_clear:N \sparagraphtype
3972   \str_clear:N \sparagraphfor
3973   \str_clear:N \sparagraphname
3974   \keys_set:nn { stex / spparagraph }{ #1 }
3975 }
3976 \newif\if@in@omtext\@in@omtextfalse
3977
3978 \NewDocumentEnvironment {sparagraph} { 0{} } {
3979   \stex_sparagraph_args:n { #1 }
3980   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
3981     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
3982   }{
3983     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
3984   }

```

```

3985 \@@in@omtexttrue
3986 \stex_smsmode_set_codes:
3987 \clist_set:No \l_tmpa_clist \sparagraphtype
3988 \tl_clear:N \l_tmpa_tl
3989 \clist_map_inline:Nn \l_tmpa_clist {
3990   \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
3991     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
3992   }
3993 }
3994 \tl_if_empty:NTF \l_tmpa_tl {
3995   \__stex_statements_sparagraph_start:
3996 }{
3997   \l_tmpa_tl
3998 }
3999 \stex_ref_new_doc_target:n \sparagraphid
4000 \stex_if_smsmode:F {
4001   \exp_args:Nnnx
4002   \begin{stex_annotate_env}{paragraph}{}
4003   \str_if_empty:NF \sparagraphtype {
4004     \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}
4005   }
4006 }
4007 \ignorespacesandpars
4008 }{
4009   \stex_if_smsmode:F {
4010     \end{stex_annotate_env}
4011   }
4012   \clist_set:No \l_tmpa_clist \sparagraphtype
4013   \tl_clear:N \l_tmpa_tl
4014   \clist_map_inline:Nn \l_tmpa_clist {
4015     \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
4016       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
4017     }
4018   }
4019   \tl_if_empty:NTF \l_tmpa_tl {
4020     \__stex_statements_sparagraph_end:
4021   }{
4022     \l_tmpa_tl
4023   }
4024   \str_if_empty:NF \sparagraphname {
4025     \tl_gset:Nx \g__stex_statements_aftergroup_tl {
4026       \symdecl*{\sparagraphname}
4027     }
4028     \aftergroup\g__stex_statements_aftergroup_tl
4029   }
4030 }

```

\stexpatchparagraph

```

4031
4032 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
4033   \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4034     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
4035       \titleemph{\l_stex_sparagraph_title_tl}:~
4036     }

```

```

4037   }{
4038     \titleemph{\l_stex_sparagraph_start_tl}~
4039   }
4040 }
4041 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
4042
4043 \newcommand\stexpatchparagraph[3] [] {
4044   \str_set:Nx \l_tmpa_str{ #1 }
4045   \str_if_empty:NTF \l_tmpa_str {
4046     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
4047     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
4048   }{
4049     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
4050     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
4051   }
4052 }

```

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

#### symboldoc

```

4053 \NewDocumentEnvironment{symboldoc}{ m }{
4054   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
4055   \seq_clear:N \l_tmpb_seq
4056   \seq_map_inline:Nn \l_tmpa_seq {
4057     \str_if_eq:nnF{ ##1 }{}{
4058       \stex_get_symbol:n { ##1 }
4059       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
4060         \l_stex_get_symbol_uri_str
4061       }
4062     }
4063   }
4064   \par
4065   \exp_args:Nnnx
4066   \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
4067 }{
4068   \end{stex_annotate_env}
4069 }
4070 \</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>

```
4071 <*package>
4072 <@@=stex_sproof>
4073
4074 %%%%%%%%%%% sproof.dtx %%%%%%%%%%%
4075
```

### 34.2 Proofs

We first define some keys for the proof environment.

```
4076 \keys_define:nn { stex / spf } {
4077   id          .str_set:N = \l__stex_sproof_spf_id_str,
4078   display     .tl_set:N  = \l__stex_sproof_spf_display_tl,
4079   for         .tl_set:N  = \l__stex_sproof_spf_for_tl ,
4080   from        .tl_set:N  = \l__stex_sproof_spf_from_tl ,
4081   proofend    .tl_set:N  = \l__stex_sproof_spf_proofend_tl,
4082   type        .tl_set:N  = \l__stex_sproof_spf_type_tl,
4083   title       .tl_set:N  = \l__stex_sproof_spf_title_tl,
4084   continues   .tl_set:N  = \l__stex_sproof_spf_continues_tl,
4085   functions   .tl_set:N  = \l__stex_sproof_spf_functions_tl,
4086   method      .tl_set:N  = \l__stex_sproof_spf_method_tl
4087 }
4088 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
4089   \str_clear:N \l__stex_sproof_spf_id_str
4090   \tl_clear:N \l__stex_sproof_spf_display_tl
4091   \tl_clear:N \l__stex_sproof_spf_for_tl
4092   \tl_clear:N \l__stex_sproof_spf_from_tl
4093   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
4094   \tl_clear:N \l__stex_sproof_spf_type_tl
4095   \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

```

4096 \tl_clear:N \l__stex_sproof_spf_continues_tl
4097 \tl_clear:N \l__stex_sproof_spf_functions_tl
4098 \tl_clear:N \l__stex_sproof_spf_method_tl
4099 \keys_set:nn { stex / spf }{ #1 }
4100 }

```

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

```

4101 \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.

```

4102 \newcount\count_ten
4103 \newenvironment{pst@with@label}[1]{
4104   \edef\pst@label{#1}
4105   \advance\count_ten by 1\relax
4106   \count_ten=1
4107 }{
4108   \advance\count_ten by -1\relax
4109 }

```

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

```

4110 \def\the@pst@label{
4111   \pst@make@label\pst@label{\number\count_ten}\l__stex_sproof_pstlabel_postfix_tl
4112 }

```

*(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.

```

4113 \keys_define:nn { stex / pstlabel }{
4114   prefix      .tl_set:N   = \l__stex_sproof_pstlabel_prefix_tl,
4115   delimiter   .tl_set:N   = \l__stex_sproof_pstlabel_delimiter_tl,
4116   postfix     .tl_set:N   = \l__stex_sproof_pstlabel_postfix_tl
4117 }
4118 \cs_new_protected:Nn \__stex_sproof_pstlabel_args:n {

```

---

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

```

4119 \tl_set:Nn \l__stex_sproof_pstlabel_prefix_tl {P}
4120 \tl_set:Nn \l__stex_sproof_pstlabel_delimiter_tl {.}
4121 \tl_clear:N \l__stex_sproof_pstlabel_postfix_tl
4122 }
4123 \__stex_sproof_pstlabel_args:n {}
4124 \newcommand\setpstlabelstyle[1]{
4125   \__stex_sproof_pstlabel_args:n {#1}
4126 }
4127 \newcommand\setpstlabelstyledefault{%
4128   \__stex_sproof_pstlabel_args:n{prefix=P,delimiter=.,postfix={}}
4129 }

```

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

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

```

4130 \ExplSyntaxOff
4131 \def\pst@make@label@long#1#2{\@for\@I:=#1\do{\expandafter\expandafter\expandafter\@I\csname
4132 \def\pst@make@label@angles#1#2{\ensuremath{\@for\@I:=#1\do{\rangle}}#2}
4133 \def\pst@make@label@short#1#2{#2}
4134 \def\pst@make@label@empty#1#2{}
4135 \ExplSyntaxOn
4136 \def\pstlabelstyle#1{%
4137   \def\pst@make@label{\use:c{pst@make@label@#1}}%
4138 }%
4139 \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.

```

4140 \def\next@pst@label{%
4141   \global\advance\count\count10 by 1%
4142 }%

```

(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

```

4143 \def\sproof@box{
4144   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
4145 }
4146 \def\spf@proofend{\sproof@box}
4147 \def\sproofend{
4148   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
4149     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
4150   }
4151 }
4152 \def\sProofEndSymbol#1{\def\sproof@box{#1}}

```

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

**spf@\*@kw**

```

4153 \def\spf@proofsketch@kw{Proof Sketch}
4154 \def\spf@proof@kw{Proof}
4155 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

4156 \cs_if_exist:NT \bbl@loaded {
4157   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4158   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4159     \input{proof-ngerman.lda}
4160   }
4161   \clist_if_in:NnT \l_tmpa_clist {finnish}{
4162     \input{proof-finnish.lda}
4163   }
4164   \clist_if_in:NnT \l_tmpa_clist {french}{
4165     \input{proof-french.lda}
4166   }
4167   \clist_if_in:NnT \l_tmpa_clist {russian}{
4168     \input{proof-russian.lda}
4169   }
4170 }
4171

```

**spfsketch**

```

4172 \newcommand\spfsketch[2][]{
4173   \__stex_sproof_spf_args:n{#1}
4174   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4175     \titleemph{
4176       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
4177         \spf@proofsketch@kw
4178       }{
4179         \l__stex_sproof_spf_type_tl
4180       }
4181     }:
4182   }
4183   {-#2}
4184   %\sref@label@id{this \ifx\spf@type\@empty\spf@proofsketch@kw\else\spf@type\fi}
4185   \sproofend
4186 }

```

(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>

```

4187 \newenvironment{spfeq}[2][]{
4188   \__stex_sproof_spf_args:n{#1}
4189   %\sref@target
4190   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4191     \titleemph{
4192       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
4193         \spf@proof@kw
4194       }{
4195         \l__stex_sproof_spf_type_tl
4196       }
4197     }:

```

<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

```

4198 }
4199 {~#2}
4200 \begin{displaymath}\begin{array}{rcll}
4201 }{
4202 \end{array}\end{displaymath}
4203 }

```

(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.

```

4204 \newenvironment{spf@proof}[2][]{
4205   \__stex_sproof_spf_args:n{#1}
4206   %\sref@target
4207   \count_ten=10
4208   \par\noindent
4209   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4210     \titleemph{
4211       \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {
4212         \spf@proof@kw
4213       }{
4214         \l__stex_sproof_spf_type_tl
4215       }
4216     }:
4217   }
4218   {~#2}
4219   %\sref@label@id{this \ifx\spf@type\@empty\spf@proof@kw\else\spf@type\fi}
4220   \def\pst@label{}
4221   \newcount\pst@count% initialize the labeling mechanism
4222   \begin{description}\begin{pst@with@label}{\l__stex_sproof_pstlabel_prefix_tl}
4223   }{
4224     \end{pst@with@label}\end{description}
4225   }
4226 \newenvironment{sproof}[2][{\begin{spf@proof}[#1][#2]}{\sproofend\end{spf@proof}}
4227 \newenvironment{sProof}[2][{\begin{spf@proof}[#1][#2]}{\end{spf@proof}}

```

**\spfidea**

```

4228 \newcommand\spfidea[2][]{
4229   \__stex_sproof_spf_args:n{#1}
4230   \titleemph{
4231     \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {Proof~Idea}{
4232       \l__stex_sproof_spf_type_tl
4233     }:
4234   }~#2
4235   \sproofend
4236 }

```

(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

```

4237 \newenvironment{spfstep}[1][]{
4238   \_stex_sproof_spf_args:n{#1}
4239   \@in@omtexttrue
4240   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4241     \item[\the@pst@label]
4242   }
4243   \tl_if_empty:NF \l__stex_sproof_spf_title_tl {
4244     {(\titleemph{\l__stex_sproof_spf_title_tl})\enspace}
4245   }
4246   %\sref@label@id{\pst@label}
4247   \ignorespacesandpars
4248 }{
4249   \next@pst@label\ignorespacesandpars
4250 }

```

sproofcomment

```

4251 \newenvironment{sproofcomment}[1][]{
4252   \_stex_sproof_spf_args:n{#1}
4253   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4254     \item[\the@pst@label]
4255   }
4256 }{
4257   \next@pst@label
4258 }

```

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.

```

4259 \newenvironment{subproof}[2][]{
4260   \_stex_sproof_spf_args:n{#1}
4261   \def\@test{#2}
4262   \ifx\@test\empty\else
4263     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4264       \item[\the@pst@label]
4265     }{#2}
4266   \fi
4267   \begin{pst@with@label}{\pst@label,\number\count_ten}
4268 }{
4269   \end{pst@with@label}\next@pst@label
4270 }

```

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

```

4271 \newenvironment{spfcases}[2][]{
4272   \def\@test{#1}
4273   \ifx\@test\empty
4274     \begin{subproof}[method=by-cases]{#2}
4275   \else
4276     \begin{subproof}[#1,method=by-cases]{#2}
4277   \fi
4278 }{

```

---

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

```

4279 \end{subproof}
4280 }

```

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

```

4281 \newenvironment{spfcase}[2] [] {
4282   \__stex_sproof_spf_args:n{#1}
4283   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4284     \item[\the@pst@label]
4285   }
4286   \def\@test{#2}
4287   \ifx\@test\@empty
4288   \else
4289     {\titleemph{#2}:~}
4290   \fi
4291   \begin{pst@with@label}{\pst@label,\number\count_ten}
4292 }{
4293   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4294     \sproofend
4295   }
4296   \end{pst@with@label}
4297   \next@pst@label
4298 }

```

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

```

4299 \newcommand\spfcasesketch[3] [] {
4300   \__stex_sproof_spf_args:n{#1}
4301   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4302     \item[\the@pst@label]
4303   }
4304   \def\@test{#2}
4305   \ifx\@test\@empty
4306   \else
4307     {\titleemph{#2}:~}
4308   \fi#3
4309   \next@pst@label
4310 }%

```

### 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.

```

4311 \keys_define:nn { stex / just }{
4312   id          .str_set:x:N = \l__stex_sproof_just_id_str,
4313   method      .tl_set:N    = \l__stex_sproof_just_method_tl,
4314   premises    .tl_set:N    = \l__stex_sproof_just_premises_tl,
4315   args        .tl_set:N    = \l__stex_sproof_just_args_tl
4316 }

```

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**

```
4317 \newenvironment{justification}[1] [] {}{}
```

**\premise**

```
4318 \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.

```
4319 \newcommand\justarg[2] [] {#2}
```

```
4320 \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

```
4321 <*package>
4322
4323 %%%%%%%%%% others.dtx %%%%%%%%%%
4324
4325 <@@=stex_others>
    Warnings and error messages
4326 % None

\MSC Math subject classifier

4327 \NewDocumentCommand \MSC {m} {
4328 % TODO
4329 }

(End definition for \MSC. This function is documented on page 21.)
    Patching tikzinput, if loaded
4330 \@ifpackageloaded{tikzinput}{
4331 \RequirePackage{stex-tikzinput}
4332 }{}
4333 </package>
```

## Chapter 36

# STEX -Metatheory Implementation

```
4334 <*package>
4335 <@@=stex_modules>
4336
4337 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
4338
4339 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
4340 \begingroup
4341 \stex_module_setup:nn{
4342   ns=\c_stex_metatheory_ns_str,
4343   meta=NONE
4344 }{Metatheory}
4345 \stex_reactivate_macro:N \symdecl
4346 \stex_reactivate_macro:N \notation
4347 \stex_reactivate_macro:N \symdef
4348 \ExplSyntaxOff
4349 \csname stex_suppress_html:n\endcsname{
4350   % is-a (a:A, a \in A, a is an A, etc.)
4351   \symdecl[args=ai]{isa}
4352   \notation[typed]{isa}{#1 \comp{:} #2}{#1 \comp, #2}
4353   \notation[in]{isa}{#1 \comp\in #2}{#1 \comp, #2}
4354   \notation[pred]{isa}{#2\comp(#1 \comp)}{#1 \comp, #2}
4355
4356   % bind (\forall, \Pi, \lambda etc.)
4357   \symdecl[args=Bi]{bind}
4358   \notation[forall]{bind}{\comp\forall #1.\;#2}{#1 \comp, #2}
4359   \notation[\Pi]{bind}{\comp\prod_{#1}#2}{#1 \comp, #2}
4360   \notation[deffun]{bind}{\comp( #1 \comp)\; \to\;}{#1 \comp, #2}
4361
4362   % dummy variable
4363   \symdecl{dummyvar}
4364   \notation[underscore]{dummyvar}{\comp\_}
4365   \notation[dot]{dummyvar}{\comp\cdot}
4366   \notation[dash]{dummyvar}{\comp{\rm --}}
4367
4368   %fromto (function space, Hom-set, implication etc.)
```

```

4369 \symdecl[args=ai]{fromto}
4370 \notation[xarrow]{fromto}{#1 \comp\to #2}{#1 \comp\times #2}
4371 \notation[arrow]{fromto}{#1 \comp\to #2}{#1 \comp\to #2}
4372
4373 % mapto (lambda etc.)
4374 %\symdecl[args=Bi]{mapto}
4375 %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
4376 %\notation[lambda]{mapto}{\comp\lambda #1 \comp. \; #2}{#1 \comp, #2}
4377 %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp. \; #2}{#1 \comp, #2}
4378
4379 % function/operator application
4380 \symdecl[args=ia]{apply}
4381 \notation[prec=0;0x\infpres,parens]{apply}{#1 \comp( #2 \comp)}{#1 \comp, #2}
4382 \notation[prec=0;0x\infpres,lambda]{apply}{#1 \; #2 }{#1 \; #2}
4383
4384 % ‘‘type’’ of all collections (sets, classes, types, kinds)
4385 \symdecl{collection}
4386 \notation[U]{collection}{\comp{\mathcal{U}}}
4387 \notation[set]{collection}{\comp{\textsf{Set}}}
4388
4389 % sequences
4390 \symdecl[args=1]{seqtype}
4391 \notation[kleene]{seqtype}{#1^{\comp\ast}}
4392
4393 \symdef[args=2,li,prec=nobrackets]{sequence-index}{#1_{#2}}
4394 \notation[ui,prec=nobrackets]{sequence-index}{#1^{#2}}
4395
4396 %\symdef[args=3,li]{sequence-from-to}{#1_{#2}\comp{\,\ellipses\,}#1_{#3}}
4397 %\notation[ui]{sequence-from-to}{#1^{#2}\comp{\,\ellipses\,}#1^{#3}}
4398 % ^ superceded by \aseqfromto and \livar/\uivar
4399
4400 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{\,\ellipses\,}}{#1\comp,#2}
4401 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{\,\ellipses\,}#2}{#1\comp,#2}
4402 \symdef[args=aui,prec=nobrackets]{aseqfromtovia}{#1\comp{\,\ellipses\,}#2\comp{\,\ellipses\,}#3}{#1\comp,#2}
4403
4404 % letin (‘‘let’’, local definitions, variable substitution)
4405 \symdecl[args=bii]{letin}
4406 \notation[let]{letin}{\comp{\rm let}}{\;#1\comp{=}\;#2\; \comp{\rm in}}{\;#3}
4407 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
4408 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
4409
4410 % structures
4411 \symdecl*[args=1]{module-type}
4412 \notation{module-type}{\mathtt{MOD} #1}
4413 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
4414 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{#1 \comp, #2}
4415
4416 }
4417 \ExplSyntaxOn
4418 \stex_add_to_current_module:n{
4419   \let\nappa\apply
4420   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
4421   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
4422   \def\livar{\csname sequence-index\endcsname[li]}

```

```

4423 \def\uivar{\csname sequence-index\endcsname[ui]}
4424 \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
4425 \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
4426 \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
4427 }
4428 \__stex_modules_end_module:
4429 \endgroup
4430 \</package>

```

## Chapter 37

# Tikzinput Implementation

```
4431 <*package>
4432
4433 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
4434
4435 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
4436 \RequirePackage{l3keys2e}
4437
4438 \keys_define:nn { tikzinput } {
4439   image .bool_set:N = \c_tikzinput_image_bool,
4440   image .default:n = false ,
4441   unknown .code:n = {}
4442 }
4443
4444 \ProcessKeysOptions { tikzinput }
4445
4446 \bool_if:NTF \c_tikzinput_image_bool {
4447   \RequirePackage{graphicx}
4448
4449   \providecommand\usetikzlibrary[]{}
4450   \newcommand\tikzinput[2] []{\includegraphics[#1]{#2}}
4451 }{
4452   \RequirePackage{tikz}
4453   \RequirePackage{standalone}
4454
4455   \newcommand \tikzinput [2] [] {
4456     \setkeys{Gin}{#1}
4457     \ifx \Gin@ewidth \Gin@exclamation
4458       \ifx \Gin@eheight \Gin@exclamation
4459         \input { #2 }
4460       \else
4461         \resizebox{!}{ \Gin@eheight }{
4462           \input { #2 }
4463         }
4464       \fi
4465     \else
4466       \ifx \Gin@eheight \Gin@exclamation
4467         \resizebox{ \Gin@ewidth }{!}{
4468           \input { #2 }
```



```

4469     }
4470     \else
4471         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
4472             \input { #2 }
4473         }
4474     \fi
4475 \fi
4476 }
4477 }
4478
4479 \newcommand \ctikzinput [2] [] {
4480     \begin{center}
4481         \tikzinput [1] {#2}
4482     \end{center}
4483 }
4484
4485 \@ifpackageloaded{stex}{
4486     \RequirePackage{stex-tikzinput}
4487 }{}
4488
4489 </package>
4490 <*stex>
4491 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
4492 \RequirePackage{stex}
4493 \RequirePackage{tikzinput}
4494
4495 \newcommand\mhtikzinput [2] [] {%
4496     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
4497     \stex_in_repository:nn\Gin@mhrepos{
4498         \tikzinput [1]{\mhpath{##1}{#2}}
4499     }
4500 }
4501 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [1] {#2}\end{center}}
4502 </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.

```
4503 \*cls)
4504 \@@=document_structure)
4505 \ProvidesExplClass{omdoc}{2020/10/19}{1.4}{OMDoc Documents}
4506 \RequirePackage{l3keys2e,expl-keystr-compatible}
```

### 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`

```
4507 \keys_define:nn{ document-structure / pkg }{
4508   class      .str_set_x:N = \c_document_structure_class_str,
4509   minimal    .bool_set:N = \c_document_structure_minimal_bool,
4510   report     .code:n      = {
4511     \ClassWarning{omdoc}{the option 'report' is deprecated, use 'class=report', instead}
4512     \str_set:Nn \c_document_structure_class_str {report}
4513   },
4514   book       .code:n      = {
4515     \ClassWarning{omdoc}{the option 'book' is deprecated, use 'class=book', instead}
4516     \str_set:Nn \c_document_structure_class_str {book}
4517   },
4518   bookpart   .code:n      = {
4519     \ClassWarning{omdoc}{the option 'bookpart' is deprecated, use 'class=book,topsect=chapter}
4520     \str_set:Nn \c_document_structure_class_str {book}
4521     \str_set:Nn \c_document_structure_topsect_str {chapter}
4522   },
```

```

4523 docopt      .str_set_x:N = \c_document_structure_docopt_str,
4524 unknown     .code:n      = {
4525   \PassOptionsToPackage{ \CurrentOption }{ omdoc }
4526 }
4527 }
4528 \ProcessKeysOptions{ document-structure / pkg }
4529 \str_if_empty:NT \c_document_structure_class_str {
4530   \str_set:Nn \c_document_structure_class_str {article}
4531 }
4532 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
4533   {\c_document_structure_class_str}
4534

```

### 38.3 Beefing up the document environment

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

```

4535 \RequirePackage{omdoc}
4536 \bool_if:NF \c_document_structure_minimal_bool {
4537   \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>

```

4538 \keys_define:nn { document-structure / document }{
4539   id .str_set_x:N = \c_document_structure_document_id_str
4540 }
4541 \let\__document_structure_orig_document=\document
4542 \renewcommand{\document}[1][]{
4543   \keys_set:nn{ document-structure / document }{ #1 }
4544   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
4545   \__document_structure_orig_document
4546 }

```

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

```

4547 }
4548 \</cls>

```

### 38.4 Implementation: OMDoc Package

```

4549 \*package>
4550 \ProvidesExplPackage{omdoc}{2020/10/19}{1.4}{OMDoc document Structure}
4551 \RequirePackage{expl-keys-compact,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

```

4552
4553 \keys_define:nn{ document-structure / pkg }{
4554   class      .str_set_x:N = \c_document_structure_class_str,
4555   topsect    .str_set_x:N = \c_document_structure_topsect_str,
4556   % showignores .bool_set:N = \c_document_structure_showignores_bool,
4557 }
4558 \ProcessKeysOptions{ document-structure / pkg }
4559 \str_if_empty:NT \c_document_structure_class_str {
4560   \str_set:Nn \c_document_structure_class_str {article}
4561 }
4562 \str_if_empty:NT \c_document_structure_topsect_str {
4563   \str_set:Nn \c_document_structure_topsect_str {section}
4564 }

```

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

```

4565 \RequirePackage{xspace}
4566 \RequirePackage{comment}
4567 \@ifpackageloaded{babel}{\RequirePackage[base]{babel}}

```

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

```

4568 \@ifpackageloaded{babel}{
4569   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4570   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4571     \input{omdoc-ngerman.ldf}
4572   }
4573 }{}
4574 %\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.

```

4575 \int_new:N \l_document_structure_section_level_int
4576 \str_case:VnF \c_document_structure_topsect_str {
4577   {part}{
4578     \int_set:Nn \l_document_structure_section_level_int {0}
4579   }
4580   {chapter}{
4581     \int_set:Nn \l_document_structure_section_level_int {1}
4582   }
4583 }{
4584   \str_case:VnF \c_document_structure_class_str {
4585     {book}{
4586       \int_set:Nn \l_document_structure_section_level_int {0}
4587     }
4588     {report}{
4589       \int_set:Nn \l_document_structure_section_level_int {0}
4590     }
4591   }{
4592     \int_set:Nn \l_document_structure_section_level_int {2}
4593   }
4594 }

```

## 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

```
4595 \def\current@section@level{document}%
4596 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
4597 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
4598 \cs_new_protected:Npn \skipomgroup {
4599   \ifcase\l_document_structure_section_level_int
4600   \or\stepcounter{part}
4601   \or\stepcounter{chapter}
4602   \or\stepcounter{section}
4603   \or\stepcounter{subsection}
4604   \or\stepcounter{subsubsection}
4605   \or\stepcounter{paragraph}
4606   \or\stepcounter{subparagraph}
4607   \fi
4608 }
```

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

`blindomgroup`

```
4609 \newcommand\at@begin@blindomgroup[1]{%
4610 \newenvironment{blindomgroup}
4611 {
4612   \int_incr:N\l_document_structure_section_level_int
4613   \at@begin@blindomgroup\l_document_structure_section_level_int
4614 }{}}
```

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

```
4615 \newcommand\omgroup@nonum[2]{
4616   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
4617   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
4618 }
```

*(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.

```
4619 \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.

```

4620 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
4621   \@nameuse{#1}{#2}
4622 }{
4623   \cs_if_exist:NTF\rdfmata@sectioning{
4624     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
4625   }{
4626     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
4627   }
4628 }
4629 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
4630 }

```

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

omgroup

```

4631 \keys_define:nn { document-structure / omgroup }{
4632   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
4633   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
4634   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
4635   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
4636   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
4637   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
4638   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
4639   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
4640   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
4641   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
4642 }
4643 \cs_new_protected:Nn \l__document_structure_omgroup_args:n {
4644   \str_clear:N \l__document_structure_omgroup_id_str
4645   \str_clear:N \l__document_structure_omgroup_date_str
4646   \clist_clear:N \l__document_structure_omgroup_creators_clist
4647   \clist_clear:N \l__document_structure_omgroup_contributors_clist
4648   \tl_clear:N \l__document_structure_omgroup_srccite_tl
4649   \tl_clear:N \l__document_structure_omgroup_type_tl
4650   \tl_clear:N \l__document_structure_omgroup_short_tl
4651   \tl_clear:N \l__document_structure_omgroup_display_tl
4652   \tl_clear:N \l__document_structure_omgroup_intro_tl
4653   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
4654   \keys_set:nn { document-structure / omgroup } { #1 }
4655 }

```

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.

```

4656 \newif\if@mainmatter\@mainmattertrue
4657 \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.

```

4658 \keys_define:nn { document-structure / sectioning }{
4659   name .str_set_x:N = \l__document_structure_sect_name_str ,
4660   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
4661   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
4662   num .bool_set:N = \l__document_structure_sect_num_bool ,
4663 }

```

```

4664 \cs_new_protected:Nn \__document_structure_sect_args:n {
4665   \str_clear:N \l__document_structure_sect_name_str
4666   \str_clear:N \l__document_structure_sect_ref_str
4667   \bool_set_false:N \l__document_structure_sect_clear_bool
4668   \bool_set_false:N \l__document_structure_sect_num_bool
4669   \keys_set:nn { document-structure / sectioning } { #1 }
4670 }
4671 \newcommand\omdoc@sectioning[3][]{
4672   \__document_structure_sect_args:n {#1 }
4673   \let\omdoc@sect@name\l__document_structure_sect_name_str
4674   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
4675   \if@mainmatter% numbering not overridden by frontmatter, etc.
4676     \bool_if:NTF \l__document_structure_sect_num_bool {
4677       \omgroup@num{#2}{#3}
4678     }{
4679       \omgroup@nonum{#2}{#3}
4680     }
4681     \def\current@section@level{\omdoc@sect@name}
4682   \else
4683     \omgroup@nonum{#2}{#3}
4684   \fi
4685 }% 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.

```

4686 \newcommand\omgroup@redefine@addtocontents[1]{%
4687   %\edef\__document_structureimport{#1}%
4688   %\@for\@I:=\__document_structureimport\do{%
4689     \edef\@path{\csname module@\@I @path\endcsname}%
4690     \@ifundefined{tf@toc}\relax%
4691     % {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
4692   %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
4693   %\def\addcontentsline##1##2##3{%
4694     \addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
4695   %\else% hyperref.sty not loaded
4696   %\def\addcontentsline##1##2##3{%
4697     \addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
4698   %\fi
4699 }% 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.

```

4700 \int_new:N \l_document_structure_omgroup_level_int
4701 \newenvironment{omgroup}[2][]{% keys, title
4702 {
4703   \__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.

```

4704 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
4705   \omgroup@redefine@addtocontents{
4706     %\@ifundefined{module@id}\used@modules%
4707     %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}

```

```

4708     }
4709 }

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

4710 \int_incr:N \l_document_structure_omgroup_level_int
4711 \int_incr:N \l_document_structure_section_level_int
4712 \ifcase\l_document_structure_section_level_int
4713   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
4714   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
4715   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
4716   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
4717   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
4718   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
4719   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
4720 \fi
4721 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
4722 \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str
4723 }% for customization
4724 {}

```

and finally, we localize the sections

```

4725 \newcommand\omdoc@part@kw{Part}
4726 \newcommand\omdoc@chapter@kw{Chapter}
4727 \newcommand\omdoc@section@kw{Section}
4728 \newcommand\omdoc@subsection@kw{Subsection}
4729 \newcommand\omdoc@subsubsection@kw{Subsubsection}
4730 \newcommand\omdoc@paragraph@kw{paragraph}
4731 \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`

```

4732 \providecommand\printindex{\IfFileExists{\jobname.ind}{\input{\jobname.ind}}{}}

```

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

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

```

4733 \cs_if_exist:NTF\frontmatter{
4734   \let\__document_structure_orig_frontmatter\frontmatter
4735   \let\frontmatter\relax
4736 }{
4737   \tl_set:Nn\__document_structure_orig_frontmatter{
4738     \clearpage
4739     \@mainmatterfalse
4740     \pagenumbering{roman}
4741   }
4742 }
4743 \cs_if_exist:NTF\backmatter{

```



```

4744 \let\__document_structure_orig_backmatter\backmatter
4745 \let\backmatter\relax
4746 }{
4747 \tl_set:Nn\__document_structure_orig_backmatter{
4748 \clearpage
4749 \@mainmatterfalse
4750 \pagenumbering{roman}
4751 }
4752 }

```

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.

```

4753 \newenvironment{frontmatter}{
4754 \__document_structure_orig_frontmatter
4755 }{
4756 \cs_if_exist:NTF\mainmatter{
4757 \mainmatter
4758 }{
4759 \clearpage
4760 \@mainmattertrue
4761 \pagenumbering{arabic}
4762 }
4763 }

```

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

```

4764 \newenvironment{backmatter}{
4765 \__document_structure_orig_backmatter
4766 }{
4767 \cs_if_exist:NTF\mainmatter{
4768 \mainmatter
4769 }{
4770 \clearpage
4771 \@mainmattertrue
4772 \pagenumbering{arabic}
4773 }
4774 }

```

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

```

4775 \@mainmattertrue\pagenumbering{arabic}

```

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

```

4776 \def \c__document_structure_document_str{document}
4777 \newcommand\afterprematurestop{}
4778 \def\prematurestop@endomgroup{
4779 \unless\ifx\@currenvir\c__document_structure_document_str
4780 \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
4781 \expandafter\prematurestop@endomgroup
4782 \fi
4783 }
4784 \providecommand\prematurestop{

```

```

4785 \message{Stopping~sTeX~processing~prematurely}
4786 \prematuarestop@endomgroup
4787 \afterprematuarestop
4788 \end{document}
4789 }

```

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

## 38.8 Global Variables

**\setSGvar** set a global variable

```

4790 \RequirePackage{etoolbox}
4791 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

**\useSGvar** use a global variable

```

4792 \newrobustcmd\useSGvar[1]{%
4793   \@ifundefined{sTeX@Gvar@#1}
4794   {\PackageError{omdoc}
4795     {The sTeX Global variable #1 is undefined}
4796     {set it with \protect\setSGvar}}
4797   \@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.

```

4798 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
4799   \@ifundefined{sTeX@Gvar@#1}
4800   {\PackageError{omdoc}
4801     {The sTeX Global variable #1 is undefined}
4802     {set it with \protect\setSGvar}}
4803   {\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.

```
4804 \*cls)
4805 \@@=mikoslides)
4806 \ProvidesExplClass{mikoslides}{2020/12/06}{1.3}{MiKo slides Class}
4807 \RequirePackage{l3keys2e,expl-keystr-compatible}
4808
4809 \keys_define:nn{mikoslides / cls}{
4810   class .code:n = {
4811     \PassOptionsToClass{\CurrentOption}{omdoc}
4812     \str_if_eq:nnT{#1}{book}{
4813       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
4814     }
4815     \str_if_eq:nnT{#1}{report}{
4816       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
4817     }
4818   },
4819   notes .bool_set:N = \c__mikoslides_notes_bool ,
4820   slides .code:n = { \bool_set_false:N \c__mikoslides_notes_bool },
4821   unknown .code:n = {
4822     \PassOptionsToClass{\CurrentOption}{omdoc}
4823     \PassOptionsToClass{\CurrentOption}{beamer}
4824     \PassOptionsToPackage{\CurrentOption}{mikoslides}
4825   }
4826 }
4827 \ProcessKeysOptions{ mikoslides / cls }
4828 \bool_if:NTF \c__mikoslides_notes_bool {
4829   \PassOptionsToPackage{notes=true}{mikoslides}
4830 }{
4831   \PassOptionsToPackage{notes=false}{mikoslides}
4832 }
4833 \</cls)
```

now we do the same for the mikoslides package.

```

4834 \*package>
4835 \ProvidesExplPackage{mikoslides}{2020/12/06}{1.3}{MiKo slides Package}
4836 \RequirePackage{l3keys2e,expl-keystr-compat}
4837
4838 \keys_define:nn{mikoslides / pkg}{
4839   topsect      .str_set_x:N = \c__mikoslides_topsect_str,
4840   defaulttopsect .str_set_x:N = \c__mikoslides_defaulttopsec_str,
4841   notes        .bool_set:N = \c__mikoslides_notes_bool ,
4842   slides       .code:n      = { \bool_set_false:N \c__mikoslides_notes_bool },
4843   sectocframes .bool_set:N = \c__mikoslides_sectocframes_bool ,
4844   frameimages  .bool_set:N = \c__mikoslides_frameimages_bool ,
4845   fiboxed      .bool_set:N = \c__mikoslides_fiboxed_bool ,
4846   nopproblems  .bool_set:N = \c__mikoslides_nopproblems_bool,
4847   unknown      .code:n      = {
4848     \PassOptionsToClass{\CurrentOption}{stex}
4849     \PassOptionsToClass{\CurrentOption}{tikzinput}
4850   }
4851 }
4852 \ProcessKeysOptions{ mikoslides / pkg }
4853 \newif\ifnotes
4854 \bool_if:NTF \c__mikoslides_notes_bool {
4855   \notesttrue
4856 }{
4857   \notesfalse
4858 }
4859

```

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

```

4860 \str_if_empty:NTF \c__mikoslides_topsect_str {
4861   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_defaulttopsec_str
4862 }{
4863   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_topsect_str
4864 }
4865 \</package>

```

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

```

4866 \*cls>
4867 \bool_if:NTF \c__mikoslides_notes_bool {
4868   \LoadClass{omdoc}
4869 }{
4870   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
4871   \newcounter{Item}
4872   \newcounter{paragraph}
4873   \newcounter{subparagraph}
4874   \newcounter{Hfootnote}
4875   \RequirePackage{omdoc}
4876 }

```

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

```

4877 \RequirePackage{mikoslides}
4878 \</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.

```

4879 \*package>
4880 \bool_if:NT \c__mikoslides_notes_bool {
4881   \RequirePackage{a4wide}
4882   \RequirePackage{marginnote}
4883   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
4884   \RequirePackage{mdframed}
4885   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
4886   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
4887 }
4888 \RequirePackage{stex-compatibility}
4889 \RequirePackage{stex-tikzinput}
4890 \RequirePackage{etoolbox}
4891 \RequirePackage{amssymb}
4892 \RequirePackage{amsmath}
4893 \RequirePackage{comment}
4894 \RequirePackage{textcomp}
4895 \RequirePackage{url}
4896 \RequirePackage{graphicx}
4897 \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>

```

4898 \bool_if:NT \c__mikoslides_notes_bool {
4899   \renewcommand\usetheme[2] [] {\usepackage[#1]{beamernotestheme#2}}
4900 }

```

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

```

4901 \newcounter{slide}
4902 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
4903 \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.

```

4904 \bool_if:NTF \c__mikoslides_notes_bool {
4905   \renewenvironment{note}{\ignorespaces}{\ignorespaces}{}
4906 }{
4907   \excludecomment{note}
4908 }

```

---

<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.

```
4909 \bool_if:NT \c__mikoslides_notes_bool {
4910   \newlength{\slideframewidth}
4911   \setlength{\slideframewidth}{1.5pt}
```

**frame** We first define the keys.

```
4912 \cs_new_protected:Nn \__mikoslides_do_yes_param:Nn {
4913   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
4914     \bool_set_true:N #1
4915   }{
4916     \bool_set_false:N #1
4917   }
4918 }
4919 \keys_define:nn{mikoslides / frame}{
4920   label .str_set_x:N = \l__mikoslides_frame_label_str,
4921   allowframebreaks .code:n = {
4922     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowframebreaks_bool { #1 }
4923   },
4924   allowdisplaybreaks .code:n = {
4925     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowdisplaybreaks_bool { #1 }
4926   },
4927   fragile .code:n = {
4928     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_fragile_bool { #1 }
4929   },
4930   shrink .code:n = {
4931     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_shrink_bool { #1 }
4932   },
4933   squeeze .code:n = {
4934     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_squeeze_bool { #1 }
4935   },
4936   t .code:n = {
4937     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_t_bool { #1 }
4938   },
4939 }
4940 \cs_new_protected:Nn \__mikoslides_frame_args:n {
4941   \str_clear:N \l__mikoslides_frame_label_str
4942   \bool_set_true:N \l__mikoslides_frame_allowframebreaks_bool
4943   \bool_set_true:N \l__mikoslides_frame_allowdisplaybreaks_bool
4944   \bool_set_true:N \l__mikoslides_frame_fragile_bool
4945   \bool_set_true:N \l__mikoslides_frame_shrink_bool
4946   \bool_set_true:N \l__mikoslides_frame_squeeze_bool
4947   \bool_set_true:N \l__mikoslides_frame_t_bool
4948   \keys_set:nn { mikoslides / frame }{ #1 }
4949 }
```

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

```
4950 \renewenvironment{frame}[1][]{
4951   \__mikoslides_frame_args:n{#1}
4952   \sffamily
4953   \stepcounter{slide}
4954   \def\@currentlabel{\theslide}
4955   \str_if_empty:NF \l__mikoslides_frame_label_str {
4956     \label{\l__mikoslides_frame_label_str}
```

```
4957 }
```

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

```
4958 \def\itemize@level{outer}
4959 \def\itemize@outer{outer}
4960 \def\itemize@inner{inner}
4961 \renewcommand\newpage{\addtocounter{framenum}{1}}
4962 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
4963 \renewenvironment{itemize}{
4964   \ifx\itemize@level\itemize@outer
4965     \def\itemize@label{$\rhd$}
4966   \fi
4967   \ifx\itemize@level\itemize@inner
4968     \def\itemize@label{$\scriptstyle\rhd$}
4969   \fi
4970   \begin{list}
4971     {\itemize@label}
4972     {\setlength{\labelsep}{.3em}
4973      \setlength{\labelwidth}{.5em}
4974      \setlength{\leftmargin}{1.5em}
4975     }
4976   \edef\itemize@level{\itemize@inner}
4977 }{
4978   \end{list}
4979 }
```

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

```
4980 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth]
4981 }{
4982   \medskip\miko@slidelabel\end{mdframed}
4983 }
```

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

`\frametitle`

```
4984 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
4985 }
```

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

EdN:21

`\pause` 21

```
4986 \bool_if:NT \c__mikoslides_notes_bool {
4987   \newcommand\pause{}
4988 }
```

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

`nomtext`

```
4989 \bool_if:NTF \c__mikoslides_notes_bool {
4990   \newenvironment{nomtext}[1][\begin{sparagraph}[#1]}{\end{sparagraph}}
4991 }{
4992   \excludecomment{nomtext}
4993 }
```

---

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

```

nomgroup
4994 \bool_if:NTF \c__mikoslides_notes_bool {
4995   \newenvironment{nomgroup}[2] [] {\begin{omgroup}[#1]{#2}}{\end{omgroup}}
4996 }{
4997   \excludecomment{nomgroup}
4998 }

ndefinition
4999 \bool_if:NTF \c__mikoslides_notes_bool {
5000   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}{\end{sdefinition}}
5001 }{
5002   \excludecomment{ndefinition}
5003 }

nassertion
5004 \bool_if:NTF \c__mikoslides_notes_bool {
5005   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}{\end{sassertion}}
5006 }{
5007   \excludecomment{nassertion}
5008 }

nsproof
5009 \bool_if:NTF \c__mikoslides_notes_bool {
5010   \newenvironment{nproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
5011 }{
5012   \excludecomment{nproof}
5013 }

nexample
5014 \bool_if:NTF \c__mikoslides_notes_bool {
5015   \newenvironment{nexample}[1] [] {\begin{example}[#1]}{\end{example}}
5016 }{
5017   \excludecomment{nexample}
5018 }

\inputref@*skip We customize the hooks for in \inputref.
5019 \def\inputref@preskip{\smallskip}
5020 \def\inputref@postskip{\medskip}

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

\inputref*
5021 \let\orig@inputref\inputref
5022 \def\inputref{\@ifstar\ninputref\orig@inputref}
5023 \newcommand\ninputref[2] [] {
5024   \bool_if:NT \c__mikoslides_notes_bool {
5025     \orig@inputref[#1]{#2}
5026   }
5027 }

(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>}`.

```

5028 \newlength{\slidelogoheight}
5029
5030 \bool_if:NTF \c__mikoslides_notes_bool {
5031   \setlength{\slidelogoheight}{.4cm}
5032 }{
5033   \setlength{\slidelogoheight}{1cm}
5034 }
5035 \newsavebox{\slidelogo}
5036 \sbox{\slidelogo}{\text{\TeX}}
5037 \newrobustcmd{\setslidelogo}[1]{
5038   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
5039 }
```

(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.

```

5040 \def\source{Michael Kohlhase}% customize locally
5041 \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.

```

5042 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
5043 \newsavebox{\cclogo}
5044 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
5045 \newif\ifcchref\cchreffalse
5046 \AtBeginDocument{
5047   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
5048 }
5049 \def\licensing{
5050   \ifcchref
5051     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
5052   \else
5053     {\usebox{\cclogo}}
5054   \fi
5055 }
5056 \newrobustcmd{\setlicensing}[2][]{
5057   \def@url{#1}
5058   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
5059   \ifx@url@empty
5060     \def\licensing{{\usebox{\cclogo}}}
5061   \else
5062     \def\licensing{
```

```

5063     \ifcchref
5064     \href{#1}{\usebox{\cclogo}}
5065     \else
5066     {\usebox{\cclogo}}
5067     \fi
5068   }
5069   \fi
5070 }

```

(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>

```

5071 \newrobustcmd\miko@slidelabel{
5072   \vbox to \slidelogoheight{
5073     \vss\hbox to \slidewidth
5074     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
5075   }
5076 }

```

(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.

```

5077 \def\Gin@mhrepos{}
5078 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
5079 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
5080 \newrobustcmd\frameimage[2][{}{
5081   \stepcounter{slide}
5082   \bool_if:NT \c__mikoslides_frameimages_bool {
5083     \def\Gin@ewidth{}\setkeys{Gin}{#1}
5084     \bool_if:NF \c__mikoslides_notes_bool { \vfill }
5085     \begin{center}
5086       \bool_if:NTF \c__mikoslides_fiboxed_bool {
5087         \fbox{
5088           \ifx\Gin@ewidth\@empty
5089             \ifx\Gin@mhrepos\@empty
5090               \mhgraphics[width=\slidewidth,#1]{#2}
5091             \else
5092               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5093             \fi
5094           \else% Gin@ewidth empty
5095             \ifx\Gin@mhrepos\@empty
5096               \mhgraphics[#1]{#2}
5097             \else
5098               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5099             \fi
5100           \fi% Gin@ewidth empty
5101         }
5102       }{
5103         \ifx\Gin@ewidth\@empty

```

---

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

```

5104         \ifx\Gin@mhrepos\empty
5105             \mhgraphics[width=\slidewidth,#1]{#2}
5106         \else
5107             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5108         \fi
5109         \ifx\Gin@mhrepos\empty
5110             \mhgraphics[#1]{#2}
5111         \else
5112             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5113         \fi
5114     \fi% Gin@ewidth empty
5115 }
5116 \end{center}
5117 \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
5118 \bool_if:NF \c__mikoslides_notes_bool { \vfill }
5119 }
5120 } % 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.

```

5121 \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

```

5122 \AddToHook{begindocument}{
5123     \definecolor{green}{rgb}{0,.5,0}
5124     \definecolor{purple}{cmyk}{.3,1,0,.17}
5125 }

```

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`.

```

5126 % \def\STpresent#1{\textcolor{blue}{#1}}
5127 \def\defemph#1{\textcolor{magenta}{#1}}
5128 \def\symrefemph#1{\textcolor{cyan}{#1}}
5129 \def\compemph#1{\textcolor{blue}{#1}}
5130 \def\titleemph#1{\textcolor{blue}{#1}}
5131 \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.

```

5132 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
5133 \def\smalltextwarning{
5134     \pgfuseimage{miko@small@dbend}
5135     \xspace
5136 }
5137 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}

```

```

5138 \newrobustcmd\textwarning{
5139   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
5140   \xspace
5141 }
5142 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}
5143 \newrobustcmd\bigtextwarning{
5144   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
5145   \xspace
5146 }

```

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

```

5147 \newrobustcmd\putgraphicsat[3]{
5148   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
5149 }
5150 \newrobustcmd\putat[2]{
5151   \begin{picture}(0,0)\put(#1){#2}\end{picture}
5152 }

```

## 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`.

```

5153 \bool_if:NT \c__mikoslides_sectocframes_bool {
5154   \str_if_eq:VnTF \__mikoslidestopsect{part}{
5155     \newcounter{chapter}\counterwithin*{section}{chapter}
5156   }{
5157     \str_if_eq:VnT\__mikoslidestopsect{chapter}{
5158       \newcounter{chapter}\counterwithin*{section}{chapter}
5159     }
5160   }
5161 }

```

`\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
5162 \def\part@prefix{}
5163 \@ifpackageloaded{omdoc}{}{
5164   \str_case:VnF \__mikoslidestopsect {
5165     {part}{
5166       \int_set:Nn \l_document_structure_section_level_int {0}
5167       \def\thesection{\arabic{chapter}.\arabic{section}}
5168       \def\part@prefix{\arabic{chapter}.}
5169     }
5170     {chapter}{
5171       \int_set:Nn \l_document_structure_section_level_int {1}
5172       \def\thesection{\arabic{chapter}.\arabic{section}}
5173       \def\part@prefix{\arabic{chapter}.}
5174     }
5175   }{
5176     \int_set:Nn \l_document_structure_section_level_int {2}
5177     \def\part@prefix{}

```

```

5178 }
5179 }
5180
5181 \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

```

5182 \renewenvironment{omgroup}[2][]{
5183   \__document_structure_omgroup_args:n { #1 }
5184   \int_incr:N \l_document_structure_omgroup_level_int
5185   \int_incr:N \l_document_structure_section_level_int
5186   \bool_if:NT \c__mikoslides_sectocframes_bool {
5187     \stepcounter{slide}
5188     \begin{frame}[noframenumbering]
5189     \vfill\Large\centering
5190     \red{
5191       \ifcase\l_document_structure_section_level_int\or
5192         \stepcounter{part}
5193         \def\__mikoslideslabel{\omdoc@part@kw~\Roman{part}}
5194         \def\currentsectionlevel{\omdoc@part@kw}
5195       \or
5196         \stepcounter{chapter}
5197         \def\__mikoslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
5198         \def\currentsectionlevel{\omdoc@chapter@kw}
5199       \or
5200         \stepcounter{section}
5201         \def\__mikoslideslabel{\part@prefix\arabic{section}}
5202         \def\currentsectionlevel{\omdoc@section@kw}
5203       \or
5204         \stepcounter{subsection}
5205         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
5206         \def\currentsectionlevel{\omdoc@subsection@kw}
5207       \or
5208         \stepcounter{subsubsection}
5209         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
5210         \def\currentsectionlevel{\omdoc@subsubsection@kw}
5211       \or
5212         \stepcounter{paragraph}
5213         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{paragraph}}
5214         \def\currentsectionlevel{\omdoc@paragraph@kw}
5215       \else
5216         \def\__mikoslideslabel{}
5217         \def\currentsectionlevel{\omdoc@paragraph@kw}
5218       \fi% end ifcase
5219       \__mikoslideslabel%\sref@label@id\__mikoslideslabel
5220       \quad #2%
5221     }%
5222     \vfill%
5223     \end{frame}%
5224   }
5225   \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str%

```

```

5226 }{}
5227 }

```

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

```

5228 \def\inserttheorembodyfont{\normalfont}
5229 %\bool_if:NF \c__mikoslides_notes_bool {
5230 % \defbeamertemplate{theorem begin}{miko}
5231 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
5232 % \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
5233 % \inserttheorempunctuation\inserttheorembodyfont\hspace}
5234 % \defbeamertemplate{theorem end}{miko}{\}

```

and we set it as the default one.

```

5235 % \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.

```

5236 % \expandafter\def\csname Parent2\endcsname{}
5237 %}
5238
5239 \AddToHook{begindocument}{% this does not work for some reasons
5240 \setbeamertemplate{theorems}[ams style]
5241 }
5242 \bool_if:NT \c__mikoslides_notes_bool {
5243 \renewenvironment{columns}[1][\%
5244 \par\noindent%
5245 \begin{minipage}%
5246 \slidewidth\centering\leavevmode%
5247 }{\%
5248 \end{minipage}\par\noindent%
5249 }%
5250 \newsavebox\columnbox%
5251 \renewenvironment<>{column}[2][\%
5252 \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
5253 }{\%
5254 \end{minipage}\end{lrbox}\usebox\columnbox%
5255 }%
5256 }
5257 \bool_if:NTF \c__mikoslides_noproblems_bool {
5258 \newenvironment{problems}{}{}
5259 }{
5260 \excludecomment{problems}
5261 }

```

## 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.

```

5262 \gdef\printexcursions{}
5263 \newcommand\excursionref[2]{% label, text
5264 \bool_if:NT \c__mikoslides_notes_bool {

```

```

5265 \begin{sparagraph}[title=Excursion]
5266 #2 \sref[fallback=the appendix]{#1}.
5267 \end{sparagraph}
5268 }
5269 }
5270 \newcommand\activate@excursion[2][]{
5271 \gappto\printexcursions{\inputref[#1]{#2}}
5272 }
5273 \newcommand\excursion[4][]{% repos, label, path, text
5274 \bool_if:NT \c__mikoslides_notes_bool {
5275 \activate@excursion[#1]{#3}\excursionref{#2}{#4}
5276 }
5277 }

```

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

\excursiongroup

```

5278 \keys_define:nn{mikoslides / excursiongroup }{
5279 id .str_set_x:N = \l__mikoslides_excursion_id_str,
5280 intro .tl_set:N = \l__mikoslides_excursion_intro_tl,
5281 mhrepos .str_set_x:N = \l__mikoslides_excursion_mhrepos_str
5282 }
5283 \cs_new_protected:Nn \__mikoslides_excursion_args:n {
5284 \tl_clear:N \l__mikoslides_excursion_intro_tl
5285 \str_clear:N \l__mikoslides_excursion_id_str
5286 \str_clear:N \l__mikoslides_excursion_mhrepos_str
5287 \keys_set:nn {mikoslides / excursiongroup }{ #1 }
5288 }
5289 \newcommand\excursiongroup[1][]{
5290 \__mikoslides_excursion_args:n{ #1 }
5291 \ifdefempty\printexcursions{}% only if there are excursions
5292 {\begin{note}
5293 \begin{omgroup}[#1]{Excursions}%
5294 \ifdefempty\l__mikoslides_excursion_intro_tl{{
5295 \inputref[\l__mikoslides_excursion_mhrepos_str]{
5296 \l__mikoslides_excursion_intro_tl
5297 }
5298 }
5299 \printexcursions%
5300 \end{omgroup}
5301 \end{note}}
5302 }
5303 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
5304 \</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.

```
5305 <*package>
5306 <@@=problems>
5307 \ProvidesExplPackage{problem}{2019/03/20}{1.3}{Semantic Markup for Problems}
5308 \RequirePackage{l3keys2e,expl-keystr-compatible}
5309
5310 \keys_define:nn { problem / pkg }{
5311   notes      .default:n    = { true },
5312   notes      .bool_set:N   = \c__problems_notes_bool,
5313   gnotes     .default:n    = { true },
5314   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
5315   hints      .default:n    = { true },
5316   hints      .bool_set:N   = \c__problems_hints_bool,
5317   solutions  .default:n    = { true },
5318   solutions  .bool_set:N   = \c__problems_solutions_bool,
5319   pts        .default:n    = { true },
5320   pts        .bool_set:N   = \c__problems_pts_bool,
5321   min        .default:n    = { true },
5322   min        .bool_set:N   = \c__problems_min_bool,
5323   boxed      .default:n    = { true },
5324   boxed      .bool_set:N   = \c__problems_boxed_bool,
5325   unknown    .code:n       = {}
5326 }
5327 \def\solutionstrue{
5328   \bool_set_true:N \c__problems_solutions_bool
5329 }
5330 \def\solutionsfalse{
5331   \bool_set_false:N \c__problems_solutions_bool
5332 }
5333
5334 \ProcessKeysOptions{ problem / pkg }
```

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



```

5335 \RequirePackage{stex-compatibility}
5336 \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.

```

5337 \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.

```

5338 \def\prob@problem@kw{Problem}
5339 \def\prob@solution@kw{Solution}
5340 \def\prob@hint@kw{Hint}
5341 \def\prob@note@kw{Note}
5342 \def\prob@gnote@kw{Grading}
5343 \def\prob@pt@kw{pt}
5344 \def\prob@min@kw{min}

```

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

For the other languages, we set up triggers

```

5345 \@ifpackageloaded{babel}{
5346   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5347   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5348     \input{problem-ngerman.ldf}
5349   }
5350   \clist_if_in:NnT \l_tmpa_clist {finnish}{
5351     \input{problem-finnish.ldf}
5352   }
5353   \clist_if_in:NnT \l_tmpa_clist {french}{
5354     \input{problem-french.ldf}
5355   }
5356   \clist_if_in:NnT \l_tmpa_clist {russian}{
5357     \input{problem-russian.ldf}
5358   }
5359 }{}

```

## 40.2 Problems and Solutions

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

```

5360 \keys_define:nn{ problem / problem }{
5361   id      .str_set:x:N = \l__problems_prob_id_str,
5362   pts     .tl_set:N    = \l__problems_prob_pts_tl,
5363   min     .tl_set:N    = \l__problems_prob_min_tl,
5364   title   .tl_set:N    = \l__problems_prob_title_tl,
5365   refnum  .int_set:N   = \l__problems_prob_refnum_int
5366 }
5367 \cs_new_protected:Nn \__problems_prob_args:n {
5368   \str_clear:N \l__problems_prob_id_str
5369   \tl_clear:N \l__problems_prob_pts_tl
5370   \tl_clear:N \l__problems_prob_min_tl
5371   \tl_clear:N \l__problems_prob_title_tl

```

```

5372 \int_zero_new:N \l__problems_prob_refnum_int
5373 \keys_set:nn { problem / problem }{ #1 }
5374 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
5375   \let\l__problems_inclprob_refnum_int\undefined
5376 }
5377 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

5378 \newcounter{problem}
5379 \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.

```

5380 \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

```

5381 \newcommand\prob@number{
5382   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
5383     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
5384   }{
5385     \int_if_exist:NTF \l__problems_prob_refnum_int {
5386       \prob@label{\int_use:N \l__problems_prob_refnum_int }
5387     }{
5388       \prob@label\theproblem
5389     }
5390   }
5391 }

```

(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.

```

5392 \newcommand\prob@title[3]{%
5393   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
5394     #2 \l__problems_inclprob_title_tl #3
5395   }{
5396     \tl_if_exist:NTF \l__problems_prob_title_tl {
5397       #2 \l__problems_prob_title_tl #3
5398     }{
5399       #1
5400     }
5401   }
5402 }

```

(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.

```

5403 \def\prob@heading{
5404   \prob@problem@kw~\prob@number\prob@title{~}{~}{~}\strut}
5405   \%sref@label@id{\prob@problem@kw~\prob@number}{~}
5406 }

```

(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`

```

5407 \newenvironment{problem}[1][ ]{
5408   \__problems_prob_args:n{#1}%\sref@target%
5409   \@in@omtexttrue% we are in a statement (for inline definitions)
5410   \stepcounter{problem}\record@problem
5411   \def\current@section@level{\prob@problem@kw}
5412   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
5413 }%
5414 {\smallskip}
5415 \bool_if:NT \c__problems_boxed_bool {
5416   \surroundwithmdframed{problem}
5417 }

```

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

```

5418 \def\record@problem{
5419   \protected@write\@auxout{}
5420   {
5421     \string\@problem{\prob@number}
5422     {
5423       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
5424         \l__problems_inclprob_pts_tl
5425       }{
5426         \l__problems_prob_pts_tl
5427       }
5428     }%
5429     {
5430       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
5431         \l__problems_inclprob_min_tl
5432       }{
5433         \l__problems_prob_min_tl
5434       }
5435     }
5436   }
5437 }

```

(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).

```

5438 \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.

```

5439 \keys_define:nn { problem / solution }{
5440   id          .str_set_x:N = \l__problems_solution_id_str ,
5441   for         .tl_set:N    = \l__problems_solution_for_tl ,
5442   height      .dim_set:N   = \l__problems_solution_height_dim ,
5443   creators    .clist_set:N = \l__problems_solution_creators_clist ,
5444   contributors .clist_set:N = \l__problems_solution_contributors_clist ,
5445   srccite     .tl_set:N    = \l__problems_solution_srccite_tl
5446 }
5447 \cs_new_protected:Nn \__problems_solution_args:n {
5448   \str_clear:N \l__problems_solution_id_str
5449   \tl_clear:N \l__problems_solution_for_tl
5450   \tl_clear:N \l__problems_solution_srccite_tl
5451   \clist_clear:N \l__problems_solution_creators_clist
5452   \clist_clear:N \l__problems_solution_contributors_clist
5453   \dim_zero:N \l__problems_solution_height_dim
5454   \keys_set:nn { problem / solution }{ #1 }
5455 }

```

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

```

5456 \newcommand\@startsolution[1][{}]{
5457   \__problems_solution_args:n { #1 }
5458   \@in@omtexttrue% we are in a statement.
5459   \bool_if:NF \c__problems_boxed_bool { \hrule }
5460   \smallskip\noindent
5461   {\textbf\prob@solution@kw : \enspace}
5462   \begin{small}
5463   \def\current@section@level{\prob@solution@kw}
5464   \ignorespacesandpars
5465 }

```

**\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.

```

5466 \newcommand\startsolutions{
5467   \specialcomment{solution}{\@startsolution}{
5468     \bool_if:NF \c__problems_boxed_bool {
5469       \hrule\medskip
5470     }
5471     \end{small}%
5472   }
5473   \bool_if:NT \c__problems_boxed_bool {
5474     \surroundwithmdframed{solution}
5475   }
5476 }

```

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

**\stopsolutions**

```

5477 \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.

```

5478 \bool_if:NTF \c_problems_solutions_bool {
5479   \startsolutions
5480 }{
5481   \stopsolutions
5482 }

```

**exnote**

```

5483 \bool_if:NTF \c_problems_notes_bool {
5484   \newenvironment{exnote}[1][]{
5485     \par\smallskip\hrule\smallskip
5486     \noindent\textbf{\prob@note@kw : }\small
5487   }{
5488     \smallskip\hrule
5489   }
5490 }{
5491   \excludecomment{exnote}
5492 }

```

**hint**

```

5493 \bool_if:NTF \c_problems_notes_bool {
5494   \newenvironment{hint}[1][]{
5495     \par\smallskip\hrule\smallskip
5496     \noindent\textbf{\prob@hint@kw :~ }\small
5497   }{
5498     \smallskip\hrule
5499   }
5500   \newenvironment{exhint}[1][]{
5501     \par\smallskip\hrule\smallskip
5502     \noindent\textbf{\prob@hint@kw :~ }\small
5503   }{
5504     \smallskip\hrule
5505   }
5506 }{
5507   \excludecomment{hint}
5508   \excludecomment{exhint}
5509 }

```

**gnote**

```

5510 \bool_if:NTF \c_problems_notes_bool {
5511   \newenvironment{gnote}[1][]{
5512     \par\smallskip\hrule\smallskip
5513     \noindent\textbf{\prob@gnote@kw : }\small
5514   }{
5515     \smallskip\hrule
5516   }
5517 }{
5518   \excludecomment{gnote}
5519 }

```

## 40.3 Multiple Choice Blocks

EdN:23

mcb 23

```
5520 \newenvironment{mcb}{
5521   \begin{enumerate}
5522 }{
5523   \end{enumerate}
5524 }
```

we define the keys for the mcc macro

```
5525 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
5526   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
5527     \bool_set_true:N #1
5528   }{
5529     \bool_set_false:N #1
5530   }
5531 }
5532 \keys_define:nn { problem / mcc }{
5533   id          .str_set:x:N = \l__problems_mcc_id_str ,
5534   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
5535   T           .default:n    = { true } ,
5536   T           .bool_set:N   = \l__problems_mcc_t_bool ,
5537   F           .default:n    = { true } ,
5538   F           .bool_set:N   = \l__problems_mcc_f_bool ,
5539   Ttext       .code:n       = {
5540     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
5541   } ,
5542   Ftext       .code:n       = {
5543     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
5544   }
5545 }
5546 \cs_new_protected:Nn \l__problems_mcc_args:n {
5547   \str_clear:N \l__problems_mcc_id_str
5548   \tl_clear:N \l__problems_mcc_feedback_tl
5549   \bool_set_true:N \l__problems_mcc_t_bool
5550   \bool_set_true:N \l__problems_mcc_f_bool
5551   \bool_set_true:N \l__problems_mcc_Ttext_bool
5552   \bool_set_false:N \l__problems_mcc_Ftext_bool
5553   \keys_set:nn { problem / mcc }{ #1 }
5554 }
```

\mcc

```
5555 \newcommand\mcc[2][] {
5556   \l__problems_mcc_args:n{ #1 }
5557   \item #2
5558   \bool_if:NT \c__problems_solutions_bool {
5559     \
5560     \bool_if:NT \l__problems_mcc_t_bool {
5561       % TODO!
5562       % \ifcsstring{mcc@T}{T}{ }\{mcc@Ttext}%
5563     }
5564     \bool_if:NT \l__problems_mcc_f_bool {
```

---

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

```

5565         % TODO!
5566         % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
5567     }
5568     \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
5569         !
5570     }{
5571         \l__problems_mcc_feedback_tl
5572     }
5573 }
5574 } %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.

```

5575
5576 \keys_define:nn{ problem / inclproblem }{
5577   % id      .str_set_x:N = \l__problems_inclprob_id_str,
5578   pts      .tl_set:N    = \l__problems_inclprob_pts_tl,
5579   min      .tl_set:N    = \l__problems_inclprob_min_tl,
5580   title    .tl_set:N    = \l__problems_inclprob_title_tl,
5581   refnum   .int_set:N    = \l__problems_inclprob_refnum_int,
5582   mhrepos  .str_set_x:N = \l__problems_inclprob_mhrepos_str
5583 }
5584 \cs_new_protected:Nn \l__problems_inclprob_args:n {
5585   % \str_clear:N \l__problems_prob_id_str
5586   \tl_clear:N \l__problems_inclprob_pts_tl
5587   \tl_clear:N \l__problems_inclprob_min_tl
5588   \tl_clear:N \l__problems_inclprob_title_tl
5589   \int_zero_new:N \l__problems_inclprob_refnum_int
5590   \str_clear:N \l__problems_inclprob_mhrepos_str
5591   \keys_set:nn { problem / inclproblem }{ #1 }
5592   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
5593     \let\l__problems_inclprob_pts_tl\undefined
5594   }
5595   \tl_if_empty:NT \l__problems_inclprob_min_tl {
5596     \let\l__problems_inclprob_min_tl\undefined
5597   }
5598   \tl_if_empty:NT \l__problems_inclprob_title_tl {
5599     \let\l__problems_inclprob_title_tl\undefined
5600   }
5601   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
5602     \let\l__problems_inclprob_refnum_int\undefined
5603   }
5604 }
5605
5606 \cs_new_protected:Nn \l__problems_inclprob_clear: {
5607   % \str_clear:N \l__problems_prob_id_str
5608   \let\l__problems_inclprob_pts_tl\undefined
5609   \let\l__problems_inclprob_min_tl\undefined

```

```

5610 \let\l__problems_inclprob_title_tl\undefined
5611 \let\l__problems_inclprob_refnum_int\undefined
5612 \let\l__problems_inclprob_mhrepos_str\undefined
5613 }
5614
5615 \newcommand\includeproblem[2][]{
5616   \__problems_inclprob_args:n{ #1 }
5617   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
5618     \input{#2}
5619   }{
5620     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
5621       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
5622     }
5623   }
5624   \__problems_inclprob_clear:
5625 }

```

(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.

```

5626 \AddToHook{enddocument}{
5627   \bool_if:NT \c__problems_pts_bool {
5628     \message{Total:~\arabic{pts}~points}
5629   }
5630   \bool_if:NT \c__problems_min_bool {
5631     \message{Total:~\arabic{min}~minutes}
5632   }
5633 }

```

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

```

5634 \def\pts#1{
5635   \bool_if:NT \c__problems_pts_bool {
5636     \marginpar{#1~\prob@pt@kw}
5637   }
5638 }
5639 \def\min#1{
5640   \bool_if:NT \c__problems_min_bool {
5641     \marginpar{#1~\prob@min@kw}
5642   }
5643 }

```

**\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.

```

5644 \newcounter{pts}
5645 \def\show@pts{
5646   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
5647     \bool_if:NT \c__problems_pts_bool {
5648       \marginpar{\l__problems_inclprob_pts_tl;\prob@pt@kw\smallskip}
5649       \addtocounter{pts}{\l__problems_inclprob_pts_tl}

```



```

5650     }
5651   }{
5652     \tl_if_exist:NT \l__problems_prob_pts_tl {
5653       \bool_if:NT \c__problems_pts_bool {
5654         \marginpar{\l__problems_prob_pts_tl;\prob@pt@kw\smallskip}
5655         \addtocounter{pts}{\l__problems_prob_pts_tl}
5656       }
5657     }
5658   }
5659 }

```

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

\show@min

```

5660 \newcounter{min}
5661 \def\show@min{
5662   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
5663     \bool_if:NT \c__problems_min_bool {
5664       \marginpar{\l__problems_inclprob_pts_tl;min}
5665       \addtocounter{min}{\l__problems_inclprob_min_tl}
5666     }
5667   }{
5668     \tl_if_exist:NT \l__problems_prob_min_tl {
5669       \bool_if:NT \c__problems_min_bool {
5670         \marginpar{\l__problems_prob_min_tl;min}
5671         \addtocounter{min}{\l__problems_prob_min_tl}
5672       }
5673     }
5674   }
5675 }
5676 \</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.

```
5677 <@@=hwexam>
5678 <*cls>
5679 \ProvidesExplClass{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
5680 \RequirePackage{l3keys2e,expl-keystr-compatible}
5681 \DeclareOption*{
5682   \PassOptionsToClass{\CurrentOption}{omdoc}
5683   \PassOptionsToPackage{\CurrentOption}{stex}
5684   \PassOptionsToPackage{\CurrentOption}{hwexam}
5685   \PassOptionsToPackage{\CurrentOption}{tikzinput}
5686 }
5687 \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.

```
5688 \LoadClass{omdoc}
5689 \RequirePackage{stex}
5690 \RequirePackage{hwexam}
5691 \RequirePackage{tikzinput}
5692 \RequirePackage{graphicx}
5693 \RequirePackage{a4wide}
5694 \RequirePackage{amssymb}
5695 \RequirePackage{amstext}
5696 \RequirePackage{amsmath}
```

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

```

5697 \newcommand\assig@default@type{\hwexam@assignment@kw}
5698 \def\document@hwexamtype{\assig@default@type}
5699 <@@=document_structure>
5700 \keys_define:nn { document-structure / document }{
5701 id .str_set_x:N = \c_document_structure_document_id_str,
5702 hwexamtype .tl_set:N = \document@hwexamtype
5703 }
5704 <@@=hwexam>
5705 </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.

```
5706 \*package>
5707 \ProvidesExplPackage{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
5708 \RequirePackage{l3keys2e,expl-keystr-compat}
5709
5710 \newif\iftest\testfalse
5711 \DeclareOption{test}{\testtrue}
5712 \newif\ifmultiple\multiplefalse
5713 \DeclareOption{multiple}{\multipletrue}
5714 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
5715 \ProcessOptions
```

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

```
5716 \RequirePackage{keyval}[1997/11/10]
5717 \RequirePackage{problem}
```

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

```
5718 \newcommand\hwexam@assignment@kw{Assignment}
5719 \newcommand\hwexam@given@kw{Given}
5720 \newcommand\hwexam@due@kw{Due}
5721 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
5722 blank~for~extra~space}%
5723 \newcommand\correction@probs@kw{prob.}%
5724 \newcommand\correction@pts@kw{total}%
5725 \newcommand\correction@reached@kw{reached}%
5726 \newcommand\correction@sum@kw{Sum}%
5727 \newcommand\correction@grade@kw{grade}%
5728 \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

```

5729 \ifpackageloaded{babel}{\RequirePackage[base]{babel}}
5730
5731 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5732 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5733   \input{hwexam-ngerman.ldf}
5734 }
5735 \clist_if_in:NnT \l_tmpa_clist {finnish}{
5736   \input{hwexam-finnish.ldf}
5737 }
5738 \clist_if_in:NnT \l_tmpa_clist {french}{
5739   \input{hwexam-french.ldf}
5740 }
5741 \clist_if_in:NnT \l_tmpa_clist {russian}{
5742   \input{hwexam-russian.ldf}
5743 }

```

## 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.

```

5744 \newcounter{assignment}
5745 \numberproblemsin{assignment}
5746 \renewcommand\prob@label[1]{\arabic{assignment}.#1}

```

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

```

5747 \keys_define:nn { hwexam / assignment } {
5748   id .str_set:N = \l__hwexam_assign_id_str,
5749   number .int_set:N = \l__hwexam_assign_number_int,
5750   title .tl_set:N = \l__hwexam_assign_title_tl,
5751   type .tl_set:N = \l__hwexam_assign_type_tl,
5752   given .tl_set:N = \l__hwexam_assign_given_tl,
5753   due .tl_set:N = \l__hwexam_assign_due_tl,
5754   loadmodules .code:n = {
5755     \bool_set_true:N \l__hwexam_assign_loadmodules_bool
5756   }
5757 }
5758 \cs_new_protected:Nn \__hwexam_assignment_args:n {
5759   \str_clear:N \l__hwexam_assign_id_str
5760   \int_set:Nn \l__hwexam_assign_number_int {-1}
5761   \tl_clear:N \l__hwexam_assign_title_tl
5762   \tl_clear:N \l__hwexam_assign_type_tl
5763   \tl_clear:N \l__hwexam_assign_given_tl
5764   \tl_clear:N \l__hwexam_assign_due_tl
5765   \bool_set_false:N \l__hwexam_assign_loadmodules_bool
5766   \keys_set:nn { hwexam / assignment }{ #1 }
5767 }

```

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.

```

5768 \newcommand\given@due[2]{
5769 \bool_lazy_all:nF {
5770 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
5771 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
5772 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
5773 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
5774 }{ #1 }
5775
5776 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
5777 \tl_if_empty:NF \l__hwexam_assign_given_tl {
5778 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
5779 }
5780 }{
5781 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
5782 }
5783
5784 \bool_lazy_or:nnF {
5785 \bool_lazy_and_p:nn {
5786 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
5787 }{
5788 \tl_if_empty_p:V \l__hwexam_assign_due_tl
5789 }
5790 }{
5791 \bool_lazy_and_p:nn {
5792 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
5793 }{
5794 \tl_if_empty_p:V \l__hwexam_assign_due_tl
5795 }
5796 }{ ,~ }
5797
5798 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
5799 \tl_if_empty:NF \l__hwexam_assign_due_tl {
5800 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
5801 }
5802 }{
5803 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
5804 }
5805
5806 \bool_lazy_all:nF {
5807 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
5808 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
5809 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
5810 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
5811 }{ #2 }
5812 }

```

`\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.

```

5813 \newcommand\assignment@title[3]{

```

```

5814 \tl_if_empty:NTF \l__hwexam_inclassassign_title_tl {
5815 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
5816 #1
5817 }{
5818 #2\l__hwexam_assign_title_tl#3
5819 }
5820 }{
5821 #2\l__hwexam_inclassassign_title_tl#3
5822 }
5823 }

```

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

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

```

5824 \newcommand\assignment@number{
5825 \int_compare:nNnTF \l__hwexam_inclassassign_number_int = {-1} {
5826 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
5827 \int_use:N \l__hwexam_assign_number_int
5828 }
5829 }{
5830 \int_use:N \l__hwexam_inclassassign_number_int
5831 }
5832 }

```

(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.

```

5833 \newenvironment{assignment}[1][]{
5834 \__hwexam_assignment_args:n { #1 }
5835 %\sref@target
5836 \let\__hwexamnum\l__hwexam_assign_number_int
5837 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
5838 \stepcounter{assignment}
5839 }{
5840 \setcounter{assignment}{\int_use:N\__hwexamnum}
5841 }
5842 \setcounter{problem}{0}
5843 \def\current@section@level{\document@hwexamtype}
5844 %\sref@label@id{\document@hwexamtype \thesection}
5845 \begin{@assignment}
5846 }{
5847 \end{@assignment}
5848 }

```

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

```

5849 \def\__hwexasstitle{
5850 \protect\document@hwexamtype~\arabic{assignment}
5851 \assignment@title{}\;{} \; -- \given@due{}{}
5852 }

```

```

5853 \ifmultiple
5854 \newenvironment{@assignment}{
5855 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
5856 \begin{omgroup}[loadmodules]{\__hwexasstitle}
5857 }{
5858 \begin{omgroup}{\__hwexasstitle}
5859 }
5860 }{
5861 \end{omgroup}
5862 }

```

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

```

5863 \else
5864 \newenvironment{@assignment}{
5865 \begin{center}\bf
5866 \Large\@title\strut\
5867 \document@hwexamtype~\arabic{assignment}\assignment@title{\;}{:};{\}\}
5868 \large\given@due{--\;}{\;}{--}
5869 \end{center}
5870 }{}
5871 \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.

```

5872 \keys_define:nn { hwexam / inclassignment } {
5873 %id .str_set_x:N = \l__hwexam_assign_id_str,
5874 number .int_set:N = \l__hwexam_inclassign_number_int,
5875 title .tl_set:N = \l__hwexam_inclassign_title_tl,
5876 type .tl_set:N = \l__hwexam_inclassign_type_tl,
5877 given .tl_set:N = \l__hwexam_inclassign_given_tl,
5878 due .tl_set:N = \l__hwexam_inclassign_due_tl,
5879 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
5880 }
5881 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
5882 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
5883 \tl_clear:N \l__hwexam_inclassign_title_tl
5884 \tl_clear:N \l__hwexam_inclassign_type_tl
5885 \tl_clear:N \l__hwexam_inclassign_given_tl
5886 \tl_clear:N \l__hwexam_inclassign_due_tl
5887 \str_clear:N \l__hwexam_inclassign_mhrepos_str
5888 \keys_set:nn { hwexam / inclassignment }{ #1 }
5889 }
5890 \__hwexam_inclassignment_args:n {}
5891
5892 \newcommand\inputassignment[2][ ]{
5893 \__hwexam_inclassignment_args:n { #1 }
5894 \str_if_empty:NTF \l__hwexam_inclassign_mhrepos_str {
5895 \input{#2}
5896 }{
5897 \stex_in_repository:nn{\l__hwexam_inclassign_mhrepos_str}{

```



```

5898 \input{\mhp\path{\l__hwexam_inclasssign_mhrepos_str}{#2}}
5899 }
5900 }
5901 \__hwexam_inclasssign_assignment_args:n {}
5902 }
5903 \newcommand\includeassignment[2][ ]{
5904 \newpage
5905 \inputassignment[#1]{#2}
5906 }

```

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

## 42.4 Typesetting Exams

\quizheading

```

5907 \ExplSyntaxOff
5908 \newcommand\quizheading[1]{%
5909 \def\@tas{#1}%
5910 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
5911 \ifx\@tas\empty\else%
5912 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
5913 \fi%
5914 }
5915 \ExplSyntaxOn

```

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

\testheading

```

5916 \keys_define:nn { hwexam / testheading } {
5917 min .tl_set:N = \l__hwexam_testheading_min_tl,
5918 duration .tl_set:N = \l__hwexam_testheading_duration_tl,
5919 reqpts .tl_set:N = \l__hwexam_testheading_reqpts_tl
5920 }
5921 \cs_new_protected:Nn \__hwexam_testheading_args:n {
5922 \tl_clear:N \l__hwexam_testheading_min_tl
5923 \tl_clear:N \l__hwexam_testheading_duration_tl
5924 \tl_clear:N \l__hwexam_testheading_reqpts_tl
5925 \keys_set:nn { hwexam / testheading }{ #1 }
5926 }
5927 \newenvironment{testheading}[1][ ]{
5928 \__hwexam_testheading_args:n{ #1 }
5929 \noindent\large{Name:~\hfill
5930 Matriculation Number:\hspace*{2cm}\strut}\[1ex]
5931 \begin{center}
5932 \Large\textbf{\@title}\[1ex]
5933 \large\@date\[3ex]
5934 \end{center}
5935 \textbf{You~have~
5936 \tl_if_empty:NTF \l__hwexam_testheading_duration_tl {
5937 {\l__hwexam_testheading_min_tl}~minutes
5938 }{
5939 {\l__hwexam_testheading_duration_tl}
5940 }~

```

```

5941 (sharp)~for~the~test
5942 };\
5943 Write~the~solutions~to~the~sheet.
5944 \par\noindent
5945 \newcount\check@time\check@time=\l__hwexam_testheading_min_tl
5946 \advance\check@time by -\theassignment@totalmin
5947 The~estimated~time~for~solving~this~exam~is~
5948 {\theassignment@totalmin}~minutes,~
5949 leaving~you~{\the\check@time}~minutes~for~revising~
5950 your~exam.
5951
5952 \par\noindent
5953 \newcount\bonus@pts\bonus@pts=\theassignment@totalpts
5954 \advance\bonus@pts by -\l__hwexam_testheading_reqpts_tl
5955 You~can~reach~{\theassignment@totalpts}~points~if~you~
5956 solve~all~problems.~You~will~only~need~
5957 {\l__hwexam_testheading_reqpts_tl}~points~for~a~perfect~score,~
5958 i.e.\ {\the\bonus@pts}~points~are~bonus~points.
5959 \vfill
5960 \begin{center}
5961 {
5962 \Large\em You~have~ample~time,~so~take~it~slow~
5963 and~avoid~rushing~to~mistakes!\}[2ex]
5964 Different~problems~test~different~skills~and~
5965 knowledge,~so~do~not~get~stuck~on~one~problem.
5966 }
5967 \vfill\par\resizebox{\textwidth}{!}{\correction@table}\}[3ex]
5968 \end{center}
5969 }{
5970 \newpage
5971 }

```

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

\testspace

```

5972 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

5973 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

5974 \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.

```

5975 <@=problems>
5976 \renewcommand\@problem[3]{
5977 \stepcounter{assignment@probs}
5978 \def\__problemspts{#2}

```

```

5979 \ifx\__problemspts\@empty\else
5980 \addtocounter{assignment@totalpts}{#2}
5981 \fi
5982 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
5983 \xdef\correction@probs{\correction@probs & #1}%
5984 \xdef\correction@pts{\correction@pts & #2}
5985 \xdef\correction@reached{\correction@reached & }
5986 }
5987 \@@=hwexam

```

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

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

```

5988 \newcounter{assignment@probs}
5989 \newcounter{assignment@totalpts}
5990 \newcounter{assignment@totalmin}
5991 \def\correction@probs{\correction@probs@kw}%
5992 \def\correction@pts{\correction@pts@kw}%
5993 \def\correction@reached{\correction@reached@kw}%
5994 \def\after@correction@table{}%
5995 \stepcounter{assignment@probs}
5996 \newcommand\correction@table{
5997 \resizebox{\textwidth}{!}{%
5998 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
5999 &\multicolumn{\theassignment@probs}{c|}||%|
6000 {\footnotesize\correction@forgrading@kw} &\\ \hline
6001 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
6002 \correction@pts & \theassignment@totalpts & \\ \hline
6003 \correction@reached & & \[.7cm]\hline
6004 \end{tabular}}
6005 \ifx\after@correction@table\@empty\else\strut\par\noindent\after@correction@table\fi}
6006 \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\wierfont=../assignments/wierfont
\font\denkerfont=../assignments/denker
\font\uhrfont=../assignments/uhr
\font\warnschildfont=../assignments/achtung

```

```

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

```