

# The sTeX3 Package \*

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

2022-02-13

## Abstract

sTeX is a collection of L<sup>A</sup>T<sub>E</sub>X packages 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 2022-02-13)

# 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 document-structure: Semantic Markup for Open Mathematical Documents in <math>\text{\LaTeX}</math></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 . . . . .	56
21.2.4 Structure Sharing . . . . .	56
21.2.5 Global Variables . . . . .	56
21.2.6 Colors . . . . .	57
21.3 Limitations . . . . .	57
<b>22 NotesSlides – 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 . . . . .	62
22.3 Limitations . . . . .	62
<b>23 problem.sty: An Infrastructure for formatting Problems</b>	<b>63</b>
23.1 Introduction . . . . .	63
23.2 The User Interface . . . . .	63
23.2.1 Package Options . . . . .	63
23.2.2 Problems and Solutions . . . . .	64
23.2.3 Multiple Choice Blocks . . . . .	65
23.2.4 Including Problems . . . . .	65
23.2.5 Reporting Metadata . . . . .	65
23.3 Limitations . . . . .	65
<b>24 hwexam.sty/cls: An Infrastructure for formatting Assignments and Exams</b>	<b>67</b>
24.1 Introduction . . . . .	68
24.2 The User Interface . . . . .	68
24.2.1 Package and Class Options . . . . .	68
24.2.2 Assignments . . . . .	68
24.2.3 Typesetting Exams . . . . .	68
24.2.4 Including Assignments . . . . .	69
24.3 Limitations . . . . .	69
 <b>IV Implementation</b>	 <b>71</b>

<b>25</b>	<b>STeX-Basics Implementation</b>	<b>72</b>
25.1	The STeXDocument Class . . . . .	72
25.2	Preliminaries . . . . .	72
25.3	Messages and logging . . . . .	73
25.4	Persistence . . . . .	74
25.5	HTML Annotations . . . . .	74
25.6	Languages . . . . .	77
25.7	Activating/Deactivating Macros . . . . .	78
<b>26</b>	<b>STeX-MathHub Implementation</b>	<b>80</b>
26.1	Generic Path Handling . . . . .	80
26.2	PWD and kpsewhich . . . . .	82
26.3	File Hooks and Tracking . . . . .	83
26.4	MathHub Repositories . . . . .	84
<b>27</b>	<b>STeX-References Implementation</b>	<b>92</b>
27.1	Document URIs and URLs . . . . .	92
27.2	Setting Reference Targets . . . . .	94
27.3	Using References . . . . .	95
<b>28</b>	<b>STeX-Modules Implementation</b>	<b>98</b>
28.1	The module environment . . . . .	101
28.2	Invoking modules . . . . .	107
<b>29</b>	<b>STeX-Module Inheritance Implementation</b>	<b>109</b>
29.1	SMS Mode . . . . .	109
29.2	Inheritance . . . . .	113
<b>30</b>	<b>STeX-Symbols Implementation</b>	<b>118</b>
30.1	Symbol Declarations . . . . .	118
30.2	Notations . . . . .	125
<b>31</b>	<b>STeX-Terms Implementation</b>	<b>134</b>
31.1	Symbol Invocations . . . . .	134
31.2	Terms . . . . .	137
31.3	Notation Components . . . . .	143
<b>32</b>	<b>STeX-Structural Features Implementation</b>	<b>146</b>
32.1	Imports with modification . . . . .	146
32.2	The feature environment . . . . .	153
32.3	Features . . . . .	155
<b>33</b>	<b>STeX-Statements Implementation</b>	<b>160</b>
33.1	Definitions . . . . .	160
33.2	Assertions . . . . .	165
33.3	Examples . . . . .	168
33.4	Logical Paragraphs . . . . .	170

<b>34 The Implementation</b>	<b>175</b>
34.1 Package Options . . . . .	175
34.2 Proofs . . . . .	175
34.3 Justifications . . . . .	181
<b>35 <math>\text{\TeX}</math>-Others Implementation</b>	<b>183</b>
<b>36 <math>\text{\TeX}</math>-Metatheory Implementation</b>	<b>184</b>
<b>37 Tikzinput Implementation</b>	<b>187</b>
<b>38 document-structure.sty Implementation</b>	<b>189</b>
38.1 The document-structure Class . . . . .	189
38.2 Class Options . . . . .	189
38.3 Beefing up the <code>document</code> environment . . . . .	190
38.4 Implementation: document-structure Package . . . . .	190
38.5 Package Options . . . . .	190
38.6 Document Structure . . . . .	192
38.7 Front and Backmatter . . . . .	195
38.8 Global Variables . . . . .	197
<b>39 NotesSlides – Implementation</b>	<b>198</b>
39.1 Class and Package Options . . . . .	198
39.2 Notes and Slides . . . . .	200
39.3 Header and Footer Lines . . . . .	204
39.4 Frame Images . . . . .	205
39.5 Colors and Highlighting . . . . .	206
39.6 Sectioning . . . . .	207
39.7 Excursions . . . . .	209
<b>40 The Implementation</b>	<b>211</b>
40.1 Package Options . . . . .	211
40.2 Problems and Solutions . . . . .	212
40.3 Multiple Choice Blocks . . . . .	218
40.4 Including Problems . . . . .	219
40.5 Reporting Metadata . . . . .	220
<b>41 Implementation: The hwexam Class</b>	<b>222</b>
41.1 Class Options . . . . .	222
<b>42 Implementation: The hwexam Package</b>	<b>224</b>
42.1 Package Options . . . . .	224
42.2 Assignments . . . . .	225
42.3 Including Assignments . . . . .	228
42.4 Typesetting Exams . . . . .	229
42.5 Leftovers . . . . .	231

**Part I**  
**Manual**

# Chapter 1

## What is sTeX?

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

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

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

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

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

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



# Chapter 2

## Quickstart

### 2.1 Setup

#### 2.1.1 The sTeX IDE

TODO: VSCode Plugin

#### 2.1.2 Manual Setup

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

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

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

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

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

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

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

---

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

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

- **RuSTeX** The MMT system will also set up RuSTeX for you, which is used to generate (semantically annotated) xhtml from tex sources. In lieu of using MMT, you can also download and use RuSTeX directly [here](#).

## 2.2 A First sTeX Document

Having set everything up, we can write a first sTeX 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 sTeX *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}`), sTeX 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}`.

sTeX 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/arithmetics,smglom/sets,smglom/topology,
              smglom/mv,smglom/linear-algebra,smglom/algebra
responsible: Michael.Kohlhase@FAU.de
title: Elementary Calculus
teaser: Terminology for the mathematical study of change.
description: desc.html
```

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

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

`source-base` or

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

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

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

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

## Chapter 5

# Creating New Modules and Symbols

TODO

### 5.1 Advanced Structuring Mechanisms

Given modules:

#### Example 1

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

Module 5.1.1[magma]

Module 5.1.2[monoid]

Module 5.1.3[group]

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

## Example 2

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

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

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

TODO: explain donotclone

## Example 3

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

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

Module 5.1.5[int]

## 5.2 Primitive Symbols (The sTeX Metatheory)



## Chapter 6

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

## Chapter 7

# Additional Packages

**7.1** Modular Document Structuring

**7.2** Slides and Course Notes

**7.3** Homework, Problems and Exams

# Chapter 8

# Stuff

## 8.1 Modules

---

`\sTeX`  
`\stex`

---

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

### 8.1.1 Semantic Macros and Notations

Semantic macros invoke a formally declared symbol.

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

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

#### Example 4

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

$ab$

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

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

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

### Example 5

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

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

.

Not using an explicit option with a semantic macro yields the first declared notation, unless changed<sup>4</sup>.

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

### Example 6

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

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

.

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

### Example 7

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

Multiplying again by  $b$  yields...

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

### Example 8

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

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

<sup>4</sup>EdNOTE: TODO

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

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

### Example 9

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

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

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

### Example 10

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

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

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

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

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

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

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

## Other Argument Types

So far, we have stated the arity of a semantic macro directly. This works if we only have “normal” (or more precisely: *i*-type) arguments. To make use of other argument types, instead of providing the arity numerically, we can provide it as a sequence of characters

representing the argument types – e.g. instead of writing `args=2`, we can equivalently write `args=ii`, indicating that the macro takes two i-type arguments.

Besides i-type arguments,  $\text{\TeX}$  has two other types, which we will discuss now.

The first are *binding* (b-type) arguments, representing variables that are *bound* by the operator. This is the case for example in the above `\forevery`-macro: The first argument is not actually an argument that the `forevery` “function” is “applied” to; rather, the first argument is a new variable (e.g.  $x$ ) that is *bound* in the subsequent argument. More accurately, the macro should therefore have been implemented thusly:

```
\symdef[args=bi]{forevery}{\forall #1.\; #2}
```

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

### Example 11

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

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

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

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

### Example 12

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

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

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

5 6

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

<sup>6</sup>EDNOTE: “decompose” a-type arguments into fixed-arity operators?

## Precedences

Every notation has an (upwards) *operator precedence* and for each argument a (downwards) *argument precedence* used for automated bracketing. For example, a notation for a binary operator `\foo` could be declared like this:

```
\notation[prec=200;500x600]{foo}{#1 \comp{+} #2}
```

assigning an operator precedence of 200, an argument precedence of 500 for the first argument, and an argument precedence of 600 for the second argument.

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

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

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

For example:

### Example 13

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

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

## 8.1.2 Archives and Imports

### Namespaces

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

Unfortunately,  $\TeX$  only provides very restricted access to the file system, so we are forced to generate namespaces systematically in such a way that they reflect the physical location of the associated files, so that  $\TeX$  can resolve them accordingly. Largely, users need not concern themselves with namespaces at all, but for completeness sake, we describe how they are constructed:

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

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

If the current file is in an archive, the procedure is the same except that the initial segment of the file path up to the archive's `source`-folder is replaced by the archive's namespace URI.

### Paths in Import-Statements

Conversely, here is how namespaces/URIs and file paths are computed in import statements, exemplary `\importmodule`:

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

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

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

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

---

<sup>1</sup>which is internally attached to the module name instead, but a user need not worry about that.



## Part II

# Documentation

## Chapter 9

# sTeX-Basics

Both the sTeX package and class offer the following package options:

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

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

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

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

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

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

### 9.1 Macros and Environments

---

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

---

---

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

---

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

---

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

---

---

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

---

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

---

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

---

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

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

---

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

---

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

---

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

---

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

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

---

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

---

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

# Chapter 10

## sTeX-MathHub

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

### 10.1 Macros and Environments

---

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

---

#### 10.1.1 Files, Paths, URIs

---

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

---

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

---

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

---

---

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

---

---

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

---

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

---

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

---

`\g_stex_currentfile_seq`

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

**Test 1**

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

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

**10.1.2 MathHub Archives**

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

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

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

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

`\l_stex_current_repository_prop`

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

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

## Test 2

```

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

```

```

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

```

## Chapter 11

# sTeX-References

Code related to links and cross-references

### 11.1 Macros and Environments

# Chapter 12

## sTeX-Modules

Code related to Modules

### 12.1 Macros and Environments

---

---

`\l_stex_current_module_str`

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

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

Additionally, it stores:

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

---

---

`\l_stex_all_modules_seq`

Stores full URIs for all modules currently in scope.



---

```
\g_stex_module_files_prop
\g_stex_modules_in_file_seq
```

---

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

---

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

---



---

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

---

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

---

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

---

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

---

```
\stex_add_constant_to_current_module:n
```

---

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

---

```
\stex_add_import_to_current_module:n
```

---

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

---

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

---

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

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

---

```
\stex_modules_current_namespace:
```

---

Computes the current namespace

### Test 3

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

```

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

```

### 12.1.1 The module-environment

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

---

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

---

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

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

### Test 4

```

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

```

```

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

```

## Test 5

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

```
Module 12.1.1[Bar] (FooBar)
Module path: http://mathhub.info/tests/Foo/Bar/Foo?Bar
Language:
Signature:
Metatheory:
```

---

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

---

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

---

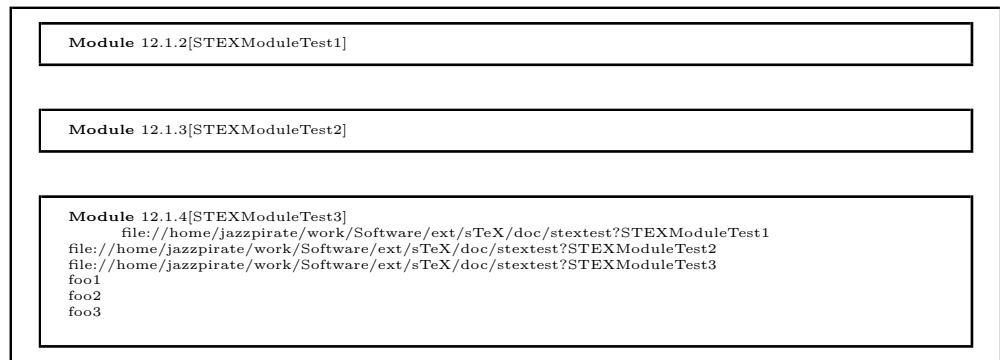
`\stex_invoke_module:n`

---

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

## Test 6

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




---

`\stex_activate_module:n`

---

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

# Chapter 13

## STEX-Module Inheritance

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

### 13.1 Macros and Environments

#### 13.1.1 SMS Mode

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

---

`\g_stex_smsmode_allowedmacros_tl`

---

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

---

`\g_stex_smsmode_allowedmacros_escape_tl`

---

Macros that are executed with the category codes restored.

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

---

`\g_stex_smsmode_allowedenvs_seq`

---

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

---

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

---

Tests whether SMS mode is currently active.

---

`\stex_smsmode_set_codes:`

---

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

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

---

**`\stex_in_smsmode:nn`**

---

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

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

### Test 7

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

## 13.1.2 Imports and Inheritance

---

**`\importmodule`**

---

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

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

### Test 8

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

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

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

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

**Module 13.1.3[Importtest2]**  
Meaning: `\macro->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo?foo}<`

---

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

---

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

## Test 9

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

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

Module 13.1.4[UseTest1]

Module 13.1.5[UseTest2]

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

Module 13.1.6[UseTest3]

Meaning: >undefined<

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

All modules: <http://mathhub.info/sTeX?Metatheory>, <file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest3>,  
 All symbols: <http://mathhub.info/sTeX?Metatheory?isa>, <http://mathhub.info/sTeX?Metatheory?bind>, <http://mathhub.info/sTeX?Metatheory?collect>,  
<http://mathhub.info/sTeX?Metatheory?fromto>, <http://mathhub.info/sTeX?Metatheory?apply>, <http://mathhub.info/sTeX?Metatheory?sequence-index>, <http://mathhub.info/sTeX?Metatheory?seqtype>, <http://mathhub.info/sTeX?Metatheory?aseqfromto>, <http://mathhub.info/sTeX?Metatheory?aseqfromtovia>, <http://mathhub.info/sTeX?Metatheory?module-type>, <http://mathhub.info/sTeX?Metatheory?mathematical-structure>,  
<http://mathhub.info/sTeX?Metatheory?isa>, <http://mathhub.info/sTeX?Metatheory?bind>, <http://mathhub.info/sTeX?Metatheory?dummyvar>,  
<http://mathhub.info/sTeX?Metatheory?fromto>, <http://mathhub.info/sTeX?Metatheory?apply>, <http://mathhub.info/sTeX?Metatheory?collect>,  
<http://mathhub.info/sTeX?Metatheory?aseqfromto>, <http://mathhub.info/sTeX?Metatheory?aseqfromtovia>, <http://mathhub.info/sTeX?Metatheory?module-type>, <http://mathhub.info/sTeX?Metatheory?mathematical-structure>,  
<file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest2?bar>

## Test 10

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

Circular dependencies:

```
Module 13.1.7[CircDep1]
  >macro:->\stex_invoke_symbol:n {http://mathhub.info/tests/Foo/Bar/circular1?Circular1?fooA}<
  >macro:->\stex_invoke_symbol:n {http://mathhub.info/tests/Bar/Foo//circular2?Circular2?fooB}<
```

---

`\stex_import_module_uri:nn`    `\stex_import_module_uri:nn {<archive-ID>} {<module-path>}`

---

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

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

1. If `<archive-ID>` is empty:

- (a) If `<path>` is empty, then `<name>` must have been declared earlier in the same file and retrievable from `\g_stex_modules_in_file_seq`, or a file with name `<name>.<lang>.tex` must exist in the same folder, containing a module `<name>`. That module should have the same namespace as the current one.
- (b) If `<path>` is not empty, it must point to the relative path of the containing file as well as the namespace.

2. Otherwise:

- (a) If `<path>` is empty, then `<name>` must have been declared earlier in the same file and retrievable from `\g_stex_modules_in_file_seq`, or a file with name `<name>.<lang>.tex` must exist in the top `source` folder of the archive, containing a module `<name>`. That module should lie directly in the namespace of the archive.
- (b) If `<path>` is not empty, it must point to the path of the containing file as well as the namespace, relative to the namespace of the archive. If a module by that namespace exists, it is returned. Otherwise, we call `\stex_require_module:nn` on the `source` directory of the archive to find the file.

---

`\stex_import_require_module:nnnn`    `{<ns>} {<archive-ID>} {<path>} {<name>}`

---

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

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



# Chapter 14

## TeX-Symbols

Code related to symbol declarations and notations

### 14.1 Macros and Environments

---

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

---

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

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

---

---

`\stex_symdecl_do:n`

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

Ultimately stores the symbol  $\langle URI \rangle$  in the property list `\l_stex_symdecl_⟨URI⟩_prop` with fields:

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

### Test 11

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

```
Module 14.1.1[SymdeclTest]
Meaning: >macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?SymdeclTest?foo}<
Result: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?SymdeclTest?foo
Meaning: >macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?SymdeclTest?bardef}<
```

---

---

`\l_stex_all_symbols_seq`

Stores full URIs for all modules currently in scope.

---

---

`\stex_get_symbol:n`

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

---

---

`\notation`

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

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

---

---

`\stex_notation_do:nn`

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

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

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

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

### Test 12

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

Module 14.1.2[NotationTest]

---

---

`\symdef`

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

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

### Test 13

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

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

# Chapter 15

## STEX-Terms

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

### 15.1 Macros and Environments

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

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

### Test 14

```

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

```

Module 15.1.1[MathTest1]  
 $\langle a^b_c \rangle$  and  $\langle a^b_c \rangle$ .

### Test 15

```

\begin{module}{MathTest2}
\importmodule{Foo}
\notation[foo, prec=500;20x20x20]{foobar}{\comp\langle #1 \comp\mid [ #2 ] ^{#3} \comp\rangle }{ {#1}_{\comp\langle #1 \comp\mid [ #2 ] ^{#3} \comp\rangle } }
 $\bar{foo} a\{b,c,d,e,f\}g$  and  $\bar{foo} a\{b,c\}g$  and  $\bar{foo} abc$ 
\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  $\mult{a,\plus{\frac{ab}{ac}}}$ 
 $\displaystyle \plus{a,\mult{b,c}}$  and  $\mult{a,\plus{\frac{ab}{ac}}}$ 
\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{spfcase}{For the induction we have to consider the following cases:}
    \begin{spfcase}{ $n=1$ }
      \begin{spfstep}[display=flow] then we compute  $1=1^2$ \end{spfstep}
    \end{spfcase}
    \begin{spfcase}{ $n=2$ }
      \begin{sproofcomment}[display=flow]
        This case is not really necessary, but we do it for the
        fun of it (and to get more intuition).
      \end{sproofcomment}
      \begin{spfstep}[display=flow] We compute  $1+3=2^2=4$ .\end{spfstep}
    \end{spfcase}
    \begin{spfcase}{ $n>1$ }
      \begin{spfstep}[type=assumption,id=ind-hyp]
        Now, we assume that the assertion is true for a certain  $k \geq 1$ ,
        i.e.  $\sum_{i=1}^k (2i-1) = k^2$ $.
      \end{spfstep}
      \begin{sproofcomment}
        We have to show that we can derive the assertion for  $n=k+1$  from
        this assumption, i.e.  $\sum_{i=1}^{k+1} (2i-1) = (k+1)^2$ $.
      \end{sproofcomment}
      \begin{spfstep}
        We obtain  $\sum_{i=1}^{k+1} (2i-1) = \sum_{i=1}^k (2i-1) + 2(k+1) - 1$ 
        \begin{justification}[method=arith:split-sum]
          by splitting the sum.
        \end{justification}
      \end{spfstep}
      \begin{spfstep}
        Thus we have  $\sum_{i=1}^{k+1} (2i-1) = k^2 + 2k + 1$ 
        \begin{justification}[method=fertilize]
          by inductive hypothesis.
        \end{justification}
      \end{spfstep}
      \begin{spfstep}[type=conclusion]
        We can \begin{justification}[method=simplify]simplify\end{justification}
        the right-hand side to  $(k+1)^2$ $, which proves the assertion.
      \end{spfstep}
    \end{spfcase}
    \begin{spfstep}[type=conclusion]
      We have considered all the cases, so we have proven the assertion.
    \end{spfstep}
  \end{spfcase}
\end{sproof}
```

Example 1: A very explicit proof, marked up semantically

We will go over the general intuition by way of our running example (see Figure 1 for the source and Figure 2 for the formatted result).<sup>7</sup>

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

## 18.2 The User Interface

### 18.2.1 Package Options

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

### 18.2.2 Proofs and Proof steps

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

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

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

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

### 18.2.3 Justifications

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

#### 18.2.4 Proof Structure

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



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: Semantic Markup for Open Mathematical Documents in L<sup>A</sup>T<sub>E</sub>X

The `document-structure` package is part of the  $\text{\S}$ TeX 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 an infrastructure for writing OMDOC documents in L<sup>A</sup>T<sub>E</sub>X. This includes a simple structure sharing mechanism for  $\text{\S}$ TeX that allows to move from a copy-and-paste document development model to a copy-and-reference model, which conserves space and simplifies document management. The augmented structure can be used by MKM systems for added-value services, either directly from the  $\text{\S}$ TeX sources, or after translation.

### 21.1 Introduction

$\text{\S}$ TeX is 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). The package supports direct translation to the OMDOC format [Koh06]

The `document-structure` 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  $\text{\S}$ TeX sources, or after translation. Currently, trans-document referencing provided by this package can only be used in the  $\text{\S}$ TeX collection.

DAG models of documents allow to replace the “Copy and Paste” in the source document with a label-and-reference model where document are shared in the document

source and the formatter does the copying during document formatting/presentation.<sup>9</sup>

## 21.2 The User Interface

The `document-structure` package generates two files: `document-structure.cls`, and `document-structure.sty`. The OMDoc class is a minimally changed variant of the standard `article` class that includes the functionality provided by `document-structure.sty`. The rest of the documentation pertains to the functionality introduced by `document-structure.sty`.

### 21.2.1 Package and Class Options

The `document-structure` 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 `document-structure` package accepts the same except the first two.

### 21.2.2 Document Structure

<code>document</code>	The top-level <code>document</code> environment can be given key/value information by the
<code>\documentkeys</code>	<code>\documentkeys</code> macro in the preamble <sup>2</sup> . This can be used to give metadata about the
<code>id</code>	document. For the moment only the <code>id</code> key is used to give an identifier to the <code>omdoc</code>
<code>omgroup</code>	element resulting from the L <sup>A</sup> T <sub>E</sub> XML transformation.
	The structure of the document is given by the <code>omgroup</code> environment just like in OM-
	DOC. In the L <sup>A</sup> T <sub>E</sub> X route, the <code>omgroup</code> environment is flexibly mapped to sectioning com-
	mands, inducing the proper sectioning level from the nesting of <code>omgroup</code> environments.
	Correspondingly, the <code>omgroup</code> environment takes an optional key/value argument for
	metadata followed by a regular argument for the (section) title of the <code>omgroup</code> . The op-
<code>id</code>	tional metadata argument has the keys <code>id</code> for an identifier, <code>creators</code> and <code>contributors</code>
<code>creators</code>	for the Dublin Core metadata [DCM03]; see [Koh20a] for details of the format. The
<code>contributors</code>	<code>short</code> allows to give a short title for the generated section. If the title contains semantic
<code>short</code>	macros, they need to be protected by <code>\protect</code> , and we need to give the <code>loadmodules</code>
<code>loadmodules</code>	key it needs no value. For instance we would have

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

<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.

`blindomgroup`

$\text{\TeX}$  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 `document-structure` package provides a variant `blindomgroup` 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.

<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.

### 21.2.3 Ignoring Inputs

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

`\prematurestop` 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.

`\afterprematurestop` `\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.

`\setSGvar`  
`\useSGvar`  
`\ifSGvar`

With `\ifSGvar` we can test for the contents of a global variable: the macro call

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

`\ifSGvar{⟨vname⟩}{⟨val⟩}{⟨ctext⟩}` tests the content of the global variable `⟨vname⟩`, only if (after expansion) it is equal to `⟨val⟩`, the conditional text `⟨ctext⟩` is formatted.

### 21.2.6 Colors

For convenience, the `document-structure` 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

# NotesSlides – 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 `notesslides` 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 `notesslides` class takes the notion of a slide frame from Till Tantau’s excellent `beamer` class and adapts its notion of frames for use in the  $\text{\LaTeX}$  and OMDoc. To support semantic course notes, it extends the notion of mixing frames and explanatory text, but rather than treating the frames as images (or integrating their contents into the flowing text), the `notesslides` 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 `notesslides` class has two modes: *slides mode* and *notes mode* which are determined by the package option.

#### 22.2.1 Package Options

The `notesslides` class takes a variety of class options:<sup>11</sup>

- |                     |   |
|---------------------|---|
| <code>slides</code> | • The options <code>slides</code> and <code>notes</code> switch between slides mode and notes mode (see |
| <code>notes</code>  | Section 22.2.2).  |



<code>sectocframes</code>	<ul style="list-style-type: none"> <li>If the option <code>sectocframes</code> is given, then for the <code>omgroups</code>, special frames with the <code>omgroup</code> title (and number) are generated.</li> </ul>
<code>showmeta</code>	<ul style="list-style-type: none"> <li><code>showmeta</code>. If this is set, then the metadata keys are shown (see [Koh20b] for details and customization options).</li> </ul>
<code>frameimages</code> <code>fiboxed</code>	<ul style="list-style-type: none"> <li>If the option <code>frameimages</code> is set, then slide mode also shows the <code>\frameimage</code>-generated frames (see section 22.2.4). If also the <code>fiboxed</code> option is given, the slides are surrounded by a box.</li> </ul>
<code>topsect</code>	<ul style="list-style-type: none"> <li><code>topsect=&lt;sect&gt;</code> can be used to specify the top-level sectioning level; the default for <code>&lt;sect&gt;</code> is <code>section</code>.</li> </ul>

## 22.2.2 Notes and Slides

`frame` Slides are represented with the `frame` just like in the `beamer` class, see [Tanb] for details.  
`note` The `notesslides` 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

<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.

`notes` and `slides` mode – manually setting `\notesttrue` or `\notesfalse` is strongly discouraged however.

⚠: 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 `notesslides` 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 `nparagraph` environment is just an `sparagraph` 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.

`nparagraph`  
`nomgroup`  
`ndefinition`  
`nexample`  
`nsproof`  
`nassertion`

### 22.2.3 Header and Footer Lines of the Slides

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

The default footer line of the `notesslides` 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 `notesslides` 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.

`\setsource`  
`\setlicensing`

### 22.2.4 Frame Images

Sometimes, we want to integrate slides as images after all – e.g. because we already have a PowerPoint presentation, to which we want to add  $\text{\TeX}$ 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>

`\frameimage`  
`\mhframeimage`

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}
```


<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.

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

```
\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{nparagraph}[title=Excursion]
  \activateexcursion{founif}{../ex/founif}
  We will cover first-order unification in \sref{founif}.
\end{nparagraph}
```

```
\activateexcursion      where \activateexcursion{<path>} augments the \printexcursions macro by a
\printexcursions        call \inputref{<path>}. In this way, the3 \printexcursions macro (usually in the
                        appendix) will collect up all excursions that are specified in the main text.
```

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

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

Finally, we usually want to put the excursions into an `omgroup` environment and add an introduction, therefore we provide the a variant of the `\printexcursions` macro:

```
\excursiongroup \excursiongroup[id=<id>,intro=<path>] is equivalent to
```

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

### 22.2.8 Miscellaneous

## 22.3 Limitations

In this section we document known limitations. If you want to help alleviate them, please feel free to contact the package author. Some of them are currently discussed in the  $\text{\TeX}$ 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. 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{sproblem}[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{sproblem}
\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).

```
Problem 0.1 (Fitting Elephants)
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{sproblem}[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{sproblem}

```

---

**Problem 0.2 (Functions)**

What is the keyword to introduce a function definition in python?

1. def
2. function
3. fun
4. public static void

---

**Problem 0.3 (Functions)**

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.

---

Name: \_\_\_\_\_ Matriculation Number: \_\_\_\_\_

2022-02-13

You can reach 30 points if you solve all problems. You will only need 27 points for a perfect score, i.e. 3 points are bonus points.

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

[illegible]

Example 8: A generated test heading.

Part IV

# Implementation

## Chapter 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:
\TeX
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\TeX{\stex}

```

(End definition for `\stex` and `\TeX`. 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:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
267     }
268   }
269   \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
270   \RequirePackage[\clist_use:Nn \l_tmpa_clist,]{babel}
271 }

```

## 25.7 Activating/Deactivating Macros

`\stex_deactivate_macro:Nn`

```

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

```

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

`\stex_reactivate_macro:N`

```

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

```

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

`\stex_do_aftergroup:nn`

```

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

```

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

```

301 </package>

```

## Chapter 26

# STEX -MathHub Implementation

```
302 <*package>
303
304 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
305
306 <@@=stex_path>
307
308 Warnings and error messages
309 \msg_new:nnn{stex}{error/norepository}{
310   No~archive~#1~found~in~#2
311 }
312 \msg_new:nnn{stex}{error/notinarchive}{
313   Not~currently~in~an~archive,~but~\detokenize{#1}~
314   needs~one!
315 }
316 \msg_new:nnn{stex}{error/nofile}{
317   \detokenize{#1}~could~not~find~file~#2
318 }
319 \msg_new:nnn{stex}{error/twofiles}{
320   \detokenize{#1}~found~two~candidates~for~#2
321 }
```

### 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
320 \cs_new_protected:Nn \stex_path_from_string:Nn {
321   \str_set:Nx \l_tmpa_str { #2 }
322   \str_if_empty:NTF \l_tmpa_str {
323     \seq_clear:N #1
324   }{
325     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
326     \sys_if_platform_windows:T{
327       \seq_clear:N \l_tmpa_tl
```

```

328     \seq_map_inline:Nn #1 {
329       \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
330       \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
331     }
332     \seq_set_eq:NN #1 \l_tmpa_tl
333   }
334   \stex_path_canonicalize:N #1
335 }
336 }
337 \cs_generate_variant:Nn \stex_path_from_string:Nn
338 { 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`

```

339 \cs_new_protected:Nn \stex_path_to_string:NN {
340   \exp_args:Nne \str_set:Nn #2 { \seq_use:Nn #1 / }
341 }
342
343 \cs_new:Nn \stex_path_to_string:N {
344   \seq_use:Nn #1 /
345 }

```

(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`

```

346 \str_const:Nn \c__stex_path_dot_str {.}
347 \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.

```

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

```

```

369         }{
370         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
371         }
372     }
373     }{
374     \str_if_empty:NF \l_tmpa_tl {
375     \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
376     }
377     }
378 }
379 }
380 \seq_gset_eq:NN #1 \l_tmpa_seq
381 }
382 }

```

(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 $\underline{TF}$`

```

383 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
384   \seq_if_empty:NTF #1 {
385     \prg_return_false:
386   }{
387     \seq_get_left:NN #1 \l_tmpa_tl
388     \str_if_empty:NTF \l_tmpa_tl {
389       \prg_return_true:
390     }{
391       \prg_return_false:
392     }
393   }
394 }

```

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

## 26.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

395 \str_new:N\l_stex_kpsewhich_return_str
396 \cs_new_protected:Nn \stex_kpsewhich:n {
397   \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
398   \exp_args:NNo \str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
399   \tl_trim_spaces:N \l_stex_kpsewhich_return_str
400 }

```

(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`

```

401 \sys_if_platform_windows:TF{
402   \stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
403 }{
404   \stex_kpsewhich:n{-var-value~PWD}
405 }
406

```



```

407 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
408 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
409 \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

```

410 <@=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 `STEX`-purposes.

`\g_stex_files_stack` keeps track of file changes

```

411 \seq_gclear_new:N\g_stex_files_stack

```

(End definition for `\g_stex_files_stack`.)

`\c_stex_mainfile_seq`  
`\c_stex_mainfile_str`

```

412 \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
413 \stex_path_from_string:Nn \c_stex_mainfile_seq
414 \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`.

```

415 \seq_gclear_new:N\g_stex_currentfile_seq
416 \AddToHook{file/before}{
417   \stex_path_from_string:Nn\g_stex_currentfile_seq{\CurrentFilePath}
418   \stex_path_if_absolute:NTF\g_stex_currentfile_seq{
419     \exp_args:NNe\seq_put_right:Nn\g_stex_currentfile_seq{\CurrentFile}
420   }{
421     \stex_path_from_string:Nn\g_stex_currentfile_seq{
422       \c_stex_pwd_str/\CurrentFilePath/\CurrentFile
423     }
424   }
425   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
426   \exp_args:NNo\seq_gpush:Nn\g_stex_files_stack\g_stex_currentfile_seq
427 }
428 \AddToHook{file/after}{
429   \seq_if_empty:NF\g_stex_files_stack{
430     \seq_gpop:NN\g_stex_files_stack\l_tmpa_seq
431   }
432   \seq_if_empty:NTF\g_stex_files_stack{
433     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
434   }{
435     \seq_get:NN\g_stex_files_stack\l_tmpa_seq
436     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
437   }
438 }

```

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

## 26.4 MathHub Repositories

```

439 <@@=stex_mathhub>

\mathhub
\c_stex_mathhub_seq
\c_stex_mathhub_str
440 \str_if_empty:NTF\mathhub{
441   \stex_kpsewhich:n{-var-value~MATHHUB}
442   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
443
444   \str_if_empty:NTF\c_stex_mathhub_str{
445     \msg_warning:nn{stex}{warning/nomathhub}
446   }{
447     \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
448     \exp_args:NNx \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
449   }
450 }{
451   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
452   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
453     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
454       \c_stex_pwd_str/\mathhub
455     }
456   }
457   \stex_path_to_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
458   \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
459 }

```

(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
460 \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
461   \str_set:Nx \l_tmpa_str { #1 }
462   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
463     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
464     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
465     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
466     \__stex_mathhub_find_manifest:N \l_tmpa_seq
467     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
468       \msg_error:nnxx{stex}{error/norepository}{#1}{
469         \stex_path_to_string:N \c_stex_mathhub_str
470       }
471     } {
472       \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
473     }
474   }
475 }

```

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

```

\l_stex_mathhub_manifest_file_seq
476 \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:

```

477 \cs_new_protected:Nn \_stex_mathhub_find_manifest:N {
478   \seq_set_eq:NN \l_tmpa_seq #1
479   \bool_set_true:N \l_tmpa_bool
480   \bool_while_do:Nn \l_tmpa_bool {
481     \seq_if_empty:NTF \l_tmpa_seq {
482       \bool_set_false:N \l_tmpa_bool
483     }{
484       \file_if_exist:nTF{
485         \stex_path_to_string:N \l_tmpa_seq/MANIFEST.MF
486       }{
487         \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
488         \bool_set_false:N \l_tmpa_bool
489       }{
490         \file_if_exist:nTF{
491           \stex_path_to_string:N \l_tmpa_seq/META-INF/MANIFEST.MF
492         }{
493           \seq_put_right:Nn \l_tmpa_seq{META-INF}
494           \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
495           \bool_set_false:N \l_tmpa_bool
496         }{
497           \file_if_exist:nTF{
498             \stex_path_to_string:N \l_tmpa_seq/meta-inf/MANIFEST.MF
499           }{
500             \seq_put_right:Nn \l_tmpa_seq{meta-inf}
501             \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
502             \bool_set_false:N \l_tmpa_bool
503           }{
504             \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
505           }
506         }
507       }
508     }
509   }
510   \seq_set_eq:NN \l__stex_mathhub_manifest_file_seq \l_tmpa_seq
511 }

```

(End definition for \\_stex\_mathhub\_find\_manifest:N.)

\c\_\_stex\_mathhub\_manifest\_ior File variable used for MANIFEST-files

```

512 \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:

```

513 \cs_new_protected:Nn \_stex_mathhub_parse_manifest:n {
514   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
515   \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
516   \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
517     \str_set:Nn \l_tmpa_str {##1}
518     \exp_args:NNoo \seq_set_split:Nnn

```

```

519     \l_tmpb_seq \c_colon_str \l_tmpa_str
520 \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
521   \exp_args:NNe \str_set:Nn \l_tmpb_tl {
522     \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
523   }
524   \exp_args:No \str_case:nnTF \l_tmpa_tl {
525     {id} {
526       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
527       { id } \l_tmpb_tl
528     }
529     {narration-base} {
530       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
531       { narr } \l_tmpb_tl
532     }
533     {url-base} {
534       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
535       { docurl } \l_tmpb_tl
536     }
537     {source-base} {
538       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
539       { ns } \l_tmpb_tl
540     }
541     {ns} {
542       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
543       { ns } \l_tmpb_tl
544     }
545     {dependencies} {
546       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
547       { deps } \l_tmpb_tl
548     }
549   }{}{}
550 }{}
551 }
552 \ior_close:N \c__stex_mathhub_manifest_ior
553 }

```

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

`\stex_set_current_repository:n`

```

554 \cs_new_protected:Nn \stex_set_current_repository:n {
555   \stex_require_repository:n { #1 }
556   \prop_set_eq:Nc \l_stex_current_repository_prop {
557     c_stex_mathhub_#1_manifest_prop
558   }
559 }

```

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

`\stex_require_repository:n`

```

560 \cs_new_protected:Nn \stex_require_repository:n {
561   \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
562     \stex_debug:nn{mathhub}{Opening~archive:~#1}
563     \__stex_mathhub_do_manifest:n { #1 }
564     \exp_args:Nx \stex_add_to_sms:n {
565       \prop_const_from_keyval:cn { c_stex_mathhub_#1_manifest_prop } {

```

```

566         id   = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { id } ,
567         ns   = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { ns } ,
568         narr = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { narr } ,
569         deps = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { deps }
570     }
571 }
572 }
573 }

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

574 %\prop_new:N \l_stex_current_repository_prop
575
576 \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
577 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
578   \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
579 } {
580   \__stex_mathhub_parse_manifest:n { main }
581   \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
582   \l_tmpa_str
583   \prop_set_eq:cN { c_stex_mathhub\_l_tmpa_str _manifest_prop }
584   \c_stex_mathhub_main_manifest_prop
585   \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
586   \stex_debug:nn{mathhub}{Current~repository:~
587     \prop_item:Nn \l_stex_current_repository_prop {id}
588   }
589 }

```

(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.

```

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

```

```

611     \prop_if_exist:NTF \l_stex_current_repository_prop {
612       \prop_item:Nn \l_stex_current_repository_prop { id }::~
613       \meaning\l_stex_current_repository_prop
614     }{
615       no~repository
616     }
617   }
618   \prop_if_exist:NTF \l_stex_current_repository_prop {
619     \stex_set_current_repository:n {
620       \prop_item:Nn \l_stex_current_repository_prop { id }
621     }
622   }{
623     \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
624   }
625 }
626 }
627 }

```

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

```

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

```

```

661     \endgroup
662   }
663 }
664 }
665
666 \NewDocumentCommand \inputref { 0{} m}{
667   \stex_inputref:nn{ #1 }{ #2 }
668 }
669
670 \cs_new_protected:Nn \stex_mhbibresource:nn {
671   \stex_in_repository:nn {#1} {
672     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
673   }
674 }
675 \newcommand\addmhbibresource[2][]{
676   \stex_mhbibresource:nn{ #1 }{ #2 }
677 }

```

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

#### `\mhpath`

```

678 \def \mhpath #1 #2 {
679   \exp_args:Ne \str_if_eq:nnTF{#1}{}{
680     \c_stex_mathhub_str /
681     \prop_item:Nn \l_stex_current_repository_prop { id }
682     / source / #2
683   }{
684     \c_stex_mathhub_str / #1 / source / #2
685   }
686 }

```

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

#### `\libinput`

```

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

```

```

707     }
708   }{}
709 }
710 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
711   / \l_tmpa_str / lib / #1.tex
712 }{
713   \bool_set_true:N \l_tmpa_bool
714   \tl_put_right:Nx \l_tmpa_tl {
715     \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
716       / \l_tmpa_str / lib / #1.tex}
717   }
718 }{}
719 \bool_if:NF \l_tmpa_bool {
720   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
721 }
722 \l_tmpa_tl
723 }

```

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

#### `\libusepackage`

```

724 \NewDocumentCommand \libusepackage {0{ } m} {
725   \prop_if_exist:NF \l_stex_current_repository_prop {
726     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
727   }
728   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
729     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
730   }
731   \bool_set_false:N \l_libusepackage_bool
732   \tl_clear:N \l_tmpa_tl
733   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
734   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
735   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str
736   \seq_pop_left:NNT \l_tmpb_seq \l_tmpb_str {
737     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
738     \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
739       / meta-inf / lib / #2.sty}{
740       \bool_set_true:N \l_libusepackage_bool
741       \tl_put_right:Nx \l_tmpa_tl {
742         \exp_not:N \usepackage[#1] { \stex_path_to_string:N \l_tmpa_seq
743           / meta-inf / lib / #2}
744       }
745     }{}
746   }
747   \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
748     / \l_tmpa_str / lib / #2.sty
749   }{
750     \bool_if:NT \l_libusepackage_bool {
751       \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
752     }
753     \bool_set_true:N \l_libusepackage_bool
754     \tl_put_right:Nx \l_tmpa_tl {
755       \exp_not:N \usepackage[#1] { \stex_path_to_string:N \l_tmpa_seq
756         / \l_tmpa_str / lib / #2}

```



```

757     }
758   }{}
759   \bool_if:NF \l_libusepackage_bool {
760     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
761   }
762   \l_tmpa_tl
763 }

```

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

```

764
765 \AddToHook{begindocument}{
766 \ltx@ifpackageloaded{graphicx}{
767   \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
768   \newcommand\mhgraphics[2][]{\%
769     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
770     \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}}
771   \newcommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
772 }{}
773 \ltx@ifpackageloaded{listings}{
774   \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
775   \newcommand\lstinputmhlising[2][]{\%
776     \def\lst@mhrepos{}\setkeys{lst}{#1}%
777     \lstinputlisting[#1]{\mhp\lst@mhrepos{#2}}}
778   \newcommand\clstinputmhlising[2][]{\begin{center}\lstinputmhlising[#1]{#2}\end{center}}
779 }{}
780 }
781
782
783 \</package>

```

## Chapter 27

# STEX -References Implementation

```
784 <*package>
785
786 %%%%%%%%%% references.dtx %%%%%%%%%%
787
788 %\RequirePackage{hyperref}
789 %\RequirePackage{cleveref}
790 <@@=stex_refs>
791
792 Warnings and error messages
793
794 \iow_new:N \c__stex_refs_refs_iow
795 \AddToHook{begindocument}{
796   \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
797 }
798 \AddToHook{enddocument}{
799   \iow_close:N \c__stex_refs_refs_iow
800 }
801
802 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
803
804 \NewDocumentCommand \STEXreftitle { m } {
805   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
806 }
807
```

### 27.1 Document URIs and URLs

```
805 \seq_new:N \g__stex_refs_all_refs_seq
806
807 \str_new:N \l_stex_current_docns_str
808
809 \cs_new_protected:Nn \stex_get_document_uri: {
810   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
811   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
812   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
813   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
814 }
815
```

```

814 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
815
816 \str_clear:N \l_tmpa_str
817 \prop_if_exist:NT \l_stex_current_repository_prop {
818   \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
819     \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
820   }
821 }
822
823 \str_if_empty:NTF \l_tmpa_str {
824   \str_set:Nx \l_stex_current_docns_str {
825     file:/\stex_path_to_string:N \l_tmpa_seq
826   }
827 }{
828   \bool_set_true:N \l_tmpa_bool
829   \bool_while_do:Nn \l_tmpa_bool {
830     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
831     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
832       {source} { \bool_set_false:N \l_tmpa_bool }
833     }{}{
834       \seq_if_empty:NT \l_tmpa_seq {
835         \bool_set_false:N \l_tmpa_bool
836       }
837     }
838   }
839
840   \seq_if_empty:NTF \l_tmpa_seq {
841     \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
842   }{
843     \str_set:Nx \l_stex_current_docns_str {
844       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
845     }
846   }
847 }
848 }
849
850 \str_new:N \l_stex_current_docurl_str
851 \cs_new_protected:Nn \stex_get_document_url: {
852   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
853   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
854   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
855   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
856   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
857
858   \str_clear:N \l_tmpa_str
859   \prop_if_exist:NT \l_stex_current_repository_prop {
860     \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
861       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
862         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
863       }
864     }
865
866     \str_if_empty:NTF \l_tmpa_str {
867       \str_set:Nx \l_stex_current_docurl_str {

```

```

868     file:/\stex_path_to_string:N \l_tmpa_seq
869   }
870 }{
871   \bool_set_true:N \l_tmpa_bool
872   \bool_while_do:Nn \l_tmpa_bool {
873     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
874     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
875       {source} { \bool_set_false:N \l_tmpa_bool }
876     }{}{
877       \seq_if_empty:NT \l_tmpa_seq {
878         \bool_set_false:N \l_tmpa_bool
879       }
880     }
881   }
882
883   \seq_if_empty:NTF \l_tmpa_seq {
884     \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
885   }{
886     \str_set:Nx \l_stex_current_docurl_str {
887       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
888     }
889   }
890 }
891 }

```

## 27.2 Setting Reference Targets

```

892 \str_const:Nn \c__stex_refs_url_str{URL}
893 \str_const:Nn \c__stex_refs_ref_str{REF}
894 % @currentlabel -> number
895 % @currentlabelname -> title
896 % @currentHref -> name.number <- id of some kind
897 % \theH# -> \arabic{section}
898 % \the# -> number
899 % \hyper@makecurrent{#}
900 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
901   \stex_get_document_uri:
902   \str_set:Nx \l_tmpa_str { #1 }
903   \str_if_empty:NT \l_tmpa_str {
904     \int_zero:N \l_tmpa_int
905     \bool_set_true:N \l_tmpa_bool
906     \bool_while_do:Nn \l_tmpa_bool {
907       \cs_if_exist:cTF {
908         sref_\l_stex_current_docns_str?? REF_\int_use:N \l_tmpa_int _type
909       }{
910         \int_incr:N \l_tmpa_int
911       }{
912         \str_set:Nx \l_tmpa_str { REF_\int_use:N \l_tmpa_int }
913         \bool_set_false:N \l_tmpa_bool
914       }
915     }
916   }
917   \str_set:Nx \l_tmpa_str {
918     \l_stex_current_docns_str??\l_tmpa_str

```

```

919 }
920 \seq_gput_right:No \g__stex_refs_all_refs_seq \l_tmpa_str
921 \stex_if_smsmode:TF {
922   \stex_get_document_url:
923   \str_gset_eq:cN {sref_url_\l_tmpa_str _str}\l_stex_current_docurl_str
924   \str_gset_eq:cN {sref_\l_tmpa_str _type}\c__stex_refs_url_str
925 }{
926   \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter\unexpanded\expandafter{
927   \exp_args:Nx\label{sref_\l_tmpa_str}
928   \exp_args:NNNx\immediate\write\@auxout{\stexauxadddocref{\l_tmpa_str}}
929   \str_gset:cx {sref_\l_tmpa_str _type}\c__stex_refs_ref_str
930 }
931 }
932 \cs_new_protected:Npn \stexauxadddocref #1 {
933   \str_set:Nx \l_tmpa_str {#1}
934   \str_gset_eq:cN{sref_\l_tmpa_str _type}\c__stex_refs_ref_str
935   \seq_gput_right:Nx \g__stex_refs_all_refs_seq {\l_tmpa_str}
936 }
937 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
938   \stex_get_document_uri:
939   \stex_if_smsmode:TF {
940     \stex_get_document_url:
941     \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str
942     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
943   }
944   }{
945     \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter{\@currentlabel\iffalse}}
946     \exp_args:Nx\label{sref_sym_#1}
947   }
948   \exp_args:NNNx\immediate\write\@auxout{\stexauxadddocref{sym_#1}}
949   \str_gset:cx {sref_sym_#1_type}\c__stex_refs_ref_str
950 }
951 }

```

## 27.3 Using References

```

952 \str_new:N \l__stex_refs_indocument_str
953 \keys_define:nn { stex / sref } {
954   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
955   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
956   pre           .tl_set:N = \l__stex_refs_pre_tl ,
957   post          .tl_set:N = \l__stex_refs_post_tl ,
958   %indoc        .str_set_x:N = \l__stex_refs_repo_str ,
959 }
960
961 \bool_new:N \c__stex_refs_hyperref_bool
962 \bool_set_false:N \c__stex_refs_hyperref_bool
963 \AddToHook{begindocument}{
964   \@ifpackageloaded{hyperref}{
965     \bool_set_true:N \c__stex_refs_hyperref_bool
966   }{}
967 }
968
969

```

```

970 \cs_new_protected:Nn \__stex_refs_args:n {
971   \tl_clear:N \l__stex_refs_linktext_tl
972   \tl_clear:N \l__stex_refs_fallback_tl
973   \tl_clear:N \l__stex_refs_pre_tl
974   \tl_clear:N \l__stex_refs_post_tl
975   \str_clear:N \l__stex_refs_repo_str
976   \keys_set:nn { stex / sref } { #1 }
977 }
978
979 \NewDocumentCommand \sref { 0{} m}{
980   \__stex_refs_args:n { #1 }
981   \str_if_empty:NTF \l__stex_refs_indocument_str {
982     \str_set:Nn \l_tmpa_str { #2 }
983     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
984     \tl_set:Nn \l_tmpa_tl {
985       \l__stex_refs_fallback_tl
986     }
987     \seq_map_inline:Nn \g__stex_refs_all_refs_seq {
988       \str_set:Nn \l_tmpb_str { ##1 }
989       \str_if_eq:eeT { \l_tmpa_str } {
990         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int }{-1 }
991       } {
992         \seq_map_break:n {
993           \tl_set:Nn \l_tmpa_tl {
994             % doc uri in \l_tmpb_str
995             \str_set:Nx \l_tmpa_str {\use:c{sref_\l_tmpb_str _type}}
996             \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
997               % reference
998               \cs_if_exist:cTF{autoref}{
999                 \l__stex_refs_pre_tl\autoref{sref_\l_tmpb_str}\l__stex_refs_post_tl
1000               }{
1001                 \l__stex_refs_pre_tl\ref{sref_\l_tmpb_str}\l__stex_refs_post_tl
1002               }
1003             }{
1004               % URL
1005               \if_bool:N \c__stex_refs_hyperref_bool {
1006                 \exp_args:Nx \href{\use:c{sref_url_\l_tmpb_str _str}}{\l__stex_refs_fallback
1007               }{
1008                 \l__stex_refs_fallback_tl
1009               }
1010             }
1011           }
1012         }
1013       }
1014     }
1015     \l_tmpa_tl
1016   }{
1017     % TODO
1018   }
1019 }
1020
1021 \NewDocumentCommand \srefsym { 0{} m}{
1022   \stex_get_symbol:n { #2 }
1023   \__stex_refs_args:n { #1 }

```

```

1024 \str_if_empty:NTF \l__stex_refs_indocument_str {
1025   \tl_set:Nn \l_tmpa_tl {
1026     \l__stex_refs_fallback_tl
1027   }
1028   \tl_if_exist:cT{sref_sym_\l_stex_get_symbol_uri_str_type}{
1029     \tl_set:Nn \l_tmpa_tl {
1030       % doc uri in \l_tmpb_str
1031       \str_set:Nx \l_tmpa_str {\use:c{sref_sym_\l_stex_get_symbol_uri_str_type}}
1032       \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1033         % reference
1034         \cs_if_exist:cTF{autoref}{
1035           \l__stex_refs_pre_tl\autoref{sref_sym_\l_stex_get_symbol_uri_str}\l__stex_refs_post_tl
1036         }{
1037           \l__stex_refs_pre_tl\ref{sref_sym_\l_stex_get_symbol_uri_str}\l__stex_refs_post_tl
1038         }
1039       }{
1040         % URL
1041         \if_bool:N \c__stex_refs_hyperref_bool {
1042           \exp_args:Nx \href{\use:c{sref_sym_url_\l_stex_get_symbol_uri_str_str}}{\l__stex_refs_ref_str}
1043         }{
1044           \l__stex_refs_fallback_tl
1045         }
1046       }
1047     }
1048   }
1049   \l_tmpa_tl
1050 }{
1051   % TODO
1052 }
1053 }
1054
1055 \cs_new_protected:Npn \srefsymuri #1 #2 {
1056   \hyperref[sref_sym_#1]{#2}
1057 }
1058
1059 </package>

```

## Chapter 28

# STEX -Modules Implementation

```
1060 <*package>
1061
1062 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1063
1064 <@@=stex_modules>
1065
1066   Warnings and error messages
1067   \msg_new:nnn{stex}{error/unknownmodule}{
1068     No~module~#1~found
1069   }
1070   \msg_new:nnn{stex}{error/syntax}{
1071     Syntax~error:~#1
1072   }
1073   \msg_new:nnn{stex}{error/siglanguage}{
1074     Module~#1~declares~signature~#2,~but~does~not~
1075     declare~its~language
1076   }
1077   \msg_new:nnn{stex}{error/conclictingmodules}{
1078     Comflicting~imports~for~module~#1
1079   }
1080
1081 \l_stex_current_module_str The current module:
1082 \str_new:N \l_stex_current_module_str
1083
1084 (End definition for \l_stex_current_module_str. This variable is documented on page 26.)
1085
1086 \l_stex_all_modules_seq Stores all available modules
1087 \seq_new:N \l_stex_all_modules_seq
1088
1089 (End definition for \l_stex_all_modules_seq. This variable is documented on page 26.)
1090
1091 \stex_if_in_module_p:
1092 \stex_if_in_module:TF
1093 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1094   \str_if_empty:NTF \l_stex_current_module_str
1095     \prg_return_false: \prg_return_true:
1096 }
```



(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`

```
1085 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1086   \prop_if_exist:cTF { c_stex_module_#1_prop }
1087   \prg_return_true: \prg_return_false:
1088 }
```

(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:

```
1089 \cs_new_protected:Nn \stex_add_to_current_module:n {
1090   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str_code} { #1 }
1091 }
1092 \cs_new_protected:Npn \STEXexport {
1093   \begingroup
1094   \newlinechar=-1\relax
1095   \endlinechar=-1\relax
1096   %\catcode'\ = 9\relax
1097   \expandafter\endgroup\STEXexport:n
1098 }
1099 \cs_new_protected:Nn \STEXexport:n {
1100   \ignorespaces #1
1101   \stex_add_to_current_module:n { \ignorespaces #1 }
1102   \stex_smsmode_set_codes:
1103 }
1104 \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`

```
1105 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1106   \str_set:Nx \l_tmpa_str { #1 }
1107   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str_constants} { \l_tmpa_str }
1108 }
1109
1110 %\cs_new_protected:Nn \stex_add_field_to_current_module:n {
1111 % \str_set:Nx \l_tmpa_str { #1 }
1112 % \seq_gput_right:co {c_stex_module_\l_stex_current_module_str_fields} { \l_tmpa_str }
1113 %}
```

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

`\stex_collect_imports:n`

```
1114 \cs_new_protected:Nn \stex_collect_imports:n {
1115   \seq_clear:N \l_stex_collect_imports_seq
1116   \__stex_modules_collect_imports:n {#1}
1117 }
1118 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1119   \seq_map_inline:cn {c_stex_module_#1_imports} {
1120     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1121       \__stex_modules_collect_imports:n { ##1 }
1122     }
1123 }
```

```

1123 }
1124 \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1125   \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1126 }
1127 }

```

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

`\stex_add_import_to_current_module:n`

```

1128 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1129   \str_set:Nx \l_tmpa_str { #1 }
1130   \exp_args:Nno
1131   \seq_if_in:cnF{c_stex_module\l_stex_current_module_str_imports}\l_tmpa_str{
1132     \seq_gput_right:co{c_stex_module\l_stex_current_module_str_imports}\l_tmpa_str
1133   }
1134 }

```

(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).

```

1135 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
1136   \str_set:Nx \l_tmpa_str { #1 }
1137   \seq_set_eq:NN \l_tmpa_seq #2
1138   % split off file extension
1139   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1140   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1141   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1142   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1143
1144   \bool_set_true:N \l_tmpa_bool
1145   \bool_while_do:Nn \l_tmpa_bool {
1146     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1147     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1148       {source} { \bool_set_false:N \l_tmpa_bool }
1149     }{}{
1150       \seq_if_empty:NT \l_tmpa_seq {
1151         \bool_set_false:N \l_tmpa_bool
1152       }
1153     }
1154   }
1155
1156   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
1157   \str_if_empty:NTF \l_stex_modules_subpath_str {
1158     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
1159   }{
1160     \str_set:Nx \l_stex_modules_ns_str {
1161       \l_tmpa_str/\l_stex_modules_subpath_str
1162     }
1163   }
1164 }

```

(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
1165 \str_new:N \l_stex_modules_ns_str
1166 \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.

```

1167 \cs_new_protected:Nn \stex_modules_current_namespace: {
1168   \str_clear:N \l_stex_modules_subpath_str
1169   \prop_if_exist:NTF \l_stex_current_repository_prop {
1170     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1171     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1172   }{
1173     % split off file extension
1174     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1175     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1176     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1177     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1178     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1179     \str_set:Nx \l_stex_modules_ns_str {
1180       file:/\stex_path_to_string:N \l_tmpa_seq
1181     }
1182   }
1183 }

```

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

## 28.1 The module environment

module arguments:

```

1184 \keys_define:nn { stex / module } {
1185   title      .str_set_x:N = \l_stex_module_title_str ,
1186   ns         .str_set_x:N = \l_stex_module_ns_str ,
1187   lang       .str_set_x:N = \l_stex_module_lang_str ,
1188   sig        .str_set_x:N = \l_stex_module_sig_str ,
1189   creators   .str_set_x:N = \l_stex_module_creators_str ,
1190   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1191   meta       .str_set_x:N = \l_stex_module_meta_str ,
1192   srccite    .str_set_x:N = \l_stex_module_srccite_str
1193 }
1194
1195 \cs_new_protected:Nn \__stex_modules_args:n {
1196   \str_clear:N \l_stex_module_title_str
1197   \str_clear:N \l_stex_module_ns_str
1198   \str_clear:N \l_stex_module_lang_str
1199   \str_clear:N \l_stex_module_sig_str
1200   \str_clear:N \l_stex_module_creators_str
1201   \str_clear:N \l_stex_module_contributors_str
1202   \str_clear:N \l_stex_module_meta_str
1203   \str_clear:N \l_stex_module_srccite_str
1204   \keys_set:nn { stex / module } { #1 }

```

```

1205 }
1206
1207 % module parameters here? In the body?
1208

```

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

```

1209 \cs_new_protected:Nn \stex_module_setup:nn {
1210   \str_set:Nx \l_stex_module_name_str { #2 }
1211   \__stex_modules_args:n { #1 }

```

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

Are we in a nested module?

```

1212 \stex_if_in_module:TF {
1213   % Nested module
1214   \prop_get:cnN {c_stex_module\_l_stex_current_module_str _prop}
1215   { ns } \l_stex_module_ns_str
1216   \str_set:Nx \l_stex_module_name_str {
1217     \prop_item:cn {c_stex_module\_l_stex_current_module_str _prop}
1218     { name } / \l_stex_module_name_str
1219   }
1220 }{
1221   % not nested:
1222   \str_if_empty:NT \l_stex_module_ns_str {
1223     \stex_modules_current_namespace:
1224     \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1225     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1226     / { \l_stex_module_ns_str }
1227     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1228     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1229       \str_set:Nx \l_stex_module_ns_str {
1230         \stex_path_to_string:N \l_tmpa_seq
1231       }
1232     }
1233   }
1234 }

```

Next, we determine the language of the module:

```

1235 \str_if_empty:NT \l_stex_module_lang_str {
1236   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1237   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1238   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1239   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1240   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1241     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1242       inferred~from~file~name}
1243     \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1244   }
1245 }
1246
1247 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
1248   \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1249   \l_tmpa_str {
1250     \ltx@ifpackageloaded{babel}{
1251       \exp_args:Nx \selectlanguage { \l_tmpa_str }

```

```

1252     }{}
1253   } {
1254     \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
1255   }
1256 }}

```

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

```

1257 \str_if_empty:NTF \l_stex_module_sig_str {
1258   \exp_args:Nnx \prop_gset_from_keyval:cn {
1259     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1260   } {
1261     name      = \l_stex_module_name_str ,
1262     ns        = \l_stex_module_ns_str ,
1263     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1264     lang      = \l_stex_module_lang_str ,
1265     sig       = \l_stex_module_sig_str ,
1266     meta      = \l_stex_module_meta_str
1267   }
1268   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1269   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _fields}
1270   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1271   \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1272   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1273 \str_if_empty:NT \l_stex_module_meta_str {
1274   \str_set:Nx \l_stex_module_meta_str {
1275     \c_stex_metatheory_ns_str ? Metatheory
1276   }
1277 }
1278 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1279   \bool_set_true:N \l_stex_in_meta_bool
1280   \exp_args:Nx \stex_add_to_current_module:n {
1281     \bool_set_true:N \l_stex_in_meta_bool
1282     \stex_activate_module:n {\l_stex_module_meta_str}
1283     \bool_set_false:N \l_stex_in_meta_bool
1284   }
1285   \stex_activate_module:n {\l_stex_module_meta_str}
1286   \bool_set_false:N \l_stex_in_meta_bool
1287 }
1288 }{
1289   \str_if_empty:NT \l_stex_module_lang_str {
1290     \msg_error:nnxx{stex}{error/siglanguage}{
1291       \l_stex_module_ns_str?\l_stex_module_name_str
1292     }{\l_stex_module_sig_str}
1293   }
1294
1295   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1296   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1297   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1298   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1299   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1300   \str_set:Nx \l_tmpa_str {

```

```

1301     \stex_path_to_string:N \l_tmpa_seq /
1302     \l_tmpa_str . \l_stex_module_sig_str .tex
1303   }
1304   \IfFileExists \l_tmpa_str {
1305     \exp_args:No \stex_in_smsmode:nn { \l_tmpa_str } {
1306       \str_clear:N \l_stex_current_module_str
1307       \seq_clear:N \l_stex_all_modules_seq
1308       \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1309       \input { \l_tmpa_str }
1310     }
1311   }{
1312     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1313   }
1314   \stex_if_smsmode:F {
1315     \stex_activate_module:n {
1316       \l_stex_module_ns_str ? \l_stex_module_name_str
1317     }
1318   }
1319   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1320 }
1321 }

```

(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}
1322 \int_new:N \l_stex_module_group_depth_int
1323 \cs_new_protected:Nn \__stex_modules_begin_module:nn {
1324   \stex_reactivate_macro:N \STEXexport
1325   \stex_reactivate_macro:N \importmodule
1326   \stex_reactivate_macro:N \symdecl
1327   \stex_reactivate_macro:N \notation
1328   \stex_reactivate_macro:N \symdef
1329   \stex_module_setup:nn{#1}{#2}
1330
1331   \stex_debug:nn{modules}{
1332     New~module:\\
1333     Namespace:~\l_stex_module_ns_str\\
1334     Name:~\l_stex_module_name_str\\
1335     Language:~\l_stex_module_lang_str\\
1336     Signature:~\l_stex_module_sig_str\\
1337     Metatheory:~\l_stex_module_meta_str\\
1338     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1339   }
1340
1341   \seq_put_right:Nx \l_stex_all_modules_seq {
1342     \l_stex_module_ns_str ? \l_stex_module_name_str
1343   }
1344
1345   % \seq_gput_right:Nx \g_stex_modules_in_file_seq
1346   % { \l_stex_module_ns_str ? \l_stex_module_name_str }
1347
1348

```

```

1349 \stex_if_smsmode:TF {
1350   \stex_smsmode_set_codes:
1351 } {
1352   \begin{stex_annotate_env} {theory} {
1353     \l_stex_module_ns_str ? \l_stex_module_name_str
1354   }
1355
1356   \stex_annotate_invisible:nnn{header}{} {
1357     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1358     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1359     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1360       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1361     }
1362   }
1363 }
1364 \int_set:Nn \l_stex_module_group_depth_int {\currentgrouplevel}
1365 % TODO: Inherit metatheory for nested modules?
1366 }
1367 \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}`

```

1368 \cs_new_protected:Nn \_stex_modules_end_module: {
1369 % \str_set:Nx \l_tmpa_str {
1370 %   c_stex_module_
1371 %   \prop_item:Nn \l_stex_current_module_prop { ns } ?
1372 %   \prop_item:Nn \l_stex_current_module_prop { name }
1373 %   _prop
1374 % }
1375 %^^A \prop_new:c { \l_tmpa_str }
1376 % \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
1377 \stex_debug:nn{modules}{Closing-module~\prop_item:cn {c_stex_module_\l_stex_current_module_
1378 }

```

(End definition for `\_stex_modules_end_module:.`)

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

```

1379 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1380 \NewDocumentEnvironment { @module } { 0{} m } {
1381   \par
1382   \_stex_modules_begin_module:nn{#1}{#2}
1383 } {
1384   \_stex_modules_end_module:
1385   \stex_if_smsmode:TF {
1386 %     \exp_args:Nx \stex_add_to_sms:n {
1387 %       \prop_gset_from_keyval:cn {
1388 %         c_stex_module_
1389 %         \prop_item:Nn \l_stex_current_module_prop { ns } ?
1390 %         \prop_item:Nn \l_stex_current_module_prop { name }
1391 %         _prop
1392 %       } {
1393 %         name      = \prop_item:cn { \l_tmpa_str } { name } ,
1394 %         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,

```

```

1395 %         file      = \prop_item:cn { \l_tmpa_str } { file } ,
1396 %         lang       = \prop_item:cn { \l_tmpa_str } { lang } ,
1397 %         sig        = \prop_item:cn { \l_tmpa_str } { sig } ,
1398 %         meta       = \prop_item:cn { \l_tmpa_str } { meta }
1399 %     }
1400 % }
1401 }{
1402   \end{stex_annotate_env}
1403 }
1404 }

```

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

```

1405 \cs_if_exist:NTF \thesection {
1406   \newcounter{module}[section]
1407 }{
1408   \newcounter{module}
1409 }
1410
1411 \bool_if:NT \c_stex_showmods_bool {
1412   \latexml_if:F { \RequirePackage{mdframed} }
1413 }
1414
1415 \cs_new_protected:Nn \stex_modules_heading: {
1416   \stepcounter{module}
1417   \par
1418   \bool_if:NT \c_stex_showmods_bool {
1419     \noindent{\textbf{Module} ~
1420       \cs_if_exist:NT \thesection {\thesection.}
1421       \themodule ~ [\l_stex_module_name_str]
1422     }
1423     \str_if_empty:NTF \l_stex_module_title_str {
1424       }{
1425         \quad(\l_stex_module_title_str)\hfill
1426       }\par
1427     }
1428     \edef\@currentlabel{Module~\thesection.\themodule~[\l_stex_module_name_str]}
1429     % TODO
1430     \stex_ref_new_doc_target:n \l_stex_module_name_str
1431   }

```

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

Finally:

```

1432 \NewDocumentEnvironment { module } { 0 } { m } {
1433   \bool_if:NT \c_stex_showmods_bool {
1434     \begin{mdframed}
1435   }
1436   \begin{@module}[#1]{#2}
1437     \stex_modules_heading:
1438   }{
1439     \end{@module}
1440     \bool_if:NT \c_stex_showmods_bool {
1441       \end{mdframed}
1442     }
1443   }

```



## 28.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n 1444 \NewDocumentCommand \STEXModule { m } {
1445   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1446   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1447   \tl_set:Nn \l_tmpa_tl {
1448     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1449   }
1450   \seq_map_inline:Nn \l_stex_all_modules_seq {
1451     \str_set:Nn \l_tmpb_str { ##1 }
1452     \str_if_eq:eeT { \l_tmpa_str } {
1453       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1454     } {
1455       \seq_map_break:n {
1456         \tl_set:Nn \l_tmpa_tl {
1457           \stex_invoke_module:n { ##1 }
1458         }
1459       }
1460     }
1461   }
1462   \l_tmpa_tl
1463 }
1464
1465 \cs_new_protected:Nn \stex_invoke_module:n {
1466   \stex_debug:nn{modules}{Invoking~module~#1}
1467   \peek_charcode_remove:NTF ! {
1468     \__stex_modules_invoke_uri:nN { #1 }
1469   } {
1470     \peek_charcode_remove:NTF ? {
1471       \__stex_modules_invoke_symbol:nn { #1 }
1472     } {
1473       \msg_error:nnx{stex}{error/syntax}{
1474         ?~or~!~expected~after~
1475         \c_backslash_str STEXModule{#1}
1476       }
1477     }
1478   }
1479 }
1480
1481 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1482   \str_set:Nn #2 { #1 }
1483 }
1484
1485 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1486   \stex_invoke_symbol:n{#1?#2}
1487 }

```

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

```

\stex_activate_module:n
1488 \bool_new:N \l_stex_in_meta_bool
1489 \bool_set_false:N \l_stex_in_meta_bool

```

```

1490 \cs_new_protected:Nn \stex_activate_module:n {
1491   \stex_debug:nn{modules}{Activating~module~#1}
1492   \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1493     \msg_error:nnn{stex}{error/concllictingmodules}{ #1 }
1494   }
1495   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1496     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1497     \use:c{ c_stex_module_#1_code }
1498   }
1499 }

```

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

```

1500 </package>

```

## Chapter 29

# STEX -Module Inheritance Implementation

```
1501 <*package>
1502
1503 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1504
```

### 29.1 SMS Mode

```
1505 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1506 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1507 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1508 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1509
1510 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1511   \makeatletter
1512   \makeatother
1513   \ExplSyntaxOn
1514   \ExplSyntaxOff
1515   \rustexBREAK
1516 }
1517
1518 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1519   \symdef
1520   \importmodule
1521   \notation
1522   \symdecl
1523   \STEXexport
1524   \inlineass
1525   \inlinedef
1526   \inlineex
1527 }
1528
```

```

1529 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1530   \tl_to_str:n {
1531     module,
1532     @module,
1533     sdefinition,
1534     sexample,
1535     sassertion,
1536     sparagraph
1537   }
1538 }

```

(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`

```

1539 \bool_new:N \g__stex_smsmode_bool
1540 \bool_set_false:N \g__stex_smsmode_bool
1541 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1542   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1543 }

```

(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`

```

1544 \bool_new:N \g__stex_smsmode_catcode_bool
1545 \bool_set_false:N \g__stex_smsmode_catcode_bool
1546 \prg_new_conditional:Nnn \_stex_smsmode_if_catcodes: { p, T, F, TF } {
1547   \bool_if:NTF \g__stex_smsmode_catcode_bool
1548     \prg_return_true: \prg_return_false:
1549 }

```

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

`\stex_smsmode_set_codes:`

```

1550 \cs_new_protected:Nn \stex_smsmode_set_codes: {
1551   \stex_if_smsmode:T {
1552     \_stex_smsmode_if_catcodes:F {
1553       \bool_gset_true:N \g__stex_smsmode_catcode_bool
1554       \exp_after:wN \char_gset_active_eq:NN
1555       \c_backslash_str \_stex_smsmode_cs:
1556       \tex_global:D \char_set_catcode_active:N \
1557       \tex_global:D \char_set_catcode_other:N $
1558       \tex_global:D \char_set_catcode_other:N ^
1559       \tex_global:D \char_set_catcode_other:N _
1560       \tex_global:D \char_set_catcode_other:N &
1561       \tex_global:D \char_set_catcode_other:N ##
1562     }
1563   }
1564 } \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.

```

1565 \cs_new_protected:Nn \_stex_smsmode_unset_codes: {
1566   \_stex_smsmode_if_catcodes:T {
1567     \bool_gset_false:N \g__stex_smsmode_catcode_bool
1568     \exp_after:wN \tex_global:D \exp_after:wN
1569     \char_set_catcode_escape:N \c_backslash_str
1570     \tex_global:D \char_set_catcode_math_toggle:N $
1571     \tex_global:D \char_set_catcode_math_superscript:N ^
1572     \tex_global:D \char_set_catcode_math_subscript:N _
1573     \tex_global:D \char_set_catcode_alignment:N &
1574     \tex_global:D \char_set_catcode_parameter:N ##
1575   }
1576 } \iffalse $ \fi % to make syntax highlighting work again

```

(End definition for `\_stex_smsmode_unset_codes:`.)

`\stex_in_smsmode:nn`

```

1577 \cs_new_protected:Nn \stex_in_smsmode:nn {
1578   \vbox_set:Nn \l_tmpa_box {
1579     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1580     \bool_gset_true:N \g__stex_smsmode_bool
1581     \stex_smsmode_set_codes:
1582     #2
1583     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1584     \stex_if_smsmode:F {
1585       \_stex_smsmode_unset_codes:
1586     }
1587   }
1588   \box_clear:N \l_tmpa_box
1589 }

```

(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:

```

1590 \cs_new_protected:Nn \_stex_smsmode_cs: {
1591   \str_clear:N \l_tmpa_str
1592   \peek_analysis_map_inline:n {
1593     % #1: token (one expansion)
1594     % #2: charcode
1595     % #3 catcode
1596     \token_if_eq_charcode:NNTF ##3 B {
1597       % token is a letter
1598       \exp_args:NNo \str_put_right:Nn \l_tmpa_str { ##1 }
1599     } {
1600       \str_if_empty:NTF \l_tmpa_str {
1601         % we don't allow (or need) single non-letter CSs
1602         % for now
1603         \peek_analysis_map_break:
1604       }{
1605         \str_if_eq:onTF \l_tmpa_str { begin } {
1606           \peek_analysis_map_break:n {
1607             \exp_after:wN \_stex_smsmode_checkbegin:n ##1
1608           }
1609         }
1610       }
1611     }
1612   }
1613 }

```

```

1609     } {
1610         \str_if_eq:onTF \l_tmpa_str { end } {
1611             \peek_analysis_map_break:n {
1612                 \exp_after:wN \__stex_smsmode_checkend:n ##1
1613             }
1614         } {
1615             \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
1616             \exp_args:NNNo \exp_args:NNo \tl_if_in:NnTF
1617                 \g_stex_smsmode_allowedmacros_tl
1618                 { \use:c{\l_tmpa_str} } {
1619                 \stex_debug:nn{modules}{Executing~1:~\l_tmpa_str}
1620                 \peek_analysis_map_break:n {
1621                     \exp_after:wN \l_tmpa_tl ##1
1622                 }
1623             } {
1624                 \exp_args:NNNo \exp_args:NNo \tl_if_in:NnTF
1625                 \g_stex_smsmode_allowedmacros_escape_tl
1626                 { \use:c{\l_tmpa_str} } {
1627                     \__stex_smsmode_unset_codes:
1628                     \stex_debug:nn{modules}{Executing~2:~\l_tmpa_str}
1629                     % TODO \__stex_smsmode_rescan_cs:
1630                     \int_compare:nNnTF {##2} = {92} {
1631                         \peek_analysis_map_break:n {
1632                             \__stex_smsmode_unset_codes:
1633                             \__stex_smsmode_rescan_cs:
1634                         }
1635                     } {
1636                         \peek_analysis_map_break:n {
1637                             \exp_after:wN \l_tmpa_tl ##1
1638                         }
1639                     }
1640                 } {
1641                     \int_compare:nNnTF {##2} = {92} {
1642                         \peek_analysis_map_break:n { \__stex_smsmode_cs: }
1643                     } {
1644                         \peek_analysis_map_break:n { \exp_after:wN\relax ##1 }
1645                     }
1646                 }
1647             }
1648         }
1649     }
1650 }
1651 }
1652 }
1653 }

```

(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:

```

1654 \cs_new_protected:Nn \__stex_smsmode_rescan_cs: {
1655     \str_clear:N \l_tmppb_str
1656     \peek_analysis_map_inline:n {
1657         \token_if_eq_charcode:NNTF ##3 B {

```

```

1658      % token is a letter
1659      \exp_args:NNo \str_put_right:Nn \l_tmpb_str { ##1 }
1660    } {
1661      \peek_analysis_map_break:n {
1662        \exp_after:wN \use:c \exp_after:wN {
1663          \exp_after:wN \l_tmpa_str\exp_after:wN
1664        } \use:c { \l_tmpb_str \exp_after:wN } ##1
1665      }
1666    }
1667  }
1668 }

```

(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.

```

1669 \cs_new_protected:Nn \__stex_smsmode_checkbegin:n {
1670   \str_set:Nn \l_tmpa_str { #1 }
1671   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1672     \__stex_smsmode_unset_codes:
1673     \begin{#1}
1674   }
1675 }

```

(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.

```

1676 \cs_new_protected:Nn \__stex_smsmode_checkend:n {
1677   \str_set:Nn \l_tmpa_str { #1 }
1678   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1679     \end{#1}
1680   }
1681 }

```

(End definition for `\__stex_smsmode_checkend:n`)

## 29.2 Inheritance

```

1682 <@@=stex_importmodule>

```

`\stex_import_module_uri:nn`

```

1683 \cs_new_protected:Nn \stex_import_module_uri:nn {
1684   \str_set:Nx \l_stex_import_archive_str { #1 }
1685   \str_set:Nn \l_stex_import_path_str { #2 }
1686
1687   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1688   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1689   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1690
1691   \stex_modules_current_namespace:
1692   \bool_lazy_all:nTF {
1693     {\str_if_empty_p:N \l_stex_import_archive_str}
1694     {\str_if_empty_p:N \l_stex_import_path_str}
1695     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1696   }{

```

```

1697 \str_set_eq:NN \l_stex_import_path_str \l_stex_modules_subpath_str
1698 \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1699 }{
1700 \str_if_empty:NT \l_stex_import_archive_str {
1701 \prop_if_exist:NT \l_stex_current_repository_prop {
1702 \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1703 }
1704 }
1705 \str_if_empty:NTF \l_stex_import_archive_str {
1706 \str_if_empty:NF \l_stex_import_path_str {
1707 \str_set:Nx \l_stex_import_ns_str {
1708 \l_stex_module_ns_str / \l_stex_import_path_str
1709 }
1710 }
1711 }{
1712 \stex_require_repository:n \l_stex_import_archive_str
1713 \prop_get:cnN { c_stex_mathhub\_l_stex_import_archive_str_manifest_prop } { ns }
1714 \l_stex_import_ns_str
1715 \str_if_empty:NF \l_stex_import_path_str {
1716 \str_set:Nx \l_stex_import_ns_str {
1717 \l_stex_import_ns_str / \l_stex_import_path_str
1718 }
1719 }
1720 }
1721 }
1722 }

```

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

```

\l_stex_import_name_str Store the return values of \stex_import_module_uri:nn.
\l_stex_import_archive_str 1723 \str_new:N \l_stex_import_name_str
\l_stex_import_path_str 1724 \str_new:N \l_stex_import_archive_str
\l_stex_import_ns_str 1725 \str_new:N \l_stex_import_path_str
1726 \str_new:N \l_stex_import_ns_str

```

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

```

\stex_import_require_module:nnnn {<ns>} {<archive-ID>} {<path>} {<name>}
1727 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1728 \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1729
1730 % archive
1731 \str_set:Nx \l_tmpa_str { #2 }
1732 \str_if_empty:NTF \l_tmpa_str {
1733 \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1734 } {
1735 \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1736 \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1737 \seq_put_right:Nn \l_tmpa_seq { source }
1738 }
1739
1740 % path
1741 \str_set:Nx \l_tmpb_str { #3 }
1742 \str_if_empty:NTF \l_tmpb_str {

```



```

1743 \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1744
1745 \ltx@ifpackageloaded{babel} {
1746   \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1747     { \language } \l_tmpb_str {
1748     \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1749   }
1750 } {
1751   \str_clear:N \l_tmpb_str
1752 }
1753
1754 \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1755 \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1756   \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1757 }{
1758   \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1759   \IfFileExists{ \l_tmpa_str.tex }{
1760     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1761   }{
1762     % try english as default
1763     \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1764     \IfFileExists{ \l_tmpa_str.en.tex }{
1765       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1766     }{
1767       \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1768     }
1769   }
1770 }
1771
1772 } {
1773   \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1774   \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1775
1776   \ltx@ifpackageloaded{babel} {
1777     \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1778       { \language } \l_tmpb_str {
1779       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1780     }
1781   } {
1782     \str_clear:N \l_tmpb_str
1783   }
1784
1785   \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1786
1787   \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1788   \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1789     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1790   }{
1791     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1792     \IfFileExists{ \l_tmpa_str/#4.tex }{
1793       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1794     }{
1795       % try english as default
1796       \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}

```

```

1797 \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1798   \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1799 }{
1800   \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1801   \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1802     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1803   }{
1804     \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1805     \IfFileExists{ \l_tmpa_str.tex }{
1806       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1807     }{
1808       % try english as default
1809       \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1810       \IfFileExists{ \l_tmpa_str.en.tex }{
1811         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1812       }{
1813         \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1814       }
1815     }
1816   }
1817 }
1818 }
1819 }
1820 }
1821
1822 \exp_args:No \stex_in_smsmode:nn { \g__stex_importmodule_file_str } {
1823   \seq_clear:N \l_stex_all_modules_seq
1824   \str_clear:N \l_stex_current_module_str
1825   \str_set:Nx \l_tmpb_str { #2 }
1826   \str_if_empty:NF \l_tmpb_str {
1827     \stex_set_current_repository:n { #2 }
1828   }
1829   \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1830   \input { \g__stex_importmodule_file_str }
1831 }
1832
1833 \stex_if_module_exists:nF { #1 ? #4 } {
1834   \msg_error:nnx{stex}{error/unknownmodule}{
1835     #1?#4~(in~file~\g__stex_importmodule_file_str)
1836   }
1837 }
1838 }
1839 \stex_activate_module:n { #1 ? #4 }
1840 }

```

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

## `\importmodule`

```

1841 \NewDocumentCommand \importmodule { 0{} m } {
1842   \stex_import_module_uri:nn { #1 } { #2 }
1843   \stex_debug:nn{modules}{Importing~module:~
1844     \l_stex_import_ns_str ? \l_stex_import_name_str
1845   }
1846   \stex_if_smsmode:F {

```

```

1847 \stex_import_require_module:nnnn
1848 { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1849 { \l_stex_import_path_str } { \l_stex_import_name_str }
1850 \stex_annotate_invisible:nnn
1851 {import} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1852 }
1853 \exp_args:Nx \stex_add_to_current_module:n {
1854 \stex_import_require_module:nnnn
1855 { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1856 { \l_stex_import_path_str } { \l_stex_import_name_str }
1857 }
1858 \exp_args:Nx \stex_add_import_to_current_module:n {
1859 \l_stex_import_ns_str ? \l_stex_import_name_str
1860 }
1861 \stex_smsmode_set_codes:
1862 }
1863 \stex_deactivate_macro:Nn \importmodule {module-environments}

```

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

## `\usemodule`

```

1864 \NewDocumentCommand \usemodule { 0{} m } {
1865 \stex_if_smsmode:F {
1866 \stex_import_module_uri:nn { #1 } { #2 }
1867 \stex_import_require_module:nnnn
1868 { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1869 { \l_stex_import_path_str } { \l_stex_import_name_str }
1870 \stex_annotate_invisible:nnn
1871 {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1872 }
1873 \stex_smsmode_set_codes:
1874 }

```

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

```

1875 \endpackage

```

## Chapter 30

# STEX -Symbols Implementation

```
1876 <*package>
1877
1878 %%%%%%%%%% symbols.dtx %%%%%%%%%%
1879
Warnings and error messages
1880
```

### 30.1 Symbol Declarations

```
1881 <@@=stex_symdecl>
\l_stex_all_symbols_seq Stores all available symbols
1882 \seq_new:N \l_stex_all_symbols_seq
(End definition for \l_stex_all_symbols_seq. This variable is documented on page 36.)

\STEXsymbol
1883 \NewDocumentCommand \STEXsymbol { m } {
1884   \stex_get_symbol:n { #1 }
1885   \exp_args:No
1886   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1887 }
(End definition for \STEXsymbol. This function is documented on page 38.)
symdecl arguments:
1888 \keys_define:nn { stex / symdecl } {
1889   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1890   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1891   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1892   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
1893   align     .str_set:N    = \l_stex_symdecl_align_str , % TODO(?)
1894   gfc       .str_set:N    = \l_stex_symdecl_gfc_str , % TODO(?)
1895   specializes .str_set:N  = \l_stex_symdecl_specializes_str , % TODO(?)
1896   def       .tl_set:N    = \l_stex_symdecl_definiens_tl
1897 }
```

```

1898
1899 \bool_new:N \l_stex_symdecl_make_macro_bool
1900
1901 \cs_new_protected:Nn \__stex_symdecl_args:n {
1902   \str_clear:N \l_stex_symdecl_name_str
1903   \str_clear:N \l_stex_symdecl_args_str
1904   \bool_set_false:N \l_stex_symdecl_local_bool
1905   \tl_clear:N \l_stex_symdecl_type_tl
1906   \tl_clear:N \l_stex_symdecl_definiens_tl
1907
1908   \keys_set:nn { stex / symdecl } { #1 }
1909 }

```

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

```

1910
1911 \NewDocumentCommand \symdecl { s O{} m } {
1912   \__stex_symdecl_args:n { #2 }
1913   \IfBooleanTF #1 {
1914     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1915   } {
1916     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1917   }
1918   \stex_symdecl_do:n { #3 }
1919   \stex_smsmode_set_codes:
1920 }
1921 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

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

```

1922 \cs_new_protected:Nn \stex_symdecl_do:n {
1923   \stex_if_in_module:F {
1924     % TODO throw error? some default namespace?
1925   }
1926
1927   \str_if_empty:NT \l_stex_symdecl_name_str {
1928     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1929   }
1930
1931   \prop_if_exist:cT { l_stex_symdecl_
1932     \l_stex_current_module_str ?
1933     \l_stex_symdecl_name_str
1934     _prop
1935   }{
1936     % TODO throw error (beware of circular dependencies)
1937   }
1938
1939   \prop_clear:N \l_tmpa_prop
1940   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
1941   \seq_clear:N \l_tmpa_seq
1942   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1943   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1944

```

```

1945 \exp_args:No \stex_add_constant_to_current_module:n {
1946   \l_stex_symdecl_name_str
1947 }
1948
1949 % arity/args
1950 \int_zero:N \l_tmpb_int
1951
1952 \bool_set_true:N \l_tmpa_bool
1953 \str_map_inline:Nn \l_stex_symdecl_args_str {
1954   \token_case_meaning:NnF ##1 {
1955     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1956     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1957     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1958     {\tl_to_str:n a} {
1959       \bool_set_false:N \l_tmpa_bool
1960       \int_incr:N \l_tmpb_int
1961     }
1962     {\tl_to_str:n B} {
1963       \bool_set_false:N \l_tmpa_bool
1964       \int_incr:N \l_tmpb_int
1965     }
1966   }{
1967     \msg_set:nnn{stex}{error/wrongargs}{
1968       args~value~in~symbol~declaration~for~
1969       \l_stex_current_module_str ?
1970       \l_stex_symdecl_name_str ~
1971       needs~to~be~
1972       i,~a,~b~or~B,~but~##1~given
1973     }
1974     \msg_error:nn{stex}{error/wrongargs}
1975   }
1976 }
1977 \bool_if:NTF \l_tmpa_bool {
1978   % possibly numeric
1979   \str_if_empty:NTF \l_stex_symdecl_args_str {
1980     \prop_put:Nnn \l_tmpa_prop { args } {}
1981     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1982   }{
1983     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1984     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1985     \str_clear:N \l_tmpa_str
1986     \int_step_inline:nn \l_tmpa_int {
1987       \str_put_right:Nn \l_tmpa_str i
1988     }
1989     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1990   }
1991 } {
1992   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1993   \prop_put:Nnx \l_tmpa_prop { arity }
1994     { \str_count:N \l_stex_symdecl_args_str }
1995 }
1996 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
1997
1998

```

```

1999 % semantic macro
2000
2001 \bool_if:NT \l_stex_symdecl_make_macro_bool {
2002   \exp_args:Nx \stex_do_aftergroup:n {
2003     \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2004       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2005     }}
2006   }
2007
2008   \bool_if:NF \l_stex_symdecl_local_bool {
2009     \exp_args:Nx \stex_add_to_current_module:n {
2010       \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2011         \l_stex_current_module_str ? \l_stex_symdecl_name_str
2012       } }
2013     }
2014   }
2015 }
2016
2017 % add to all symbols
2018
2019 \bool_if:NF \l_stex_symdecl_local_bool {
2020   \exp_args:Nx \stex_add_to_current_module:n {
2021     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2022       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2023     }
2024   }
2025 %   \exp_args:Nx \stex_add_field_to_current_module:n {
2026 %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2027 %   }
2028 }
2029
2030 \stex_debug:nn{symbols}{New~symbol:~
2031   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2032   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2033   Args:~\prop_item:Nn \l_tmpa_prop { args }
2034 }
2035
2036 % circular dependencies require this:
2037
2038 \prop_if_exist:cF {
2039   l_stex_symdecl_
2040   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2041   _prop
2042 } {
2043   \prop_set_eq:cN {
2044     l_stex_symdecl_
2045     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2046     _prop
2047   } \l_tmpa_prop
2048 }
2049
2050 \seq_clear:c {
2051   l_stex_symdecl_
2052   \l_stex_current_module_str ? \l_stex_symdecl_name_str

```

```

2053   _notations
2054 }
2055
2056 \bool_if:NF \l_stex_symdecl_local_bool {
2057   \exp_args:Nx
2058   \stex_add_to_current_module:n {
2059     \seq_clear:c {
2060       l_stex_symdecl_
2061       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2062       _notations
2063     }
2064     \prop_set_from_keyval:cn {
2065       l_stex_symdecl_
2066       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2067       _prop
2068     } {
2069       name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2070       module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2071       type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2072       args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2073       arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2074       assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
2075     }
2076   }
2077 }
2078
2079 \stex_if_smsmode:TF {
2080   \bool_if:NF \l_stex_symdecl_local_bool {
2081     % \exp_args:Nx \stex_add_to_sms:n {
2082     %   \prop_set_from_keyval:cn {
2083     %     l_stex_symdecl_
2084     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2085     %     _prop
2086     %   } {
2087     %     name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2088     %     module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2089     %     local     = \prop_item:Nn \l_tmpa_prop { local }     ,
2090     %     type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2091     %     args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2092     %     arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2093     %     assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
2094     %   }
2095     %   \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2096     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2097     %   }
2098     % }
2099   }
2100 }{
2101   \exp_args:Nx \stex_do_aftergroup:n {
2102     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2103       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2104     }
2105   }
2106   \stex_if_do_html:T {

```



```

2107 \stex_annotate_invisible:nnn {symdecl} {
2108   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2109 } {
2110   \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}{}}{ $\l_st
2111   \stex_annotate_invisible:nnn{args}{}{
2112     \prop_item:Nn \l_tmpa_prop { args }
2113   }
2114   \stex_annotate_invisible:nnn{macroname}{#1}{}
2115   \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2116     \stex_annotate_invisible:nnn{definiens}{}
2117     { $\l_stex_symdecl_definiens_tl$ }
2118   }
2119 }
2120 }
2121 }
2122 }

```

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

`\stex_get_symbol:n`

```

2123 \str_new:N \l_stex_get_symbol_uri_str
2124
2125 \cs_new_protected:Nn \stex_get_symbol:n {
2126   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2127     \__stex_symdecl_get_symbol_from_cs:n { #1 }
2128   }{
2129     % argument is a string
2130     % is it a command name?
2131     \cs_if_exist:cTF { #1 }{
2132       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2133       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2134       \str_if_empty:NNTF \l_tmpa_str {
2135         \exp_args:Nx \cs_if_eq:NNTF {
2136           \tl_head:N \l_tmpa_tl
2137         } \stex_invoke_symbol:n {
2138           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
2139         }{
2140           \__stex_symdecl_get_symbol_from_string:n { #1 }
2141         }
2142       } {
2143         \__stex_symdecl_get_symbol_from_string:n { #1 }
2144       }
2145     }{
2146       % argument is not a command name
2147       \__stex_symdecl_get_symbol_from_string:n { #1 }
2148       % \l_stex_all_symbols_seq
2149     }
2150   }
2151 }
2152
2153 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2154   \str_set:Nn \l_tmpa_str { #1 }
2155   \bool_set_false:N \l_tmpa_bool
2156   \stex_if_in_module:T {

```

```

2157 \exp_args:Nno \seq_if_in:cnT {c_stex_module_\l_stex_current_module_str _constants} { \l_
2158 \bool_set_true:N \l_tmpa_bool
2159 \str_set:Nx \l_stex_get_symbol_uri_str {
2160 \l_stex_current_module_str ? #1
2161 }
2162 }
2163 }
2164 \bool_if:NF \l_tmpa_bool {
2165 \tl_set:Nn \l_tmpa_tl {
2166 \msg_set:nnn{stex}{error/unknownsymbol}{
2167 No~symbol~#1~found!
2168 }
2169 \msg_error:nn{stex}{error/unknownsymbol}
2170 }
2171 \str_set:Nn \l_tmpa_str { #1 }
2172 \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2173 \seq_map_inline:Nn \l_stex_all_symbols_seq {
2174 \str_set:Nn \l_tmpb_str { ##1 }
2175 \str_if_eq:eeT { \l_tmpa_str } {
2176 \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
2177 } {
2178 \seq_map_break:n {
2179 \tl_set:Nn \l_tmpa_tl {
2180 \str_set:Nn \l_stex_get_symbol_uri_str {
2181 ##1
2182 }
2183 }
2184 }
2185 }
2186 }
2187 \l_tmpa_tl
2188 }
2189 }
2190
2191 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
2192 \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2193 { \tl_tail:N \l_tmpa_tl }
2194 \tl_if_single:NTF \l_tmpa_tl {
2195 \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2196 \exp_after:wN \str_set:Nn \exp_after:wN
2197 \l_stex_get_symbol_uri_str \l_tmpa_tl
2198 }{
2199 % TODO
2200 % tail is not a single group
2201 }
2202 }{
2203 % TODO
2204 % tail is not a single group
2205 }
2206 }

```

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

## 30.2 Notations

```

2207 <@@=stex_notation>

      notation arguments:
2208 \keys_define:nn { stex / notation } {
2209   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2210   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2211   prec .str_set_x:N = \l__stex_notation_prec_str ,
2212   op .tl_set:N = \l__stex_notation_op_tl ,
2213   primary .bool_set:N = \l__stex_notation_primary_bool ,
2214   primary .default:n = {true} ,
2215   unknown .code:n = \str_set:Nx
2216     \l__stex_notation_variant_str \l_keys_key_str
2217 }
2218
2219 \cs_new_protected:Nn \_stex_notation_args:n {
2220   \str_clear:N \l__stex_notation_lang_str
2221   \str_clear:N \l__stex_notation_variant_str
2222   \str_clear:N \l__stex_notation_prec_str
2223   \tl_clear:N \l__stex_notation_op_tl
2224   \bool_set_false:N \l__stex_notation_primary_bool
2225
2226   \keys_set:nn { stex / notation } { #1 }
2227 }

```

**\notation**

```

2228 \NewDocumentCommand \notation { 0{ } m } {
2229   \_stex_notation_args:n { #1 }
2230   \tl_clear:N \l_stex_symdecl_definiens_tl
2231   \stex_get_symbol:n { #2 }
2232   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
2233 }
2234 \stex_deactivate_macro:Nn \notation {module-environments}

```

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

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

```

2235 \cs_new_protected:Nn \stex_notation_do:nn {
2236   \let\l_stex_current_symbol_str\relax
2237   \prop_set_eq:Nc \l_tmpa_prop {
2238     \l_stex_symdecl_ #1 _prop
2239   }
2240
2241   \prop_clear:N \l_tmpb_prop
2242   \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
2243   \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
2244   \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
2245
2246   % precedences
2247   \seq_clear:N \l_tmpb_seq
2248   \exp_args:NNno
2249   \str_if_empty:NTF \l__stex_notation_prec_str {
2250     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2251     \int_compare:nNnTF \l_tmpa_str = 0 {

```

```

2252     \exp_args:NNnx
2253     \prop_put:Nno \l_tmpb_prop { opprec }
2254     { \neginfprec }
2255   }{
2256     \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
2257   }
2258 } {
2259   \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2260     \exp_args:NNnx
2261     \prop_put:Nno \l_tmpb_prop { opprec }
2262     { \neginfprec }
2263     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2264     \int_step_inline:nn { \l_tmpa_str } {
2265       \exp_args:NNx
2266       \seq_put_right:Nn \l_tmpb_seq { \infprec }
2267     }
2268   }{
2269     \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2270     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2271       \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
2272       \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2273         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2274         \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2275         \seq_map_inline:Nn \l_tmpa_seq {
2276           \seq_put_right:Nn \l_tmpb_seq { ##1 }
2277         }
2278       }
2279       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2280     }{
2281       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2282       \int_compare:nNnTF \l_tmpa_str = 0 {
2283         \exp_args:NNnx
2284         \prop_put:Nno \l_tmpb_prop { opprec }
2285         { \infprec }
2286       }{
2287         \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
2288       }
2289     }
2290   }
2291 }
2292
2293 \seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
2294 \int_step_inline:nn { \l_tmpa_str } {
2295   \seq_pop_left:NNTF \l_tmpa_seq \l_tmpb_str {
2296     \exp_args:NNx
2297     \seq_put_right:Nn \l_tmpb_seq {
2298       \prop_item:Nn \l_tmpb_prop { opprec }
2299     }
2300   }
2301 }
2302
2303 \prop_put:Nno \l_tmpb_prop { argprecs } \l_tmpb_seq
2304 \tl_clear:N \l_tmpa_tl
2305

```

```

2306 \int_compare:nNnTF \l_tmpa_str = 0 {
2307   \exp_args:NNe
2308   \cs_set:Npn \l__stex_notation_macrocode_cs {
2309     \_stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2310     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2311     { \prop_item:Nn \l_tmpb_prop { opprec } }
2312     { \exp_not:n { #2 } }
2313   }
2314   \__stex_notation_final:
2315 }{
2316   \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
2317   \str_if_in:NnTF \l_tmpb_str b {
2318     \exp_args:Nne \use:nn
2319     {
2320       \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2321       \cs_set:Npn \l_tmpa_str } { {
2322         \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2323         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2324         { \prop_item:Nn \l_tmpb_prop { opprec } }
2325         { \exp_not:n { #2 } }
2326       } }
2327   }{
2328     \str_if_in:NnTF \l_tmpb_str B {
2329       \exp_args:Nne \use:nn
2330       {
2331         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2332         \cs_set:Npn \l_tmpa_str } { {
2333           \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2334           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2335           { \prop_item:Nn \l_tmpb_prop { opprec } }
2336           { \exp_not:n { #2 } }
2337         } }
2338     }{
2339       \exp_args:Nne \use:nn
2340       {
2341         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2342         \cs_set:Npn \l_tmpa_str } { {
2343           \_stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2344           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2345           { \prop_item:Nn \l_tmpb_prop { opprec } }
2346           { \exp_not:n { #2 } }
2347         } }
2348     }
2349   }
2350
2351   \int_zero:N \l_tmpa_int
2352   \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2353   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2354   \__stex_notation_arguments:
2355 }
2356 }

```

(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

```

2357 \cs_new_protected:Nn \_stex_notation_arguments: {
2358   \int_incr:N \l_tmpa_int
2359   \str_if_empty:NTF \l_tmpa_str {
2360     \_stex_notation_final:
2361   }{
2362     \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2363     \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2364     \str_if_eq:VnTF \l_tmpb_str a {
2365       \_stex_notation_argument_assoc:n
2366     }{
2367       \str_if_eq:VnTF \l_tmpb_str B {
2368         \_stex_notation_argument_assoc:n
2369       }{
2370         \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2371         \tl_put_right:Nx \l_tmpa_tl {
2372           { \_stex_term_math_arg:nnn
2373             { \int_use:N \l_tmpa_int }
2374             { \l_tmpb_str }
2375             { ####\int_use:N \l_tmpa_int }
2376           }
2377         }
2378         \_stex_notation_arguments:
2379       }
2380     }
2381   }
2382 }
```

(End definition for `\_stex_notation_arguments:.`)

`\_stex_notation_argument_assoc:n`

```

2383 \cs_new_protected:Nn \_stex_notation_argument_assoc:n {
2384   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2385   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
2386   \tl_put_right:Nx \l_tmpa_tl {
2387     { \_stex_term_math_assoc_arg:nnnn
2388       { \int_use:N \l_tmpa_int }
2389       { \l_tmpb_str }
2390       \exp_args:No \exp_not:n
2391       {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
2392       { ####\int_use:N \l_tmpa_int }
2393     }
2394   }
2395   \_stex_notation_arguments:
2396 }
```

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

`\_stex_notation_final:` Called after processing all notation arguments

```

2397 \cs_new_protected:Nn \_stex_notation_final: {
2398   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
2399   \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
2400   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2401   \exp_args:Nne \use:nn
```

```

2402 {
2403 \cs_generate_from_arg_count:cNnn {
2404     stex_notation_ \l_tmpa_str \c_hash_str
2405     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2406     _cs
2407 }
2408 \cs_set:Npn \l_tmpb_str } { {
2409     \exp_after:wN \exp_after:wN \exp_after:wN
2410     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2411     { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2412 } }
2413
2414 \tl_if_empty:NF \l__stex_notation_op_tl {
2415     \cs_set:cpx {
2416         stex_op_notation_ \l_tmpa_str \c_hash_str
2417         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2418         _cs
2419     } {
2420         \_stex_term_oms:nnn {
2421             \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2422             \l__stex_notation_lang_str
2423         }{
2424             \l_tmpa_str
2425         }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2426     }
2427 }
2428
2429 \exp_args:Ne
2430 \stex_add_to_current_module:n {
2431     \cs_generate_from_arg_count:cNnn {
2432         stex_notation_ \l_tmpa_str \c_hash_str
2433         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2434         _cs
2435     } \cs_set:Npn {\l_tmpb_str} {
2436         \exp_after:wN \exp_after:wN \exp_after:wN
2437         \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2438         { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2439     }
2440     \tl_if_empty:NF \l__stex_notation_op_tl {
2441         \cs_set:cpn {
2442             stex_op_notation_ \l_tmpa_str \c_hash_str
2443             \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2444             _cs
2445         } {
2446             \_stex_term_oms:nnn {
2447                 \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2448                 \l__stex_notation_lang_str
2449             }{
2450                 \l_tmpa_str
2451             }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2452         }
2453     }
2454 }
2455

```

```

2456 \seq_put_right:cx {
2457   l_stex_symdecl_
2458   \prop_item:Nn \l_tmpb_prop { symbol }
2459   _notations
2460 } {
2461   \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2462 }
2463
2464 \stex_debug:nn{symbols}{
2465   Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2466   ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
2467   Operator~precedence:~
2468   \prop_item:Nn \l_tmpb_prop { opprec }^^J
2469   Argument~precedences:~
2470   \seq_use:Nn \l_tmpa_seq {,~}^^J
2471   Notation: \cs_meaning:c {
2472     stex_notation_ \l_tmpa_str \c_hash_str
2473     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2474     _cs
2475   }
2476 }
2477
2478 \prop_set_eq:cN {
2479   l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2480   \c_hash_str \l__stex_notation_lang_str _prop
2481 } \l_tmpb_prop
2482
2483 \exp_args:Ne
2484 \stex_add_to_current_module:n {
2485   \seq_put_right:cn {
2486     l_stex_symdecl_
2487     \prop_item:Nn \l_tmpb_prop { symbol }
2488     _notations
2489   } {
2490     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2491   }
2492   \prop_set_from_keyval:cn {
2493     l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2494     \c_hash_str \l__stex_notation_lang_str _prop
2495   } {
2496     symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2497     language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2498     variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,
2499     opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2500     argprecs    = \prop_item:Nn \l_tmpb_prop { argprecs }    ,
2501   }
2502 }
2503
2504 \stex_if_smsmode:TF {
2505   \stex_smsmode_set_codes:
2506   % \exp_args:Nx \stex_add_to_sms:n {
2507   %   \prop_set_from_keyval:cn {
2508   %     l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2509   %     \c_hash_str \l__stex_notation_lang_str _prop

```



```

2510 %      } {
2511 %          symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2512 %          language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2513 %          variant      = \prop_item:Nn \l_tmpb_prop { variant }      ,
2514 %          opprec       = \prop_item:Nn \l_tmpb_prop { opprec }       ,
2515 %          argprec      = \prop_item:Nn \l_tmpb_prop { argprec }      ,
2516 %      }
2517 %  }
2518 }{
2519
2520 % HTML annotations
2521 \stex_if_do_html:T {
2522     \stex_annotate_invisible:nnn { notation }
2523     { \prop_item:Nn \l_tmpb_prop { symbol } } {
2524         \stex_annotate_invisible:nnn { notationfragment }
2525         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
2526         \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2527         \stex_annotate_invisible:nnn { precedence }
2528         { \prop_item:Nn \l_tmpb_prop { opprec };
2529           \seq_use:Nn \l_tmpa_seq { x }
2530         }{}
2531
2532         \int_zero:N \l_tmpa_int
2533         \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2534         \tl_clear:N \l_tmpa_tl
2535         \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
2536             \int_incr:N \l_tmpa_int
2537             \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2538             \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2539             \str_if_eq:VnTF \l_tmpb_str a {
2540                 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2541                     \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2542                     \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2543                 } }
2544             }{
2545                 \str_if_eq:VnTF \l_tmpb_str B {
2546                     \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2547                         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2548                         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2549                     } }
2550                 }{
2551                     \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2552                         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2553                     } }
2554                 }
2555             }
2556         }
2557         \stex_annotate_invisible:nnn { notationcomp }{}{
2558             \str_set:Nx \l_stex_current_symbol_str {\prop_item:Nn \l_tmpb_prop { symbol }}
2559             $ \exp_args:Nno \use:nn { \use:c {
2560                 stex_notation_ \l_stex_current_symbol_str
2561                 \c_hash_str \l__stex_notation_variant_str
2562                 \c_hash_str \l__stex_notation_lang_str _cs
2563             } } { \l_tmpa_tl } $

```

```

2564     }
2565   }
2566 }
2567 }
2568 }

```

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

\setnotation

```

2569 \keys_define:nn { stex / setnotation } {
2570   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2571   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2572   unknown .code:n = \str_set:Nx
2573     \l__stex_notation_variant_str \l_keys_key_str
2574 }
2575
2576 \cs_new_protected:Nn \_stex_setnotation_args:n {
2577   \str_clear:N \l__stex_notation_lang_str
2578   \str_clear:N \l__stex_notation_variant_str
2579   \keys_set:nn { stex / setnotation } { #1 }
2580 }
2581
2582 \NewDocumentCommand \setnotation {m m} {
2583   \stex_get_symbol:n { #1 }
2584   \_stex_setnotation_args:n { #2 }
2585   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl\_l_stex_get_symbol_uri_str _notations }
2586     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2587     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notation
2588       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2589     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notation
2590       { \c_hash_str }
2591     \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notations
2592       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2593     \exp_args:Nx \stex_add_to_current_module:n {
2594       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notati
2595         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2596       \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notation
2597         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2598       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notati
2599         { \c_hash_str }
2600     }
2601     \stex_debug:nn {notations}{
2602       Setting~default~notation~
2603       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }~for~
2604       \l_stex_get_symbol_uri_str \
2605       \expandafter\meaning\csname
2606       l_stex_symdecl\_l_stex_get_symbol_uri_str _notations\endcsname
2607     }
2608   }{
2609     % todo throw error
2610   }
2611 }
2612

```

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

**\symdef**

```
2613 \keys_define:nn { stex / symdef } {
2614   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2615   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2616   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2617   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
2618   def       .tl_set:N    = \l_stex_symdecl_definiens_tl ,
2619   op        .tl_set:N    = \l__stex_notation_op_tl ,
2620   lang      .str_set_x:N = \l__stex_notation_lang_str ,
2621   variant   .str_set_x:N = \l__stex_notation_variant_str ,
2622   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2623   unknown   .code:n      = \str_set:Nx
2624             \l__stex_notation_variant_str \l_keys_key_str
2625 }
2626
2627 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2628   \str_clear:N \l_stex_symdecl_name_str
2629   \str_clear:N \l_stex_symdecl_args_str
2630   \bool_set_false:N \l_stex_symdecl_local_bool
2631   \tl_clear:N \l_stex_symdecl_type_tl
2632   \tl_clear:N \l_stex_symdecl_definiens_tl
2633   \str_clear:N \l__stex_notation_lang_str
2634   \str_clear:N \l__stex_notation_variant_str
2635   \str_clear:N \l__stex_notation_prec_str
2636   \tl_clear:N \l__stex_notation_op_tl
2637
2638   \keys_set:nn { stex / symdef } { #1 }
2639 }
2640
2641 \NewDocumentCommand \symdef { 0{} m } {
2642   \__stex_notation_symdef_args:n { #1 }
2643   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2644   \stex_symdecl_do:n { #2 }
2645   \exp_args:Nx \stex_notation_do:nn {
2646     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2647   }
2648 }
2649 \stex_deactivate_macro:Nn \symdef {module~environments}
2650
2650 (End definition for \symdef. This function is documented on page 37.)
2650 \endpackage
```

## Chapter 31

# STEX -Terms Implementation

```
2651 <*package>
2652
2653 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2654
2655 <@@=stex_terms>
2656
2657 Warnings and error messages
2658 \msg_new:nnn{stex}{error/nonotation}{
2659   Symbol~#1~invoked,~but~has~no~notation#2!
2660 }
2661 \msg_new:nnn{stex}{error/notationarg}{
2662   Error~in~parsing~notation~#1
2663 }
2664 \msg_new:nnn{stex}{error/noop}{
2665   Symbol~#1~has~no~operator~notation~for~notation~#2
2666 }
```

### 31.1 Symbol Invocations

Arguments:

```
2666 \keys_define:nn { stex / terms } {
2667   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2668   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2669   unknown .code:n = \str_set:Nx
2670     \l__stex_terms_variant_str \l_keys_key_str
2671 }
2672
2673 \cs_new_protected:Nn \__stex_terms_args:n {
2674   \str_clear:N \l__stex_terms_lang_str
2675   \str_clear:N \l__stex_terms_variant_str
2676   \str_clear:N \l__stex_terms_prec_str
2677   \tl_clear:N \l__stex_terms_op_tl
2678 }
2679 \keys_set:nn { stex / terms } { #1 }
```

2680 }

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

```
2681 \cs_new_protected:Nn \stex_invoke_symbol:n {
2682   \if_mode_math:
2683     \exp_after:wN \__stex_terms_invoke_math:n
2684   \else:
2685     \exp_after:wN \__stex_terms_invoke_text:n
2686   \fi: { #1 }
2687 }
```

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

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

```
2688 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2689   \peek_charcode_remove:NTF ! {
2690     \peek_charcode:NTF [ {
2691       \__stex_terms_invoke_op:nw { #1 }
2692     }{
2693       \peek_charcode_remove:NTF ! {
2694         \peek_charcode:NTF [ {
2695           \__stex_terms_invoke_op_custom:nw
2696         }{
2697           % TODO throw error
2698         }
2699       }{
2700         \__stex_terms_invoke_op:nw { #1 } []
2701       }
2702     }
2703   }{
2704     \peek_charcode_remove:NTF * {
2705       \__stex_terms_invoke_text:n { #1 }
2706     }{
2707       \peek_charcode:NTF [ {
2708         \__stex_terms_invoke_math:nw { #1 }
2709       }{
2710         \__stex_terms_invoke_math:nw { #1 } []
2711       }
2712     }
2713   }
2714 }
```

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

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

```
2715 \cs_new_protected:Npn \__stex_terms_invoke_op_custom:nw #1 [#2] {
2716   \stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
2717     \stex_highlight_term:nn{#1}{#2}
2718   }
2719 }
```

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

\\_stex\_terms\_invoke\_op:nw

```

2720 \cs_new_protected:Npn \_stex_terms_invoke_op:nw #1 [#2] {
2721   \_stex_terms_args:n { #2 }
2722   \cs_if_exist:cTF {
2723     stex_op_notation_ #1 \c_hash_str
2724     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2725   }{
2726     \csname stex_op_notation_ #1 \c_hash_str
2727       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2728     \endcsname
2729   }{
2730     \msg_error:nnxx{stex}{error/noop}{#1}{\l__stex_terms_variant_str \c_hash_str \l__stex_te
2731   }
2732 }

```

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

\\_stex\_terms\_invoke\_math:nw

```

2733 \cs_new_protected:Npn \_stex_terms_invoke_math:nw #1 [#2] {
2734   \_stex_terms_args:n { #2 }
2735   \seq_if_empty:cTF {
2736     l_stex_symdecl_ #1 _notations
2737   } {
2738     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
2739   } {
2740     \seq_if_in:cxTF {
2741       l_stex_symdecl_ #1 _notations
2742     }
2743     { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
2744       \str_set:Nn \l_stex_current_symbol_str { #1 }
2745       \use:c{
2746         stex_notation_ #1 \c_hash_str
2747         \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2748         _cs
2749       }
2750     }{
2751       \str_if_empty:NTF \l__stex_terms_variant_str {
2752         \str_if_empty:NTF \l__stex_terms_lang_str {
2753           \seq_get_left:cN {
2754             l_stex_symdecl_ #1 _notations
2755           } \l_tmpa_str
2756           \str_set:Nn \l_stex_current_symbol_str { #1 }
2757           \use:c{
2758             stex_notation_ #1 \c_hash_str \l_tmpa_str
2759             _cs
2760           }
2761         }{
2762           \msg_error:nnxx{stex}{error/nonotation}{#1}{
2763             ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2764           }
2765         }
2766       }{
2767         \msg_error:nnxx{stex}{error/nonotation}{#1}{
2768           ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str

```

```

2769     }
2770   }
2771 }
2772 }
2773 }

```

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

`\_stex_terms_invoke_text:n`

```

2774 \cs_new_protected:Nn \_stex_terms_invoke_text:n {
2775   \peek_charcode_remove:NTF ! {
2776     \stex_term_custom:nn { #1 } { }
2777   }{
2778     \prop_set_eq:Nc \l_tmpa_prop {
2779       l_stex_symdecl_ #1 _prop
2780     }
2781     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2782     \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2783   }
2784 }

```

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

## 31.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
2785 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2786 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2787 \int_new:N \l__stex_terms_downprec
2788 \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
2789 \tl_set:Nn \l__stex_terms_left_bracket_str (
2790 \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

```

2791 \cs_new_protected:Nn \_stex_terms_maybe_brackets:nn {
2792   \bool_if:NTF \l__stex_terms_brackets_done_bool {
2793     \bool_set_false:N \l__stex_terms_brackets_done_bool
2794     #2
2795   } {
2796     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
2797       \bool_if:NTF \l_stex_inarray_bool { #2 }{
2798         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
2799         \dobrackets { #2 }
2800       }

```

```

2801     }{ #2 }
2802   }
2803 }

```

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

### `\dobrackets`

```

2804 \bool_new:N \l__stex_terms_brackets_done_bool
2805 %\RequirePackage{scalerel}
2806 \cs_new_protected:Npn \dobrackets #1 {
2807   %\ThisStyle{\if D\m@switch
2808   %   \exp_args:Nnx \use:nn
2809   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
2810   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
2811   % \else
2812   \exp_args:Nnx \use:nn
2813   {
2814     \bool_set_true:N \l__stex_terms_brackets_done_bool
2815     \int_set:Nn \l__stex_terms_downprec \infprec
2816     \l__stex_terms_left_bracket_str
2817     #1
2818   }
2819   {
2820     \bool_set_false:N \l__stex_terms_brackets_done_bool
2821     \l__stex_terms_right_bracket_str
2822     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2823   }
2824   %\fi}
2825 }

```

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

### `\withbrackets`

```

2826 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2827   \exp_args:Nnx \use:nn
2828   {
2829     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
2830     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
2831     #3
2832   }
2833   {
2834     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
2835     {\l__stex_terms_left_bracket_str}
2836     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
2837     {\l__stex_terms_right_bracket_str}
2838   }
2839 }

```

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

### `\STEXinvisible`

```

2840 \cs_new_protected:Npn \STEXinvisible #1 {
2841   \stex_annotate_invisible:n { #1 }
2842 }

```



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

OMDoc terms:

`\_stex_term_math_oms:nnnn`

```

2843 \cs_new_protected:Nn \_stex_term_oms:nnn {
2844   \stex_annotate:nnn{ OMID }{ #2 }{
2845     \stex_highlight_term:nn { #1 } { #3 }
2846   }
2847 }
2848
2849 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2850   \__stex_terms_maybe_brackets:nn { #3 }{
2851     \stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2852   }
2853 }

```

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

`\_stex_term_math_oma:nnnn`

```

2854 \cs_new_protected:Nn \_stex_term_oma:nnn {
2855   \stex_annotate:nnn{ OMA }{ #2 }{
2856     \stex_highlight_term:nn { #1 } { #3 }
2857   }
2858 }
2859
2860 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2861   \__stex_terms_maybe_brackets:nn { #3 }{
2862     \stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2863   }
2864 }

```

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

`\_stex_term_math_omb:nnnn`

```

2865 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2866   \stex_annotate:nnn{ OMBIND }{ #2 }{
2867     \stex_highlight_term:nn { #1 } { #3 }
2868   }
2869 }
2870
2871 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2872   \__stex_terms_maybe_brackets:nn { #3 }{
2873     \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2874   }
2875 }

```

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

`\_stex_term_math_arg:nnn`

```

2876 \cs_new_protected:Nn \_stex_term_arg:nn {
2877   \stex_unhighlight_term:n {
2878     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2879   }
2880 }

```

```

2881 \cs_new_protected:Nn \stex_term_math_arg:nnn {
2882   \exp_args:Nnx \use:nn
2883     { \int_set:Nn \l__stex_terms_downprec { #2 }
2884       \stex_term_arg:nn { #1 }{ #3 }
2885     }
2886     { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2887   }

```

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

`\stex_term_math_assoc_arg:nnnn`

```

2888 \cs_new_protected:Nn \stex_term_math_assoc_arg:nnnn {
2889   \clist_set:Nn \l_tmpa_clist{ #4 }
2890   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
2891     \tl_set:Nn \l_tmpa_tl { #4 }
2892   }{
2893     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
2894     \clist_reverse:N \l_tmpa_clist
2895     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
2896
2897     \clist_map_inline:Nn \l_tmpa_clist {
2898       \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
2899         \exp_args:Nno
2900           \l_tmpa_cs { ##1 } \l_tmpa_tl
2901       }
2902     }
2903
2904   }
2905   \exp_args:Nnno
2906   \stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2907 }

```

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

`\stex_term_custom:nn`

```

2908 \cs_new_protected:Nn \stex_term_custom:nn {
2909   \str_set:Nn \l__stex_terms_custom_uri { #1 }
2910   \str_set:Nn \l_tmpa_str { #2 }
2911   \tl_clear:N \l_tmpa_tl
2912   \int_zero:N \l_tmpa_int
2913   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2914   \__stex_terms_custom_loop:
2915 }

```

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

`\__stex_terms_custom_loop:`

```

2916 \cs_new_protected:Nn \__stex_terms_custom_loop: {
2917   \bool_set_false:N \l_tmpa_bool
2918   \bool_while_do:nn {
2919     \str_if_eq_p:ee X {
2920       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2921     }
2922   }{
2923     \int_incr:N \l_tmpa_int

```

```

2924 }
2925
2926 \peek_charcode:NTF [ {
2927   % notation/text component
2928   \__stex_terms_custom_component:w
2929 } {
2930   \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2931     % all arguments read => finish
2932     \__stex_terms_custom_final:
2933   } {
2934     % arguments missing
2935     \peek_charcode_remove:NTF * {
2936       % invisible, specific argument position or both
2937       \peek_charcode:NTF [ {
2938         % visible specific argument position
2939         \__stex_terms_custom_arg:wn
2940       } {
2941         % invisible
2942         \peek_charcode_remove:NTF * {
2943           % invisible specific argument position
2944           \__stex_terms_custom_arg_inv:wn
2945         } {
2946           % invisible next argument
2947           \__stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2948         }
2949       }
2950     } {
2951       % next normal argument
2952       \__stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
2953     }
2954   }
2955 }
2956 }

```

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

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

```

2957 \cs_new_protected:Npn \__stex_terms_custom_arg_inv:wn [ #1 ] #2 {
2958   \bool_set_true:N \l_tmpa_bool
2959   \__stex_terms_custom_arg:wn [ #1 ] { #2 }
2960 }

```

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

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

```

2961 \cs_new_protected:Npn \__stex_terms_custom_arg:wn [ #1 ] #2 {
2962   \str_set:Nx \l_tmpb_str {
2963     \str_item:Nn \l_tmpa_str { #1 }
2964   }
2965   \str_case:VnTF \l_tmpb_str {
2966     { X } {
2967       \msg_error:nnx{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2968     }
2969     { i } { \__stex_terms_custom_set_X:n { #1 } }
2970     { b } { \__stex_terms_custom_set_X:n { #1 } }

```

```

2971     { a } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
2972     { B } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
2973   }{}{
2974     \msg_error:nnx{stex}{error/notationarg}{\l\_stex_terms_custom_uri}
2975   }
2976
2977   \bool_if:nTF \l_tmpa_bool {
2978     \tl_put_right:Nx \l_tmpa_tl {
2979       \stex_annotate_invisible:n {
2980         \stex_term_arg:nn { \int_eval:n { #1 } }
2981         \exp_not:n { { #2 } }
2982       }
2983     }
2984   } {
2985     \tl_put_right:Nx \l_tmpa_tl {
2986       \stex_term_arg:nn { \int_eval:n { #1 } }
2987       \exp_not:n { { #2 } }
2988     }
2989   }
2990
2991   \_stex_terms_custom_loop:
2992 }

```

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

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

```

2993 \cs_new_protected:Nn \_stex_terms_custom_set_X:n {
2994   \str_set:Nx \l_tmpa_str {
2995     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
2996     X
2997     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
2998   }
2999 }

```

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

\\_stex\_terms\_custom\_component:

```

3000 \cs_new_protected:Npn \_stex_terms_custom_component:w [ #1 ] {
3001   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
3002   \_stex_terms_custom_loop:
3003 }

```

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

\\_stex\_terms\_custom\_final:

```

3004 \cs_new_protected:Nn \_stex_terms_custom_final: {
3005   \int_compare:nNnTF \l_tmpb_int = 0 {
3006     \exp_args:Nnno \stex_term_oms:nnn
3007   }{
3008     \str_if_in:NnTF \l_tmpa_str {b} {
3009       \exp_args:Nnno \stex_term_ombind:nnn
3010     } {
3011       \exp_args:Nnno \stex_term_oma:nnn
3012     }
3013   }

```

```

3014 { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
3015 }

```

(End definition for `\_stex_terms_custom_final:`.)

`\symref`

`\symname`

```

3016 \NewDocumentCommand \symref { m m }{
3017   \let\compemph_uri_prev:\compemph@uri
3018   \let\compemph@uri\symrefemph@uri
3019   \STEXsymbol{#1}! [#2]
3020   \let\compemph@uri\compemph_uri_prev:
3021 }
3022
3023 \keys_define:nn { stex / symname } {
3024   post      .str_set_x:N    = \l_stex_symname_post_str
3025 }
3026
3027 \cs_new_protected:Nn \stex_symname_args:n {
3028   \str_clear:N \l_stex_symname_post_str
3029   \keys_set:nn { stex / symname } { #1 }
3030 }
3031
3032 \NewDocumentCommand \symname { O{} m }{
3033   \stex_symname_args:n { #1 }
3034   \stex_get_symbol:n { #2 }
3035   \str_set:Nx \l_tmpa_str {
3036     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3037   }
3038   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3039
3040   \let\compemph_uri_prev:\compemph@uri
3041   \let\compemph@uri\symrefemph@uri
3042   \exp_args:NNx \use:nn
3043   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }! [
3044     \l_tmpa_str \l_stex_symname_post_str
3045   ] }
3046   \let\compemph@uri\compemph_uri_prev:
3047 }

```

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

### 31.3 Notation Components

```

3048 <@@=stex_notationcomps>

```

`\stex_highlight_term:nn`

```

3049
3050 \str_new:N \l_stex_current_symbol_str
3051 \cs_new_protected:Nn \stex_highlight_term:nn {
3052   \exp_args:Nnx
3053   \use:nn {
3054     \str_set:Nx \l_stex_current_symbol_str { #1 }
3055     #2
3056   } {

```

```

3057 \str_set:Nx \exp_not:N \l_stex_current_symbol_str
3058 { \l_stex_current_symbol_str }
3059 }
3060 }
3061
3062 \cs_new_protected:Nn \stex_unhighlight_term:n {
3063 % \latexml_if:TF {
3064 % #1
3065 % } {
3066 % \rustex_if:TF {
3067 % #1
3068 % } {
3069 % #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
3070 % }
3071 % }
3072 }

```

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

```

\comp
\compemph@uri 3073 \cs_new_protected:Npn \comp #1 {
\compemph 3074 \str_if_empty:NF \l_stex_current_symbol_str {
\defemph 3075 \rustex_if:TF {
\defemph@uri 3076 \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
\symrefemph 3077 }{
\symrefemph@uri 3078 \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3079 }
3080 }
3081 }
3082
3083 \cs_new_protected:Npn \compemph@uri #1 #2 {
3084 \compemph{ #1 }
3085 }
3086
3087
3088 \cs_new_protected:Npn \compemph #1 {
3089 #1
3090 }
3091
3092 \cs_new_protected:Npn \defemph@uri #1 #2 {
3093 \defemph{#1}
3094 }
3095
3096 \cs_new_protected:Npn \defemph #1 {
3097 \textbf{#1}
3098 }
3099
3100 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3101 \symrefemph{#1}
3102 }
3103
3104 \cs_new_protected:Npn \symrefemph #1 {
3105 \textbf{#1}
3106 }

```

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

## `\ellipses`

```
3107 \NewDocumentCommand \ellipses {} { \ldots }
```

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

```

\parray
\prmatrix 3108 \bool_new:N \l_stex_inarray_bool
\parrayline 3109 \bool_set_false:N \l_stex_inarray_bool
\parraylineh 3110 \NewDocumentCommand \parray { m m } {
\parraycell 3111 \begin{group}
3112 \bool_set_true:N \l_stex_inarray_bool
3113 \begin{array}{#1}
3114 #2
3115 \end{array}
3116 \end{group}
3117 }
3118
3119 \NewDocumentCommand \prmatrix { m } {
3120 \begin{group}
3121 \bool_set_true:N \l_stex_inarray_bool
3122 \begin{matrix}
3123 #1
3124 \end{matrix}
3125 \end{group}
3126 }
3127
3128 \def \maybepline {
3129 \bool_if:NT \l_stex_inarray_bool {\hline}
3130 }
3131
3132 \def \parrayline #1 #2 {
3133 #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3134 }
3135
3136 \def \pmrow #1 { \parrayline{}{ #1 } }
3137
3138 \def \parraylineh #1 #2 {
3139 #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3140 }
3141
3142 \def \parraycell #1 {
3143 #1 \bool_if:NT \l_stex_inarray_bool {&}
3144 }

```

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

```
3145 \end{package}
```

## Chapter 32

# STEX -Structural Features Implementation

```
3146 <*package>
3147
3148 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3149
3150 <@@=stex_features>
3151
3152   Warnings and error messages
3153 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3154   Symbol~#1~can~not~be~assigned~in~copymodule~#2
3155 }
3156 \msg_new:nnn{stex}{error/interpretmodule/noddefinens}{
3157   Symbol~#1~not~assigned~in~interpretmodule~#2
3158 }
```

### 32.1 Imports with modification

```
3158 \cs_new_protected:Nn \stex_get_symbol_in_copymodule:n {
3159   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3160     \__stex_features_get_symbol_from_cs:n { #1 }
3161   }{
3162     % argument is a string
3163     % is it a command name?
3164     \cs_if_exist:cTF { #1 }{
3165       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3166       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3167       \str_if_empty:NNTF \l_tmpa_str {
3168         \exp_args:Nx \cs_if_eq:NNTF {
3169           \tl_head:N \l_tmpa_tl
3170         } \stex_invoke_symbol:n {
3171           \exp_args:No \__stex_features_get_symbol_from_cs:n { \use:c { #1 } }
3172         }{
3173           \__stex_features_get_symbol_from_string:n { #1 }
3174         }
3175       }
3176     }
3177   }
```



```

3174     }
3175   } {
3176     \__stex_features_get_symbol_from_string:n { #1 }
3177   }
3178   ){
3179     % argument is not a command name
3180     \__stex_features_get_symbol_from_string:n { #1 }
3181     % \l_stex_all_symbols_seq
3182   }
3183 }
3184 }
3185
3186 \cs_new_protected:Nn \__stex_features_get_symbol_from_string:n {
3187   \str_set:Nn \l_tmpa_str { #1 }
3188   \bool_set_false:N \l_tmpa_bool
3189   \bool_if:NF \l_tmpa_bool {
3190     \tl_set:Nn \l_tmpa_tl {
3191       \msg_set:nnn{stex}{error/unknownsymbol}{
3192         No~symbol~#1~found!
3193       }
3194       \msg_error:nn{stex}{error/unknownsymbol}
3195     }
3196     \str_set:Nn \l_tmpa_str { #1 }
3197     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3198     \seq_map_inline:Nn \l__stex_features_copymodule_fields_seq {
3199       \str_set:Nn \l_tmpb_str { ##1 }
3200       \str_if_eq:eeT { \l_tmpa_str } {
3201         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3202       } {
3203         \seq_map_break:n {
3204           \tl_set:Nn \l_tmpa_tl {
3205             \str_set:Nn \l_stex_get_symbol_uri_str {
3206               ##1
3207             }
3208             \__stex_features_get_symbol_check:
3209           }
3210         }
3211       }
3212     }
3213     \l_tmpa_tl
3214   }
3215 }
3216
3217 \cs_new_protected:Nn \__stex_features_get_symbol_from_cs:n {
3218   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3219   { \tl_tail:N \l_tmpa_tl }
3220   \tl_if_single:NTF \l_tmpa_tl {
3221     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3222       \exp_after:wN \str_set:Nn \exp_after:wN
3223       \l_stex_get_symbol_uri_str \l_tmpa_tl
3224       \__stex_features_get_symbol_check:
3225     }{
3226       % TODO
3227       % tail is not a single group

```

```

3228     }
3229 }{
3230     % TODO
3231     % tail is not a single group
3232 }
3233 }
3234
3235 \cs_new_protected:Nn \__stex_features_get_symbol_check: {
3236     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq {?} \l_stex_get_symbol_uri_str
3237     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 3 {
3238         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
3239         \str_set:Nx \l_tmpa_str {\seq_use:Nn \l_tmpa_seq ?}
3240         \seq_if_in:Nof \l__stex_features_copymodule_modules_seq \l_tmpa_str {
3241             \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3242                 \l_stex_current_copymodule_name_str\Allowed:~\seq_use:Nn \l__stex_features_copymodule_modules_seq \l_tmpa_str
3243             }
3244         }
3245     }{
3246         \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3247             \l_stex_current_copymodule_name_str~(inexplicably)
3248         }
3249     }
3250 }
3251
3252 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
3253     \stex_import_module_uri:nn { #1 } { #2 }
3254     \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3255     \stex_import_require_module:nnnn
3256     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3257     { \l_stex_import_path_str } { \l_stex_import_name_str }
3258     \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3259     \seq_set_eq:NN \l__stex_features_copymodule_modules_seq \l_stex_collect_imports_seq
3260     \seq_clear:N \l__stex_features_copymodule_fields_seq
3261     \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3262         \seq_map_inline:cn {c_stex_module_###1_constants}{
3263             \exp_args:NNx \seq_put_right:Nn \l__stex_features_copymodule_fields_seq {
3264                 ###1 ? #####1
3265             }
3266         }
3267     }
3268     \seq_clear:N \l_tmpa_seq
3269     \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3270         name      = \l_stex_current_copymodule_name_str ,
3271         module     = \l_stex_current_module_str ,
3272         from       = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3273         includes   = \l_tmpa_seq ,
3274         fields     = \l_tmpa_seq
3275     }
3276     \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3277         as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3278     \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_features_copymodule_modules_seq
3279     \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_features_copymodule_fields_seq {,~}
3280     \stex_if_smsmode:TF {
3281         \stex_smsmode_set_codes:

```

```

3282 } {
3283   \begin{stex_annotate_env} {#4} {
3284     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3285   }
3286   \stex_annotate_invisible:nnn{from}{\l_stex_import_ns_str ?\l_stex_import_name_str}{ }
3287 }
3288 \bool_set_eq:NN \l__stex_features_oldhtml_bool \l_stex_html_do_output_bool
3289 \bool_set_false:N \l_stex_html_do_output_bool
3290 }
3291 \cs_new_protected:Nn \stex_copymodule_end:n {
3292   \def \l_tmpa_cs ##1 ##2 {#1}
3293   \bool_set_eq:NN \l_stex_html_do_output_bool \l__stex_features_oldhtml_bool
3294   \tl_clear:N \l_tmpa_tl
3295   \prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3296   \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3297     \seq_map_inline:cn {c_stex_module_##1_constants}{\stex_annotate:nnn{assignment} {##1?###
3298       \l_tmpa_cs{##1}{####1}
3299       \str_if_exist:cTF {l__stex_features_copymodule_##1?####1_name_str} {
3300         \tl_put_right:Nx \l_tmpa_tl {
3301           \prop_set_from_keyval:cn {
3302             l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_#
3303           }{
3304             \exp_after:wN \prop_to_keyval:N \csname
3305               l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_#
3306             \endcsname
3307           }
3308           \seq_clear:c {
3309             l_stex_symdecl_
3310             \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_name_s
3311             _notations
3312           }
3313         }
3314         \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_features_copymodule_##1?####1_name
3315         \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{l__stex_features_
3316         \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3317           \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?###
3318           \tl_put_right:Nx \l_tmpa_tl {
3319             \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3320               \stex_invoke_symbol:n {
3321                 \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_na
3322               }
3323             }
3324           }
3325         }
3326       }{
3327         \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1_prop}
3328         \prop_put:Nnx \l_tmpa_prop { name }{\l_stex_current_copymodule_name_str / ####1 }
3329         \prop_put:Nnx \l_tmpa_prop { module }{\l_stex_current_module_str }
3330         \tl_put_right:Nx \l_tmpa_tl {
3331           \prop_set_from_keyval:cn {
3332             l_stex_symdecl_\l_stex_current_module_str ? \l_stex_current_copymodule_name_str
3333           }{
3334             \prop_to_keyval:N \l_tmpa_prop
3335           }

```

```

3336         \seq_clear:c {
3337             l_stex_symdecl_
3338             \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / #####1
3339             _notations
3340         }
3341     }
3342     \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str / #####1}
3343     \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3344         \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}
3345         \tl_put_right:Nx \l_tmpa_tl {
3346             \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3347                 \stex_invoke_symbol:n {
3348                     \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / #####1
3349                 }
3350             }
3351         }
3352     }
3353 }
3354 \tl_if_exist:cT {l__stex_features_copymodule_##1?####1_def_tl}{
3355     \stex_annotate_invisible:nnn{definiens}{\use:c{l__stex_features_copymodule_##1?####1_def_tl}}
3356 }
3357 % todo notations
3358 }}
3359 }
3360 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3361 \tl_put_left:Nx \l_tmpa_tl {
3362     \prop_set_from_keyval:cn {
3363         l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _pro
3364     }{
3365         \prop_to_keyval:N \l_stex_current_copymodule_prop
3366     }
3367 }
3368 \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl
3369 \stex_debug:nn{copymodule}{result:\meaning \l_tmpa_tl}
3370 \exp_args:Nx \stex_do_aftergroup:n {
3371     \exp_args:No \exp_not:n \l_tmpa_tl
3372 }
3373 \stex_if_smsmode:F {
3374     \end{stex_annotate_env}
3375 }
3376 }
3377
3378 \NewDocumentEnvironment {copymodule} { 0{} m m}{
3379     \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ structure }
3380     \stex_deactivate_macro:Nn \symdecl {module~environments}
3381     \stex_deactivate_macro:Nn \symdef {module~environments}
3382     \stex_deactivate_macro:Nn \notation {module~environments}
3383     \stex_reactivate_macro:N \assign
3384     \stex_reactivate_macro:N \renamedec1
3385     \stex_reactivate_macro:N \donotcopy
3386 }{
3387     \stex_copymodule_end:n {}
3388 }
3389

```

```

3390 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
3391   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ realization }
3392   \stex_deactivate_macro:Nn \symdecl {module~environments}
3393   \stex_deactivate_macro:Nn \symdef {module~environments}
3394   \stex_deactivate_macro:Nn \notation {module~environments}
3395   \stex_reactivate_macro:N \assign
3396   \stex_reactivate_macro:N \renamedekl
3397   \stex_reactivate_macro:N \donotcopy
3398 }{
3399   \stex_copymodule_end:n {
3400     \tl_if_exist:cF {
3401       l__stex_features_copymodule_##1?##2_def_tl
3402     }{
3403       \msg_error:nnxx{stex}{error/interpretmodule/nodedefiniens}{
3404         ##1?##2
3405       }{\l_stex_current_copymodule_name_str}
3406     }
3407   }
3408 }
3409
3410 \NewDocumentCommand \donotcopy { 0{} m}{
3411   \stex_import_module_uri:nn { #1 } { #2 }
3412   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3413   \seq_map_inline:Nn \l_stex_collect_imports_seq {
3414     \seq_remove_all:Nn \l__stex_features_copymodule_modules_seq { ##1 }
3415     \seq_map_inline:cn {c_stex_module_##1_constants}{
3416       \seq_remove_all:Nn \l__stex_features_copymodule_fields_seq { ##1 ? #####1 }
3417       \bool_lazy_any_p:nT {
3418         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_name_str}}
3419         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_macroname_str}}
3420         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_def_tl}}
3421       }{
3422         % TODO throw error
3423       }
3424     }
3425   }
3426
3427   \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
3428   \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_ns_str ?\l_stex_import_name_str }
3429   \prop_put:Nnx \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
3430 }
3431
3432 \NewDocumentCommand \assign { m m }{
3433   \stex_get_symbol_in_copymodule:n {#1}
3434   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
3435   \tl_set:cn {l__stex_features_copymodule_\l_stex_get_symbol_uri_str_def_tl}{#2}
3436 }
3437
3438 \keys_define:nn { stex / renamedekl } {
3439   name .str_set_x:N = \l_stex_renamedekl_name_str
3440 }
3441 \cs_new_protected:Nn \__stex_features_renamedekl_args:n {
3442   \str_clear:N \l_stex_renamedekl_name_str
3443

```

```

3444 \keys_set:nn { stex / renamedecl } { #1 }
3445 }
3446
3447 \NewDocumentCommand \renamedecl { 0{} m m}{
3448   \__stex_features_renamedecl_args:n { #1 }
3449   \stex_get_symbol_in_copymodule:n {#2}
3450   \stex_debug:nn{renamedecl}{renaming-{\l_stex_get_symbol_uri_str}-to~#3}
3451   \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
3452   \str_if_empty:NTF \l_stex_renamedecl_name_str {
3453     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3454       \l_stex_get_symbol_uri_str
3455     } }
3456   } {
3457     \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex_r
3458     \stex_debug:nn{renamedecl}{@~\l_stex_current_module_str ? \l_stex_renamedecl_name_str}
3459     \prop_set_eq:cc {l_stex_symdecl_
3460       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
3461     _prop
3462     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
3463     \seq_set_eq:cc {l_stex_symdecl_
3464       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
3465     _notations
3466     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
3467     \prop_put:cnx {l_stex_symdecl_
3468       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
3469     _prop
3470     }{ name }{ \l_stex_renamedecl_name_str }
3471     \prop_put:cnx {l_stex_symdecl_
3472       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
3473     _prop
3474     }{ module }{ \l_stex_current_module_str }
3475     \exp_args:NNx \seq_put_left:Nn \l__stex_features_copymodule_fields_seq {
3476       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
3477     }
3478     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3479       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
3480     } }
3481   }
3482 }
3483 %\NewDocumentCommand \notation_in_copymodules: { 0{} m } {
3484 % \_stex_notation_args:n { #1 }
3485 % \tl_clear:N \l_stex_symdecl_definiens_tl
3486 % \stex_get_symbol_in_copymodule:n { #2 }
3487 % \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
3488 % % todo
3489 %}
3490 \stex_deactivate_macro:Nn \assign {copymodules}
3491 \stex_deactivate_macro:Nn \renamedecl {copymodules}
3492 \stex_deactivate_macro:Nn \donotcopy {copymodules}
3493
3494
3495 \seq_new:N \l_stex_implicit_morphisms_seq
3496 \NewDocumentCommand \implicitmorphism { 0{} m m}{
3497   \stex_import_module_uri:nn { #1 } { #2 }

```

```

3498 \stex_debug:nn{implicits}{
3499   Implicit~morphism:~
3500   \l_stex_module_ns_str ? \l__stex_features_name_str
3501 }
3502 \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
3503   \l_stex_module_ns_str ? \l__stex_features_name_str
3504 }{
3505   \msg_error:nnn{stex}{error/conflictingmodules}{
3506     \l_stex_module_ns_str ? \l__stex_features_name_str
3507   }
3508 }
3509
3510 % TODO
3511
3512
3513
3514 \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
3515   \l_stex_module_ns_str ? \l__stex_features_name_str
3516 }
3517 }
3518

```

## 32.2 The feature environment

structural@feature

```

3519
3520 \NewDocumentEnvironment{structural@feature}{ m m m }{
3521   \stex_if_in_module:F {
3522     \msg_set:nnn{stex}{error/nomodule}{
3523       Structural~Feature~has~to~occur~in~a~module:\\
3524       Feature~#2~of~type~#1\\
3525       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
3526     }
3527     \msg_error:nn{stex}{error/nomodule}
3528   }
3529
3530   \str_set:Nx \l_stex_module_name_str {
3531     \prop_item:Nn \l_stex_current_module_prop
3532     { name } / #2 - feature
3533   }
3534
3535   \str_set:Nx \l_stex_module_ns_str {
3536     \prop_item:Nn \l_stex_current_module_prop
3537     { ns }
3538   }
3539
3540
3541   \str_clear:N \l_tmpa_str
3542   \seq_clear:N \l_tmpa_seq
3543   \tl_clear:N \l_tmpa_tl
3544   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
3545     origname = #2,
3546     name      = \l_stex_module_name_str ,
3547     ns        = \l_stex_module_ns_str ,

```

```

3548 imports = \exp_not:o { \l_tmpa_seq } ,
3549 constants = \exp_not:o { \l_tmpa_seq } ,
3550 content = \exp_not:o { \l_tmpa_tl } ,
3551 file = \exp_not:o { \g_stex_currentfile_seq } ,
3552 lang = \l_stex_module_lang_str ,
3553 sig = \l_tmpa_str ,
3554 meta = \l_tmpa_str ,
3555 feature = #1 ,
3556 }
3557
3558 \stex_if_smsmode:TF {
3559   \stex_smsmode_set_codes:
3560 } {
3561   \begin{stex_annotate_env}{ feature:#1 }{}
3562   \stex_annotate_invisible:nnn{header}{}{ #3 }
3563 }
3564 }{
3565   \str_set:Nx \l_tmpa_str {
3566     c_stex_feature_
3567     \prop_item:Nn \l_stex_current_module_prop { ns } ?
3568     \prop_item:Nn \l_stex_current_module_prop { name }
3569     _prop
3570   }
3571   \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
3572   \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
3573   \stex_if_smsmode:TF {
3574     \exp_args:Nx \stex_add_to_sms:n {
3575       \prop_gset_from_keyval:cn {
3576         c_stex_feature_
3577         \prop_item:Nn \l_stex_current_module_prop { ns } ?
3578         \prop_item:Nn \l_stex_current_module_prop { name }
3579         _prop
3580       } {
3581         origname = #2,
3582         name = \prop_item:cn { \l_tmpa_str } { name } ,
3583         ns = \prop_item:cn { \l_tmpa_str } { ns } ,
3584         imports = \prop_item:cn { \l_tmpa_str } { imports } ,
3585         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
3586         content = \prop_item:cn { \l_tmpa_str } { content } ,
3587         file = \prop_item:cn { \l_tmpa_str } { file } ,
3588         lang = \prop_item:cn { \l_tmpa_str } { lang } ,
3589         sig = \prop_item:cn { \l_tmpa_str } { sig } ,
3590         meta = \prop_item:cn { \l_tmpa_str } { meta } ,
3591         feature = \prop_item:cn { \l_tmpa_str } { feature }
3592       }
3593     }
3594   } {
3595     \end{stex_annotate_env}
3596   }
3597 }
3598

```



## 32.3 Features

structure

```

3599
3600 \prop_new:N \l_stex_all_structures_prop
3601
3602 \keys_define:nn { stex / features / structure } {
3603   name          .str_set_x:N = \l__stex_features_structure_name_str ,
3604 }
3605
3606 \cs_new_protected:Nn \__stex_features_structure_args:n {
3607   \str_clear:N \l__stex_features_structure_name_str
3608   \keys_set:nn { stex / features / structure } { #1 }
3609 }
3610
3611 %\stex_new_feature:nnnn { structure } { 0{} m } {
3612 %   \__stex_features_structure_args:n { ##1 }
3613 %   \str_if_empty:NT \l__stex_features_structure_name_str {
3614 %     \str_set:Nx \l__stex_features_structure_name_str { ##2 }
3615 %   }
3616 %} {
3617 %
3618 %}
3619
3620 \NewDocumentEnvironment{mathstructure}{ 0{} m }{
3621   \__stex_features_structure_args:n { #1 }
3622   \str_if_empty:NT \l__stex_features_structure_name_str {
3623     \str_set:Nx \l__stex_features_structure_name_str { #2 }
3624   }
3625   \exp_args:Nnnx
3626   \begin{structural@feature}{ structure }
3627     { \l__stex_features_structure_name_str }{}
3628     \seq_clear:N \l_tmpa_seq
3629     \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
3630
3631   }{
3632     \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
3633     \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
3634     \str_set:Nx \l_tmpa_str {
3635       \prop_item:Nn \l_stex_current_module_prop { ns } ?
3636       \prop_item:Nn \l_stex_current_module_prop { name }
3637     }
3638     \seq_map_inline:Nn \l_tmpa_seq {
3639       \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
3640     }
3641     \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
3642     \exp_args:Nnx
3643     \AddToHookNext { env / mathstructure / after }{
3644       \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}}{
3645         \stex_term_math_oms:nnnn { \l_tmpa_str }{}{0}{}
3646       }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
3647     \STEXexport {
3648       \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3649       { \prop_item:Nn \l_stex_current_module_prop { origname } }

```

```

3650         {\l_tmpa_str}
3651         \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3652         {#2}{\l_tmpa_str}
3653     % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3654     % \prop_item:Nn \l_stex_current_module_prop { origname },
3655     % \l_tmpa_str
3656     % }
3657     % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3658     % #2,\l_tmpa_str
3659     % }
3660     % \tl_set:cx { #2 } {
3661     % \stex_invoke_structure:n { \l_tmpa_str }
3662     % }
3663 }
3664
3665 \end{structural@feature}
3666 % \g_stex_last_feature_prop
3667 }

```

\instantiate

```

3668 \seq_new:N \l__stex_features_structure_field_seq
3669 \str_new:N \l__stex_features_structure_field_str
3670 \str_new:N \l__stex_features_structure_def_tl
3671 \prop_new:N \l__stex_features_structure_prop
3672 \NewDocumentCommand \instantiate { m O{} m }{
3673     \stex_smsmode_set_codes:
3674     \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
3675     \prop_set_eq:Nc \l__stex_features_structure_prop {
3676         c_stex_feature_\l_tmpa_str _prop
3677     }
3678     \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
3679     \seq_map_inline:Nn \l__stex_features_structure_field_seq {
3680         \seq_set_split:Nnn \l_tmpa_seq={}{ ##1 }
3681         \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3682             \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
3683             \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
3684             {!} \l_tmpa_tl
3685             \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3686                 \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
3687                 \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3688                 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3689             }{
3690                 \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
3691                 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3692                 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
3693                 \l_tmpa_tl
3694                 \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3695                     \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
3696                     \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3697                 }{
3698                     \tl_clear:N \l_tmpb_tl
3699                 }
3700             }
3701         }{

```

```

3702     \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
3703     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3704         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
3705         \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
3706         \tl_clear:N \l_tmpa_tl
3707     }{
3708         % TODO throw error
3709     }
3710 }
3711 % \l_tmpa_str: name
3712 % \l_tmpa_tl: definiens
3713 % \l_tmpb_tl: notation
3714 \tl_if_empty:NT \l__stex_features_structure_field_str {
3715     % TODO throw error
3716 }
3717 \str_clear:N \l_tmpb_str
3718
3719 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3720 \seq_map_inline:Nn \l_tmpa_seq {
3721     \seq_set_split:Nnn \l_tmpb_seq ? { ###1 }
3722     \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
3723     \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
3724         \seq_map_break:n {
3725             \str_set:Nn \l_tmpb_str { ###1 }
3726         }
3727     }
3728 }
3729 \prop_get:cnN { l_stex_symdecl_ \l_tmpb_str _prop } {args}
3730 \l_tmpb_str
3731
3732 \tl_if_empty:NTF \l_tmpb_tl {
3733     \tl_if_empty:NF \l_tmpa_tl {
3734         \exp_args:Nx \use:n {
3735             \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fe
3736         }
3737     }
3738 }{
3739     \tl_if_empty:NTF \l_tmpa_tl {
3740         \exp_args:Nx \use:n {
3741             \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\
3742         }
3743     }
3744 }{
3745     \exp_args:Nx \use:n {
3746         \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
3747         \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
3748     }
3749 }
3750 }
3751 % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
3752 % \prop_item:Nn \l_stex_current_module_prop {name} ?
3753 % #3/\l__stex_features_structure_field_str
3754 % \par
3755 % \expandafter\present\csname

```

```

3756 %      l_stex_symdecl_
3757 %      \prop_item:Nn \l_stex_current_module_prop {ns} ?
3758 %      \prop_item:Nn \l_stex_current_module_prop {name} ?
3759 %      #3/\l__stex_features_structure_field_str
3760 %      _prop
3761 %      \endcsname
3762 }
3763
3764 \tl_clear:N \l__stex_features_structure_def_tl
3765
3766 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3767 \seq_map_inline:Nn \l_tmpa_seq {
3768   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3769   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3770   \exp_args:Nx \use:n {
3771     \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
3772
3773     }
3774   }
3775
3776   \prop_if_exist:cF {
3777     l_stex_symdecl_
3778     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3779     \prop_item:Nn \l_stex_current_module_prop {name} ?
3780     #3/\l_tmpa_str
3781     _prop
3782   }{
3783     \prop_get:cnN { l_stex_symdecl_ ##1 _prop } {args}
3784     \l_tmpb_str
3785     \exp_args:Nx \use:n {
3786       \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
3787     }
3788   }
3789 }
3790
3791 \symdecl*[type={\STEXsymbol{module-type}}{
3792   \_stex_term_math_oms:nnnn {
3793     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3794     \prop_item:Nn \l__stex_features_structure_prop {name}
3795     }{}{0}{}
3796   }{}{#3}
3797
3798 % TODO: -> sms file
3799
3800 \tl_set:cx{ #3 }{
3801   \stex_invoke_structure:nnn {
3802     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3803     \prop_item:Nn \l_stex_current_module_prop {name} ? #3
3804   } {
3805     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3806     \prop_item:Nn \l__stex_features_structure_prop {name}
3807   }
3808 }
3809

```

```
3810 }
```

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

```
\stex_invoke_structure:nnn
```

```
3811 % #1: URI of the instance
3812 % #2: URI of the instantiated module
3813 \cs_new_protected:Nn \stex_invoke_structure:nnn {
3814   \tl_if_empty:nTF{ #3 }{
3815     \prop_set_eq:Nc \l__stex_features_structure_prop {
3816       c_stex_feature_ #2 _prop
3817     }
3818     \tl_clear:N \l_tmpa_tl
3819     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3820     \seq_map_inline:Nn \l_tmpa_seq {
3821       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3822       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3823       \cs_if_exist:cT {
3824         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
3825       }{
3826         \tl_if_empty:NF \l_tmpa_tl {
3827           \tl_put_right:Nn \l_tmpa_tl {,}
3828         }
3829         \tl_put_right:Nx \l_tmpa_tl {
3830           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
3831         }
3832       }
3833     }
3834     \exp_args:No \mathstrut \l_tmpa_tl
3835   }{
3836     \stex_invoke_symbol:n{#1/#3}
3837   }
3838 }
```

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

```
3839 \endpackage
```

## Chapter 33

# STEX -Statements Implementation

```
3840 <*package>
3841
3842 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3843
3844 \protected\def\ignorespacesandpars{
3845   \begingroup\catcode13=10\relax
3846   \@ifnextchar\par{
3847     \endgroup\expandafter\ignorespacesandpars\@gobble
3848   }{
3849     \endgroup
3850   }
3851 }
3852
3853 <@@=stex_statements>
3854
3855   Warnings and error messages
3856
3857 \titleemph
3858
3859 \def\titleemph#1{\textbf{#1}}
3860
3861 (End definition for \titleemph. This function is documented on page ??.)
```

### 33.1 Definitions

```
definiendum
3862 \keys_define:nn {stex / definiendum }{
3863   post      .tl_set:N      = \l__stex_statements_definiendum_post_tl,
3864   root      .str_set_x:N   = \l__stex_statements_definiendum_root_str,
3865   gfa       .str_set_x:N   = \l__stex_statements_definiendum_gfa_str
3866 }
3867 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
3868   \str_clear:N \l__stex_statements_definiendum_root_str
3869   \tl_clear:N \l__stex_statements_definiendum_post_tl
3870   \str_clear:N \l__stex_statements_definiendum_gfa_str
3871 }
```

```

3865 \keys_set:nn { stex / definiendum } { #1 }
3866 }
3867 \NewDocumentCommand \definiendum { 0{} m m } {
3868   \__stex_statements_definiendum_args:n { #1 }
3869   \stex_get_symbol:n { #2 }
3870   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3871   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
3872     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
3873       \tl_set:Nn \l_tmpa_tl { #3 }
3874     } {
3875       \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
3876       \tl_set:Nn \l_tmpa_tl {
3877         \l__stex_statements_definiendum_root_str\l__stex_statements_definiendum_post_tl
3878       }
3879     }
3880   } {
3881     \tl_set:Nn \l_tmpa_tl { #3 }
3882   }
3883
3884   % TODO root
3885   \rustex_if:TF {
3886     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
3887   } {
3888     \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
3889   }
3890 }
3891 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

## definame

```

3892
3893 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3894
3895 \NewDocumentCommand \definame { 0{} m } {
3896   \__stex_statements_definiendum_args:n { #1 }
3897   % TODO: root
3898   \stex_get_symbol:n { #2 }
3899   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3900   \str_set:Nx \l_tmpa_str {
3901     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3902   }
3903   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3904   \rustex_if:TF {
3905     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3906       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3907     }
3908   } {
3909     \defemph@uri {
3910       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3911     } { \l_stex_get_symbol_uri_str }
3912   }
3913 }
3914 \stex_deactivate_macro:Nn \definame {definition~environments}

```

```

3915
3916 \NewDocumentCommand \Definame { 0{ } m } {
3917   \_stex_statements_definiendum_args:n { #1 }
3918   \stex_get_symbol:n { #2 }
3919   \str_set:Nx \l_tmpa_str {
3920     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3921   }
3922   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3923   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3924   \rustex_if:TF {
3925     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3926       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3927     }
3928   } {
3929     \defemph@uri {
3930       \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
3931     } { \l_stex_get_symbol_uri_str }
3932   }
3933 }
3934 \stex_deactivate_macro:Nn \Definame {definition-environments}
3935
3936 \NewDocumentCommand \Symname { 0{ } m } {
3937   \stex_symname_args:n { #1 }
3938   \stex_get_symbol:n { #2 }
3939   \str_set:Nx \l_tmpa_str {
3940     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3941   }
3942   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3943   \let\compemph_uri_prev:\compemph@uri
3944   \let\compemph@uri\symrefemph@uri
3945   \exp_args:NNx \use:nn
3946   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }! [
3947     \exp_after:wN \stex_capitalize:n \l_tmpa_str
3948     \l_stex_symname_post_str
3949   ] }
3950   \let\compemph@uri\compemph_uri_prev:
3951 }

```

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

## sdefinition

```

3952
3953 \keys_define:nn {stex / sdefinition }{
3954   type      .str_set_x:N = \sdefinitiontype,
3955   id        .str_set_x:N = \sdefinitionid,
3956   name      .str_set_x:N = \sdefinitionname,
3957   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
3958   title     .tl_set:N     = \sdefinitiontitle
3959 }
3960 \cs_new_protected:Nn \_stex_statements_sdefinition_args:n {
3961   \str_clear:N \sdefinitiontype
3962   \str_clear:N \sdefinitionid
3963   \str_clear:N \sdefinitionname
3964   \clist_clear:N \l__stex_statements_sdefinition_for_clist

```



```

3965 \tl_clear:N \sdefinitiontitle
3966 \keys_set:nn { stex / sdefinition }{ #1 }
3967 }
3968
3969 \NewDocumentEnvironment{sdefinition}{0{}}{
3970   \__stex_statements_sdefinition_args:n{ #1 }
3971   \stex_reactivate_macro:N \definiendum
3972   \stex_reactivate_macro:N \definame
3973   \stex_reactivate_macro:N \Definame
3974   \stex_smsmode_set_codes:
3975   \stex_if_smsmode:TF {
3976     \stex_smsmode_set_codes:
3977   } {
3978     \seq_clear:N \l_tmpa_seq
3979     \seq_clear:N
3980     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
3981       \str_if_eq:nnF{ ##1 }{ }{
3982         \stex_get_symbol:n { ##1 }
3983         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
3984           \l_stex_get_symbol_uri_str
3985         }
3986       }
3987     }
3988     \exp_args:Nnnx
3989     \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {,}}
3990     \str_if_empty:NF \sdefinitiontype {
3991       \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{ }
3992     }
3993   }
3994   \clist_set:No \l_tmpa_clist \sdefinitiontype
3995   \tl_clear:N \l_tmpa_tl
3996   \clist_map_inline:Nn \l_tmpa_clist {
3997     \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
3998       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
3999     }
4000   }
4001   \tl_if_empty:NTF \l_tmpa_tl {
4002     \__stex_statements_sdefinition_start:
4003   }{
4004     \l_tmpa_tl
4005   }
4006   \stex_ref_new_doc_target:n \sdefinitionid
4007 }{
4008   \clist_set:No \l_tmpa_clist \sdefinitiontype
4009   \tl_clear:N \l_tmpa_tl
4010   \clist_map_inline:Nn \l_tmpa_clist {
4011     \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
4012       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
4013     }
4014   }
4015   \str_if_empty:NF \sdefinitionname { \symdecl*{\sdefinitionname} }
4016   \tl_if_empty:NTF \l_tmpa_tl {
4017     \__stex_statements_sdefinition_end:
4018   }{

```

```

4019   \l_tmpa_tl
4020 }
4021 \stex_if_smsmode:F {
4022   \end{stex_annotate_env}
4023 }
4024 }

```

`\stexpatchdefinition`

```

4025 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
4026   \par\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
4027     ~(\sdefinitiontitle)
4028   }~}
4029 }
4030 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\par\medskip}
4031
4032 \newcommand\stexpatchdefinition[3][] {
4033   \str_set:Nx \l_tmpa_str{ #1 }
4034   \str_if_empty:NTF \l_tmpa_str {
4035     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
4036     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
4037   }{
4038     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
4039     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
4040   }
4041 }

```

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

`\inlinedef` inline:

```

4042 \keys_define:nn {stex / inlinedef }{
4043   type      .str_set_x:N = \sdefinitiontype,
4044   id        .str_set_x:N = \sdefinitionid,
4045   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4046   name      .str_set_x:N = \sdefinitionname
4047 }
4048 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
4049   \str_clear:N \sdefinitiontype
4050   \str_clear:N \sdefinitionid
4051   \str_clear:N \sdefinitionname
4052   \clist_clear:N \l__stex_statements_sdefinition_for_clist
4053   \keys_set:nn { stex / inlinedef }{ #1 }
4054 }
4055 \NewDocumentCommand \inlinedef { 0{} m } {
4056   \beginngroup
4057   \__stex_statements_inlinedef_args:n{ #1 }
4058   \stex_ref_new_doc_target:n \definitionid
4059   \stex_reactivate_macro:N \definiendum
4060   \stex_reactivate_macro:N \definame
4061   \stex_if_smsmode:TF{
4062     \stex_smsmode_set_codes:
4063     \str_if_empty:NF \sdefinitionname { \symdecl*{\sdefinitionname} }
4064   }{
4065     \seq_clear:N \l_tmpa_seq
4066     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {

```

```

4067     \str_if_eq:nnF{ ##1 }{}{
4068       \stex_get_symbol:n { ##1 }
4069       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4070         \l_stex_get_symbol_uri_str
4071       }
4072     }
4073   }
4074   \exp_args:Nnx
4075   \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
4076     \str_if_empty:NF \sdefinitiontype {
4077       \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{}
4078     }
4079     #2
4080     \str_if_empty:NF \sdefinitionname { \symdecl*{\sdefinitionname} }
4081   }
4082 }
4083 \endgroup
4084 }

```

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

## 33.2 Assertions

**sassertion**

```

4085
4086 \keys_define:nn {stex / sassertion }{
4087   type      .str_set_x:N = \sassertiontype,
4088   id        .str_set_x:N = \sassertionid,
4089   title     .tl_set:N    = \sassertiontitle ,
4090   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4091   name      .str_set_x:N = \sassertionname
4092 }
4093 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
4094   \str_clear:N \sassertiontype
4095   \str_clear:N \sassertionid
4096   \str_clear:N \sassertionname
4097   \clist_clear:N \l__stex_statements_sassertion_for_clist
4098   \tl_clear:N \sassertiontitle
4099   \keys_set:nn { stex / sassertion }{ #1 }
4100 }
4101
4102 %\tl_new:N \g__stex_statements_aftergroup_tl
4103
4104 \NewDocumentEnvironment{sassertion}{0{}}{
4105   \__stex_statements_sassertion_args:n{ #1 }
4106   \stex_smsmode_set_codes:
4107   \stex_if_smsmode:TF {
4108     \stex_smsmode_set_codes:
4109   } {
4110     \seq_clear:N \l_tmpa_seq
4111     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4112       \str_if_eq:nnF{ ##1 }{}{
4113         \stex_get_symbol:n { ##1 }

```

```

4114         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4115             \l_stex_get_symbol_uri_str
4116         }
4117     }
4118 }
4119 \exp_args:Nnnx
4120 \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}
4121 \str_if_empty:NF \sassertiontype {
4122     \stex_annotate_invisible:nnn{type}{\sassertiontype}{ }
4123 }
4124 }
4125 \clist_set:Nn \l_tmpa_clist \sassertiontype
4126 \tl_clear:N \l_tmpa_tl
4127 \clist_map_inline:Nn \l_tmpa_clist {
4128     \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
4129         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
4130     }
4131 }
4132 \tl_if_empty:NTF \l_tmpa_tl {
4133     \__stex_statements_sassertion_start:
4134 }{
4135     \l_tmpa_tl
4136 }
4137 \stex_ref_new_doc_target:n \sassertionid
4138 }{
4139     \clist_set:Nn \l_tmpa_clist \sassertiontype
4140     \tl_clear:N \l_tmpa_tl
4141     \clist_map_inline:Nn \l_tmpa_clist {
4142         \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
4143             \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
4144         }
4145     }
4146     \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
4147     \tl_if_empty:NTF \l_tmpa_tl {
4148         \__stex_statements_sassertion_end:
4149     }{
4150         \l_tmpa_tl
4151     }
4152     \stex_if_smsmode:F {
4153         \end{stex_annotate_env}
4154     }
4155 }

```

\stexpatchassertion

```

4156
4157 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
4158     \par\noindent\titllemph{Assertion~\tl_if_empty:NF \sassertiontitle {
4159         (\sassertiontitle)
4160     }~}
4161 }
4162 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
4163
4164 \newcommand\stexpatchassertion[3] [] {
4165     \str_set:Nx \l_tmpa_str{ #1 }

```

```

4166 \str_if_empty:NTF \l_tmpa_str {
4167   \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
4168   \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
4169 }{
4170   \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
4171   \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
4172 }
4173 }

```

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

`\inlineass` inline:

```

4174 \keys_define:nn {stex / inlineass }{
4175   type      .str_set_x:N = \sassertiontype,
4176   id        .str_set_x:N = \sassertionid,
4177   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4178   name      .str_set_x:N = \sassertionname
4179 }
4180 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
4181   \str_clear:N \sassertiontype
4182   \str_clear:N \sassertionid
4183   \str_clear:N \sassertionname
4184   \clist_clear:N \l__stex_statements_sassertion_for_clist
4185   \keys_set:nn { stex / inlineass }{ #1 }
4186 }
4187 \NewDocumentCommand \inlineass { 0{} m } {
4188   \begingroup
4189   \__stex_statements_inlineass_args:n{ #1 }
4190   \stex_ref_new_doc_target:n \sassertionid
4191   \stex_if_smsmode:TF{
4192     \stex_smsmode_set_codes:
4193     \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
4194   }{
4195     \seq_clear:N \l_tmpa_seq
4196     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4197       \str_if_eq:nnF{ ##1 }{}{
4198         \stex_get_symbol:n { ##1 }
4199         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4200           \l_stex_get_symbol_uri_str
4201         }
4202       }
4203     }
4204     \exp_args:Nnx
4205     \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{
4206       \str_if_empty:NF \sassertiontype {
4207         \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
4208       }
4209       #2
4210       \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
4211     }
4212   }
4213   \endgroup
4214 }

```

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

## 33.3 Examples

sexample

```

4215
4216 \keys_define:nn {stex / sexample }{
4217   type      .str_set_x:N = \exampletype,
4218   id        .str_set_x:N = \sexampleid,
4219   title     .tl_set:N     = \sexampletile,
4220   for       .clist_set:N  = \l__stex_statements_sexample_for_clist,
4221 }
4222 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
4223   \str_clear:N \sexampletype
4224   \str_clear:N \sexampleid
4225   \tl_clear:N \sexampletile
4226   \clist_clear:N \l__stex_statements_sexample_for_clist
4227   \keys_set:nn { stex / sexample }{ #1 }
4228 }
4229
4230 \NewDocumentEnvironment{sexample}{0{}}{
4231   \__stex_statements_sexample_args:n{ #1 }
4232   \stex_smsmode_set_codes:
4233   \stex_if_smsmode:TF {
4234     \stex_smsmode_set_codes:
4235   } {
4236     \seq_clear:N \l_tmpa_seq
4237     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4238       \str_if_eq:nnF{ ##1 }{}{
4239         \stex_get_symbol:n { ##1 }
4240         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4241           \l_stex_get_symbol_uri_str
4242         }
4243       }
4244     }
4245     \exp_args:Nnnx
4246     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
4247     \str_if_empty:NF \sexampletype {
4248       \stex_annotate_invisible:nnn{type}{\sexampletype}{}
4249     }
4250   }
4251   \stex_ref_new_doc_target:n \sexampleid
4252   \clist_set:No \l_tmpa_clist \sexampletype
4253   \tl_clear:N \l_tmpa_tl
4254   \clist_map_inline:Nn \l_tmpa_clist {
4255     \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
4256       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
4257     }
4258   }
4259   \tl_if_empty:NTF \l_tmpa_tl {
4260     \__stex_statements_sexample_start:
4261   }{
4262     \l_tmpa_tl
4263   }
4264 }{
4265   \clist_set:No \l_tmpa_clist \sexampletype

```

```

4266 \tl_clear:N \l_tmpa_tl
4267 \clist_map_inline:Nn \l_tmpa_clist {
4268   \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
4269     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
4270   }
4271 }
4272 \tl_if_empty:NTF \l_tmpa_tl {
4273   \__stex_statements_sexample_end:
4274 }{
4275   \l_tmpa_tl
4276 }
4277 \stex_if_smsmode:F {
4278   \end{stex_annotate_env}
4279 }
4280 }

```

`\stexpatchexample`

```

4281
4282 \cs_new_protected:Nn \__stex_statements_sexample_start: {
4283   \par\noindent\titleemph{Example~\tl_if_empty:NF \sexamplename {
4284     (\sexamplename)
4285   }~}
4286 }
4287 \cs_new_protected:Nn \__stex_statements_sexample_end: {\par\medskip}
4288
4289 \newcommand\stexpatchexample[3]{} {
4290   \str_set:Nx \l_tmpa_str{ #1 }
4291   \str_if_empty:NTF \l_tmpa_str {
4292     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
4293     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
4294   }{
4295     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
4296     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
4297   }
4298 }

```

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

`\inlineex` inline:

```

4299 \keys_define:nn {stex / inlineex }{
4300   type      .str_set_x:N = \sexamplename,
4301   id        .str_set_x:N = \sexampleid,
4302   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
4303   name      .str_set_x:N = \sexamplename
4304 }
4305 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
4306   \str_clear:N \sexamplename
4307   \str_clear:N \sexampleid
4308   \str_clear:N \sexamplename
4309   \clist_clear:N \l__stex_statements_sexample_for_clist
4310   \keys_set:nn { stex / inlineex }{ #1 }
4311 }
4312 \NewDocumentCommand \inlineex { 0{} m } {
4313   \begingroup

```

```

4314 \__stex_statements_inlineex_args:n{ #1 }
4315 \stex_ref_new_doc_target:n \sexampleid
4316 \stex_if_smsmode:TF{
4317   \stex_smsmode_set_codes:
4318   \str_if_empty:NF \sexamplename { \symdecl*{\examplename} }
4319 }{
4320   \seq_clear:N \l_tmpa_seq
4321   \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4322     \str_if_eq:nnF{ ##1 }{}{
4323       \stex_get_symbol:n { ##1 }
4324       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4325         \l_stex_get_symbol_uri_str
4326       }
4327     }
4328   }
4329   \exp_args:Nnx
4330   \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpa_seq {},,}{
4331     \str_if_empty:NF \sexamplotype {
4332       \stex_annotate_invisible:nnn{type}{\sexamplotype}{
4333     }
4334     #2
4335     \str_if_empty:NF \sexamplename { \symdecl*{\sexamplename} }
4336   }
4337 }
4338 \endgroup
4339 }

```

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

## 33.4 Logical Paragraphs

sparagraph

```

4340 \keys_define:nn { stex / sparagraph } {
4341   id      .str_set:x:N = \sparagraphid ,
4342   title   .tl_set:N    = \l_stex_sparagraph_title_tl ,
4343   type    .str_set:x:N = \sparagraphtype ,
4344   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
4345   from    .tl_set:N    = \sparagraphfrom ,
4346   to      .tl_set:N    = \sparagraphto ,
4347   start   .tl_set:N    = \l_stex_sparagraph_start_tl ,
4348   name    .str_set:N    = \sparagraphname
4349 }
4350
4351 \cs_new_protected:Nn \stex_sparagraph_args:n {
4352   \tl_clear:N \l_stex_sparagraph_title_tl
4353   \tl_clear:N \sparagraphfrom
4354   \tl_clear:N \sparagraphto
4355   \tl_clear:N \l_stex_sparagraph_start_tl
4356   \str_clear:N \sparagraphid
4357   \str_clear:N \sparagraphtype
4358   \clist_clear:N \l__stex_statements_sparagraph_for_clist
4359   \str_clear:N \sparagraphname
4360   \keys_set:nn { stex / sparagraph }{ #1 }

```



```

4361 }
4362 \newif\if@in@omtext\@in@omtextfalse
4363
4364 \NewDocumentEnvironment {sparagraph} { 0{} } {
4365   \stex_sparagraph_args:n { #1 }
4366   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4367     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
4368   }{
4369     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
4370   }
4371   \@in@omtexttrue
4372   \stex_if_smsmode:TF {
4373     \stex_smsmode_set_codes:
4374   } {
4375     \seq_clear:N \l_tmpa_seq
4376     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
4377       \str_if_eq:nnF{ ##1 }{}{
4378         \stex_get_symbol:n { ##1 }
4379         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4380           \l_stex_get_symbol_uri_str
4381         }
4382       }
4383     }
4384     \exp_args:Nnnx
4385     \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
4386     \str_if_empty:NF \sparagraphtype {
4387       \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}
4388     }
4389     \str_if_empty:NF \sparagraphfrom {
4390       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
4391     }
4392     \str_if_empty:NF \sparagraphto {
4393       \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
4394     }
4395   }
4396   \clist_set:No \l_tmpa_clist \sparagraphtype
4397   \tl_clear:N \l_tmpa_tl
4398   \clist_map_inline:Nn \l_tmpa_clist {
4399     \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
4400       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
4401     }
4402   }
4403   \tl_if_empty:NTF \l_tmpa_tl {
4404     \__stex_statements_sparagraph_start:
4405   }{
4406     \l_tmpa_tl
4407   }
4408   \stex_ref_new_doc_target:n \sparagraphid
4409   \ignorespacesandpars
4410 }{
4411   \clist_set:No \l_tmpa_clist \sparagraphtype
4412   \tl_clear:N \l_tmpa_tl
4413   \clist_map_inline:Nn \l_tmpa_clist {
4414     \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{

```

```

4415     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
4416   }
4417 }
4418 \str_if_empty:NF \sparagraphname { \symdecl*{\sparagraphname} }
4419 \tl_if_empty:NTF \l_tmpa_tl {
4420   \__stex_statements_sparagraph_end:
4421 }{
4422   \l_tmpa_tl
4423 }
4424 \stex_if_smsmode:F {
4425   \end{stex_annotate_env}
4426 }
4427 }

```

## \stexpatchparagraph

```

4428
4429 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
4430   \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4431     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
4432       \titleemph{\l_stex_sparagraph_title_tl}:~
4433     }
4434   }{
4435     \titleemph{\l_stex_sparagraph_start_tl}~
4436   }
4437 }
4438 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
4439
4440 \newcommand\stexpatchparagraph[3] [] {
4441   \str_set:Nx \l_tmpa_str{ #1 }
4442   \str_if_empty:NTF \l_tmpa_str {
4443     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
4444     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
4445   }{
4446     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
4447     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
4448   }
4449 }
4450
4451 \keys_define:nn { stex / inlinepara } {
4452   id      .str_set_x:N = \sparagraphid ,
4453   type    .str_set_x:N = \sparagraphtype ,
4454   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
4455   from    .tl_set:N    = \sparagraphfrom ,
4456   to      .tl_set:N    = \sparagraphto ,
4457   name    .str_set:N   = \sparagraphname
4458 }
4459 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
4460   \tl_clear:N \sparagraphfrom
4461   \tl_clear:N \sparagraphto
4462   \str_clear:N \sparagraphid
4463   \str_clear:N \sparagraphtype
4464   \clist_clear:N \l__stex_statements_sparagraph_for_clist
4465   \str_clear:N \sparagraphname
4466   \keys_set:nn { stex / inlinepara }{ #1 }

```

```

4467 }
4468 \NewDocumentCommand \inlinepara { 0{} m } {
4469   \beginngroup
4470     \__stex_statements_inlinepara_args:n{ #1 }
4471     \stex_ref_new_doc_target:n \sparagraphid
4472     \stex_if_smsmode:TF{
4473       \stex_smsmode_set_codes:
4474       \str_if_empty:NF \sparagraphname { \symdecl*\sparagraphname } }
4475   }{
4476     \seq_clear:N \l_tmpa_seq
4477     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
4478       \str_if_eq:nnF{ ##1 }{}{
4479         \stex_get_symbol:n { ##1 }
4480         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4481           \l_stex_get_symbol_uri_str
4482         }
4483       }
4484     }
4485     \exp_args:Nnx
4486     \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
4487       \str_if_empty:NF \sparagraphtype {
4488         \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}}
4489     }
4490     \str_if_empty:NF \sparagraphfrom {
4491       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}}
4492     }
4493     \str_if_empty:NF \sparagraphto {
4494       \stex_annotate_invisible:nnn{to}{\sparagraphto}{}}
4495     }
4496     #2
4497     \str_if_empty:NF \sparagraphname { \symdecl*\sparagraphname } }
4498   }
4499 }
4500 \endgroup
4501 }
4502

```

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

**symboldoc**

```

4503 \NewDocumentEnvironment{symboldoc}{ m }{
4504   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
4505   \seq_clear:N \l_tmpb_seq
4506   \seq_map_inline:Nn \l_tmpa_seq {
4507     \str_if_eq:nnF{ ##1 }{}{
4508       \stex_get_symbol:n { ##1 }
4509       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
4510         \l_stex_get_symbol_uri_str
4511       }
4512     }
4513   }
4514   \par
4515   \exp_args:Nnnx
4516   \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}

```

```
4517 }{  
4518   \end{stex_annotate_env}  
4519 }  
  
4520 \</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>

```
4521 <*package>
4522 <@@=stex_sproof>
4523
4524 %%%%%%%%%%% sproof.dtx %%%%%%%%%%%
4525
```

### 34.2 Proofs

We first define some keys for the proof environment.

```
4526 \keys_define:nn { stex / spf } {
4527   id          .str_set:N = \l__stex_sproof_spf_id_str,
4528   display     .tl_set:N  = \l__stex_sproof_spf_display_tl,
4529   for         .tl_set:N  = \l__stex_sproof_spf_for_tl ,
4530   from       .tl_set:N  = \l__stex_sproof_spf_from_tl ,
4531   proofend    .tl_set:N  = \l__stex_sproof_spf_proofend_tl,
4532   type       .tl_set:N  = \l__stex_sproof_spf_type_tl,
4533   title      .tl_set:N  = \l__stex_sproof_spf_title_tl,
4534   continues   .tl_set:N  = \l__stex_sproof_spf_continues_tl,
4535   functions   .tl_set:N  = \l__stex_sproof_spf_functions_tl,
4536   method     .tl_set:N  = \l__stex_sproof_spf_method_tl
4537 }
4538 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
4539   \str_clear:N \l__stex_sproof_spf_id_str
4540   \tl_clear:N \l__stex_sproof_spf_display_tl
4541   \tl_clear:N \l__stex_sproof_spf_for_tl
4542   \tl_clear:N \l__stex_sproof_spf_from_tl
4543   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
4544   \tl_clear:N \l__stex_sproof_spf_type_tl
4545   \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

```

4546 \tl_clear:N \l__stex_sproof_spf_continues_tl
4547 \tl_clear:N \l__stex_sproof_spf_functions_tl
4548 \tl_clear:N \l__stex_sproof_spf_method_tl
4549 \keys_set:nn { stex / spf }{ #1 }
4550 }

```

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

```

4551 \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.

```

4552 \newcount\count_ten
4553 \newenvironment{pst@with@label}[1]{
4554   \edef\pst@label{#1}
4555   \advance\count_ten by 1\relax
4556   \count_ten=1
4557 }{
4558   \advance\count_ten by -1\relax
4559 }

```

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

```

4560 \def\the@pst@label{
4561   \pst@make@label\pst@label{\number\count_ten}\l__stex_sproof_pstlabel_postfix_tl
4562 }

```

(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.

```

4563 \keys_define:nn { stex / pstlabel }{
4564   prefix      .tl_set:N   = \l__stex_sproof_pstlabel_prefix_tl,
4565   delimiter   .tl_set:N   = \l__stex_sproof_pstlabel_delimiter_tl,
4566   postfix     .tl_set:N   = \l__stex_sproof_pstlabel_postfix_tl
4567 }
4568 \cs_new_protected:Nn \__stex_sproof_pstlabel_args:n {

```

---

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

```

4569 \tl_set:Nn \l__stex_sproof_pstlabel_prefix_tl {P}
4570 \tl_set:Nn \l__stex_sproof_pstlabel_delimiter_tl {.}
4571 \tl_clear:N \l__stex_sproof_pstlabel_postfix_tl
4572 }
4573 \__stex_sproof_pstlabel_args:n {}
4574 \newcommand\setpstlabelstyle[1]{
4575   \__stex_sproof_pstlabel_args:n {#1}
4576 }
4577 \newcommand\setpstlabelstyledefault{%
4578   \__stex_sproof_pstlabel_args:n{prefix=P,delimiter=.,postfix={}}
4579 }

```

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

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

```

4580 \ExplSyntaxOff
4581 \def\pst@make@label@long#1#2{\@for\@I:=#1\do{\expandafter\expandafter\expandafter\@I\csname
4582 \def\pst@make@label@angles#1#2{\ensuremath{\@for\@I:=#1\do{\rangle}}#2}
4583 \def\pst@make@label@short#1#2{#2}
4584 \def\pst@make@label@empty#1#2{}
4585 \ExplSyntaxOn
4586 \def\pstlabelstyle#1{%
4587   \def\pst@make@label{\use:c{pst@make@label@#1}}%
4588 }%
4589 \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.

```

4590 \def\next@pst@label{%
4591   \global\advance\count\count10 by 1%
4592 }%

```

(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

```

4593 \def\sproof@box{
4594   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
4595 }
4596 \def\spf@proofend{\sproof@box}
4597 \def\sproofend{
4598   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
4599     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
4600   }
4601 }
4602 \def\sProofEndSymbol#1{\def\sproof@box{#1}}

```

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

**spf@\*@kw**

```

4603 \def\spf@proofsketch@kw{Proof Sketch}
4604 \def\spf@proof@kw{Proof}
4605 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

4606 \AddToHook{begindocument}{
4607   \ltx@ifpackageloaded{babel}{
4608     \makeatletter
4609     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4610     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4611       \input{sproof-ngerman.ldf}
4612     }
4613     \clist_if_in:NnT \l_tmpa_clist {finnish}{
4614       \input{sproof-finnish.ldf}
4615     }
4616     \clist_if_in:NnT \l_tmpa_clist {french}{
4617       \input{sproof-french.ldf}
4618     }
4619     \clist_if_in:NnT \l_tmpa_clist {russian}{
4620       \input{sproof-russian.ldf}
4621     }
4622     \makeatother
4623   }{}
4624 }

```

`spfsketch`

```

4625 \newcommand\spfsketch[2][]{
4626   \__stex_sproof_spf_args:n{#1}
4627   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4628     \titleemph{
4629       \tl_if_empty:NnT \l__stex_sproof_spf_type_tl {
4630         \spf@proofsketch@kw
4631       }{
4632         \l__stex_sproof_spf_type_tl
4633       }
4634     }:
4635   }
4636   {~#2}
4637   %\sref@label@id{this \ifx\spf@type\@empty\spf@proofsketch@kw\else\spf@type\fi}
4638   \sproofend
4639 }

```

(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>

```

4640 \newenvironment{spfeq}[2][]{
4641   \__stex_sproof_spf_args:n{#1}
4642   %\sref@target
4643   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4644     \titleemph{
4645       \tl_if_empty:NnT \l__stex_sproof_spf_type_tl {
4646         \spf@proof@kw
4647       }{

```

<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



```

4648         \l__stex_sproof_spf_type_tl
4649     }
4650     }:
4651 }
4652 {-#2}
4653 \begin{displaymath}\begin{array}{rcll}
4654 }{
4655 \end{array}\end{displaymath}
4656 }

```

(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.

```

4657 \newenvironment{spf@proof}[2] []{
4658   \l__stex_sproof_spf_args:n{#1}
4659   %\sref@target
4660   \count_ten=10
4661   \par\noindent
4662   \tl_if_eq:NNTF \l__stex_sproof_spf_display_tl\spf@flow{
4663     \titleemph{
4664       \tl_if_empty:NNTF \l__stex_sproof_spf_type_tl {
4665         \spf@proof@kw
4666       }{
4667         \l__stex_sproof_spf_type_tl
4668       }
4669     }:
4670   }
4671   {-#2}
4672   %\sref@label@id{this \ifx\spf@type\empty\spf@proof@kw\else\spf@type\fi}
4673   \def\pst@label{}
4674   \newcount\pst@count% initialize the labeling mechanism
4675   \begin{description}\begin{pst@with@label}{\l__stex_sproof_pstlabel_prefix_tl}
4676   }{
4677     \end{pst@with@label}\end{description}
4678   }
4679   \newenvironment{sproof}[2] []{\begin{spf@proof}[#1]{#2}}{\sproofend\end{spf@proof}}
4680   \newenvironment{sProof}[2] []{\begin{spf@proof}[#1]{#2}}{\end{spf@proof}}

```

**\spfidea**

```

4681 \newcommand\spfidea[2] []{
4682   \l__stex_sproof_spf_args:n{#1}
4683   \titleemph{
4684     \tl_if_empty:NNTF \l__stex_sproof_spf_type_tl {Proof~Idea}{
4685       \l__stex_sproof_spf_type_tl
4686     }:
4687   }-#2
4688   \sproofend
4689 }

```

(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.

EdN:16

```

spfstep 16
4690 \newenvironment{spfstep}[1][]{
4691   \_stex_sproof_spf_args:n{#1}
4692   \@in@omtexttrue
4693   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4694     \item[\the@pst@label]
4695   }
4696   \tl_if_empty:NF \l__stex_sproof_spf_title_tl {
4697     {(\titleemph{\l__stex_sproof_spf_title_tl})\enspace}
4698   }
4699   %\sref@label{id{\pst@label}
4700   \ignorespacesandpars
4701 }{
4702   \next@pst@label\ignorespacesandpars
4703 }

```

**sproofcomment**

```

4704 \newenvironment{sproofcomment}[1][]{
4705   \_stex_sproof_spf_args:n{#1}
4706   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4707     \item[\the@pst@label]
4708   }
4709 }{
4710   \next@pst@label
4711 }

```

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.

```

4712 \newenvironment{subproof}[2][]{
4713   \_stex_sproof_spf_args:n{#1}
4714   \def\@test{#2}
4715   \ifx\@test\empty\else
4716     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4717       \item[\the@pst@label]
4718     }{#2}
4719   \fi
4720   \begin{pst@with@label}{\pst@label,\number\count_ten}
4721 }{
4722   \end{pst@with@label}\next@pst@label
4723 }

```

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

```

4724 \newenvironment{spfcases}[2][]{
4725   \def\@test{#1}
4726   \ifx\@test\empty
4727     \begin{subproof}[method=by-cases]{#2}

```

---

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

```

4728 \else
4729   \begin{subproof}[#1,method=by-cases]{#2}
4730 \fi
4731 }{
4732   \end{subproof}
4733 }

```

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

```

4734 \newenvironment{spfcase}[2] [] {
4735   \__stex_sproof_spf_args:n{#1}
4736   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4737     \item[\the@pst@label]
4738   }
4739   \def\@test{#2}
4740   \ifx\@test\@empty
4741     \else
4742       {\titleemph{#2}:~}
4743     \fi
4744     \begin{pst@with@label}{\pst@label,\number\count_ten}
4745   }{
4746     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4747       \sproofend
4748     }
4749     \end{pst@with@label}
4750     \next@pst@label
4751   }

```

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

```

4752 \newcommand\spfcasesketch[3] [] {
4753   \__stex_sproof_spf_args:n{#1}
4754   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4755     \item[\the@pst@label]
4756   }
4757   \def\@test{#2}
4758   \ifx\@test\@empty
4759     \else
4760       {\titleemph{#2}:~}
4761     \fi#3
4762     \next@pst@label
4763   }%

```

### 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.

```

4764 \keys_define:nn { stex / just }{
4765   id      .str_set:x:N = \l__stex_sproof_just_id_str,
4766   method  .tl_set:N    = \l__stex_sproof_just_method_tl,
4767   premises .tl_set:N    = \l__stex_sproof_just_premises_tl,
4768   args    .tl_set:N    = \l__stex_sproof_just_args_tl
4769 }

```

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>

`justification`

4770 `\newenvironment{justification}[1] [] {}{}`

`\premise`

4771 `\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.

4772 `\newcommand\justarg[2] [] {#2}`

4773 `\</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.

---

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

## Chapter 35

# STEX -Others Implementation

```
4774 <*package>
4775
4776 %%%%%%%%%% others.dtx %%%%%%%%%%
4777
4778 <@@=stex_others>
      Warnings and error messages
4779 % None

\MSC Math subject classifier

4780 \NewDocumentCommand \MSC {m} {
4781   % TODO
4782 }

(End definition for \MSC. This function is documented on page 21.)
      Patching tikzinput, if loaded
4783 \@ifpackageloaded{tikzinput}{
4784   \RequirePackage{stex-tikzinput}
4785 }{}
4786 </package>
```

## Chapter 36

# STEX -Metatheory Implementation

```
4787 <*package>
4788 <@@=stex_modules>
4789
4790 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
4791
4792 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
4793 \begingroup
4794 \stex_module_setup:nn{
4795   ns=\c_stex_metatheory_ns_str,
4796   meta=NONE
4797 }{Metatheory}
4798 \stex_reactivate_macro:N \symdecl
4799 \stex_reactivate_macro:N \notation
4800 \stex_reactivate_macro:N \symdef
4801 \ExplSyntaxOff
4802 \csname stex_suppress_html:n\endcsname{
4803   % is-a (a:A, a \in A, a is an A, etc.)
4804   \symdecl[args=ai]{isa}
4805   \notation[typed]{isa}{#1 \comp{:} #2}{#1 \comp, #2}
4806   \notation[in]{isa}{#1 \comp\in #2}{#1 \comp, #2}
4807   \notation[pred]{isa}{#2\comp(#1 \comp)}{#1 \comp, #2}
4808
4809   % bind (\forall, \Pi, \lambda etc.)
4810   \symdecl[args=Bi]{bind}
4811   \notation[forall]{bind}{\comp\forall #1.\;#2}{#1 \comp, #2}
4812   \notation[\Pi]{bind}{\comp\prod_{#1}#2}{#1 \comp, #2}
4813   \notation[depfun]{bind}{\comp( #1 \comp)\; \to\;}{#1 \comp, #2}
4814
4815   % dummy variable
4816   \symdecl{dummyvar}
4817   \notation[underscore]{dummyvar}{\comp\_}
4818   \notation[dot]{dummyvar}{\comp\cdot}
4819   \notation[dash]{dummyvar}{\comp{\rm --}}
4820
4821   %fromto (function space, Hom-set, implication etc.)
```

```

4822 \symdecl[args=ai]{fromto}
4823 \notation[xarrow]{fromto}{#1 \comp\to #2}{#1 \comp\times #2}
4824 \notation[arrow]{fromto}{#1 \comp\to #2}{#1 \comp\to #2}
4825
4826 % mapto (lambda etc.)
4827 %\symdecl[args=Bi]{mapto}
4828 %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
4829 %\notation[lambda]{mapto}{\comp\lambda #1 \comp. \; #2}{#1 \comp, #2}
4830 %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp. \; #2}{#1 \comp, #2}
4831
4832 % function/operator application
4833 \symdecl[args=ia]{apply}
4834 \notation[prec=0;0x\infpres,parens]{apply}{#1 \comp( #2 \comp)}{#1 \comp, #2}
4835 \notation[prec=0;0x\infpres,lambda]{apply}{#1 \; #2 }{#1 \; #2}
4836
4837 % ‘type’ of all collections (sets, classes, types, kinds)
4838 \symdecl{collection}
4839 \notation[U]{collection}{\comp{\mathcal{U}}}
4840 \notation[set]{collection}{\comp{\textsf{Set}}}
4841
4842 % sequences
4843 \symdecl[args=1]{seqtype}
4844 \notation[kleene]{seqtype}{#1^{\comp\ast}}
4845
4846 \symdef[args=2,li,prec=nobrackets]{sequence-index}{#1}_{#2}
4847 \notation[ui,prec=nobrackets]{sequence-index}{#1}^{#2}
4848
4849 %\symdef[args=3,li]{sequence-from-to}{#1}_{#2}\comp{\,\ellipses,}#1_{#3}
4850 %\notation[ui]{sequence-from-to}{#1}^{#2}\comp{\,\ellipses,}#1^{#3}
4851 % ^ superceded by \aseqfromto and \livar/\uivar
4852
4853 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{\,\ellipses}}{#1\comp,#2}
4854 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{\,\ellipses,}#2}{#1\comp,#2}
4855 \symdef[args=aui,prec=nobrackets]{aseqfromtovia}{#1\comp{\,\ellipses,}#2\comp{\,\ellipses,}#3}
4856
4857 % letin (‘let’, local definitions, variable substitution)
4858 \symdecl[args=bii]{letin}
4859 \notation[let]{letin}{\comp{\rm let}}\;#1\comp{=}\;#2\; \comp{\rm in}\;#3}
4860 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
4861 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
4862
4863 % structures
4864 \symdecl*[args=1]{module-type}
4865 \notation{module-type}{\mathtt{MOD} #1}
4866 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
4867 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{#1 \comp, #2}
4868
4869 }
4870 \ExplSyntaxOn
4871 \stex_add_to_current_module:n{
4872   \let\nappa\apply
4873   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
4874   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
4875   \def\livar{\csname sequence-index\endcsname[li]}

```

```

4876 \def\uivar{\csname sequence-index\endcsname[ui]}
4877 \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
4878 \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
4879 \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
4880 }
4881 \__stex_modules_end_module:
4882 \endgroup
4883 \endpackage

```



## Chapter 37

# Tikzinput Implementation

```
4884 <*package>
4885
4886 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
4887
4888 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
4889 \RequirePackage{l3keys2e}
4890
4891 \keys_define:nn { tikzinput } {
4892   image .bool_set:N = \c_tikzinput_image_bool,
4893   image .default:n = false ,
4894   unknown .code:n = {}
4895 }
4896
4897 \ProcessKeysOptions { tikzinput }
4898
4899 \bool_if:NTF \c_tikzinput_image_bool {
4900   \RequirePackage{graphicx}
4901
4902   \providecommand\usetikzlibrary[]{}
4903   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
4904 }{
4905   \RequirePackage{tikz}
4906   \RequirePackage{standalone}
4907
4908   \newcommand \tikzinput [2] [] {
4909     \setkeys{Gin}{#1}
4910     \ifx \Gin@ewidth \Gin@exclamation
4911       \ifx \Gin@eheight \Gin@exclamation
4912         \input { #2 }
4913       \else
4914         \resizebox{!}{ \Gin@eheight }{
4915           \input { #2 }
4916         }
4917       \fi
4918     \else
4919       \ifx \Gin@eheight \Gin@exclamation
4920         \resizebox{ \Gin@ewidth }{!}{
4921           \input { #2 }
```

```

4922     }
4923     \else
4924         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
4925             \input { #2 }
4926         }
4927     \fi
4928 \fi
4929 }
4930 }
4931
4932 \newcommand \ctikzinput [2] [] {
4933     \begin{center}
4934         \tikzinput [1] {#2}
4935     \end{center}
4936 }
4937
4938 \@ifpackageloaded{stex}{
4939     \RequirePackage{stex-tikzinput}
4940 }{}
4941
4942 </package>
4943 <*stex>
4944 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
4945 \RequirePackage{stex}
4946 \RequirePackage{tikzinput}
4947
4948 \newcommand\mhtikzinput [2] [] {%
4949     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
4950     \stex_in_repository:nn\Gin@mhrepos{
4951         \tikzinput [1]{\mhpath{##1}{#2}}
4952     }
4953 }
4954 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [1] {#2}\end{center}}
4955 </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 document-structure Class

The functionality is spread over the `document-structure` class and package. The class provides the `document` environment and the `document-structure` element corresponds to it, whereas the package provides the concrete functionality.

```
4956 \*cls)
4957 \@@=document_structure)
4958 \ProvidesExplClass{document-structure}{2022/02/10}{3.0}{Modular Document Structure Class}
4959 \RequirePackage{l3keys2e,expl-keystr-compatible}
```

### 38.2 Class Options

To initialize the `document-structure` 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`

```
4960 \keys_define:nn{ document-structure / pkg }{
4961   class      .str_set_x:N = \c_document_structure_class_str,
4962   minimal    .bool_set:N = \c_document_structure_minimal_bool,
4963   report     .code:n      = {
4964     \ClassWarning{document-structure}{the option 'report' is deprecated, use 'class=report',
4965     \str_set:Nn \c_document_structure_class_str {report}
4966   },
4967   book       .code:n      = {
4968     \ClassWarning{document-structure}{the option 'book' is deprecated, use 'class=book', ins
4969     \str_set:Nn \c_document_structure_class_str {book}
4970   },
4971   bookpart   .code:n      = {
4972     \ClassWarning{document-structure}{the option 'bookpart' is deprecated, use 'class=book,t
4973     \str_set:Nn \c_document_structure_class_str {book}
4974     \str_set:Nn \c_document_structure_topsect_str {chapter}
4975   },
```

```

4976 docopt      .str_set_x:N = \c_document_structure_docopt_str,
4977 unknown     .code:n      = {
4978   \PassOptionsToPackage{ \CurrentOption }{ document-structure }
4979 }
4980 }
4981 \ProcessKeysOptions{ document-structure / pkg }
4982 \str_if_empty:NT \c_document_structure_class_str {
4983   \str_set:Nn \c_document_structure_class_str {article}
4984 }
4985 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
4986   {\c_document_structure_class_str}
4987

```

### 38.3 Beefing up the document environment

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

```

4988 \RequirePackage{document-structure}
4989 \bool_if:NF \c_document_structure_minimal_bool {

```

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>

```

4990 \keys_define:nn { document-structure / document }{
4991   id .str_set_x:N = \c_document_structure_document_id_str
4992 }
4993 \let\__document_structure_orig_document=\document
4994 \renewcommand{\document}[1][]{
4995   \keys_set:nn{ document-structure / document }{ #1 }
4996   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
4997   \__document_structure_orig_document
4998 }

```

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

```

4999 }
5000 \</cls>

```

### 38.4 Implementation: document-structure Package

```

5001 \<*package>
5002 \ProvidesExplPackage{document-structure}{2022/02/10}{3.0}{Modular Document Structure}
5003 \RequirePackage{expl-keystr-compat,13keys2e}

```

### 38.5 Package Options

We declare some switches which will modify the behavior according to the package options. Generally, an option `xxx` will just set the appropriate switches to true (otherwise they stay false).

---

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

```

5004
5005 \keys_define:nn{ document-structure / pkg }{
5006   class      .str_set_x:N = \c_document_structure_class_str,
5007   topsect    .str_set_x:N = \c_document_structure_topsect_str,
5008   % showignores .bool_set:N = \c_document_structure_showignores_bool,
5009 }
5010 \ProcessKeysOptions{ document-structure / pkg }
5011 \str_if_empty:NT \c_document_structure_class_str {
5012   \str_set:Nn \c_document_structure_class_str {article}
5013 }
5014 \str_if_empty:NT \c_document_structure_topsect_str {
5015   \str_set:Nn \c_document_structure_topsect_str {section}
5016 }

```

Then we need to set up the packages by requiring the `sref` package to be loaded, and set up triggers for other languages

```

5017 \RequirePackage{xspace}
5018 \RequirePackage{comment}
5019 \AddToHook{begindocument}{
5020   \ltx@ifpackageloaded{babel}{
5021     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5022     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5023       \makeatletter\input{omdoc-ngerman.ldf}\makeatother
5024     }
5025   }{}
5026 }

```

`\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.

```

5027 \int_new:N \l_document_structure_section_level_int
5028 \str_case:VnF \c_document_structure_topsect_str {
5029   {part}}{
5030     \int_set:Nn \l_document_structure_section_level_int {0}
5031   }
5032   {chapter}}{
5033     \int_set:Nn \l_document_structure_section_level_int {1}
5034   }
5035 }{
5036   \str_case:VnF \c_document_structure_class_str {
5037     {book}}{
5038       \int_set:Nn \l_document_structure_section_level_int {0}
5039     }
5040     {report}}{
5041       \int_set:Nn \l_document_structure_section_level_int {0}
5042     }
5043   }{
5044     \int_set:Nn \l_document_structure_section_level_int {2}
5045   }
5046 }

```

## 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

```
5047 \def\current@section@level{document}%
5048 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
5049 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
5050 \cs_new_protected:Npn \skipomgroup {
5051   \ifcase\l_document_structure_section_level_int
5052   \or\stepcounter{part}
5053   \or\stepcounter{chapter}
5054   \or\stepcounter{section}
5055   \or\stepcounter{subsection}
5056   \or\stepcounter{subsubsection}
5057   \or\stepcounter{paragraph}
5058   \or\stepcounter{subparagraph}
5059   \fi
5060 }
```

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

`blindomgroup`

```
5061 \newcommand\at@begin@blindomgroup[1]{%
5062 \newenvironment{blindomgroup}
5063 {
5064   \int_incr:N\l_document_structure_section_level_int
5065   \at@begin@blindomgroup\l_document_structure_section_level_int
5066 }{}}
```

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

```
5067 \newcommand\omgroup@nonum[2]{
5068   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
5069   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
5070 }
```

*(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.

```
5071 \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.

```

5072 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
5073   \@nameuse{#1}{#2}
5074 }{
5075   \cs_if_exist:NTF\rdfmata@sectioning{
5076     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
5077   }{
5078     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
5079   }
5080 }
5081 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
5082 }

```

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

omgroup

```

5083 \keys_define:nn { document-structure / omgroup }{
5084   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
5085   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
5086   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
5087   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
5088   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
5089   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
5090   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
5091   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
5092   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
5093   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
5094 }
5095 \cs_new_protected:Nn \__document_structure_omgroup_args:n {
5096   \str_clear:N \l__document_structure_omgroup_id_str
5097   \str_clear:N \l__document_structure_omgroup_date_str
5098   \clist_clear:N \l__document_structure_omgroup_creators_clist
5099   \clist_clear:N \l__document_structure_omgroup_contributors_clist
5100   \tl_clear:N \l__document_structure_omgroup_srccite_tl
5101   \tl_clear:N \l__document_structure_omgroup_type_tl
5102   \tl_clear:N \l__document_structure_omgroup_short_tl
5103   \tl_clear:N \l__document_structure_omgroup_display_tl
5104   \tl_clear:N \l__document_structure_omgroup_intro_tl
5105   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
5106   \keys_set:nn { document-structure / omgroup } { #1 }
5107 }

```

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.

```

5108 \newif\if@mainmatter\@mainmattertrue
5109 \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.

```

5110 \keys_define:nn { document-structure / sectioning }{
5111   name .str_set_x:N = \l__document_structure_sect_name_str ,
5112   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
5113   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
5114   num .bool_set:N = \l__document_structure_sect_num_bool ,
5115 }

```

```

5116 \cs_new_protected:Nn \l__document_structure_sect_args:n {
5117   \str_clear:N \l__document_structure_sect_name_str
5118   \str_clear:N \l__document_structure_sect_ref_str
5119   \bool_set_false:N \l__document_structure_sect_clear_bool
5120   \bool_set_false:N \l__document_structure_sect_num_bool
5121   \keys_set:nn { document-structure / sectioning } { #1 }
5122 }
5123 \newcommand\omdoc@sectioning[3][]{
5124   \l__document_structure_sect_args:n {#1}
5125   \let\omdoc@sect@name\l__document_structure_sect_name_str
5126   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
5127   \if@mainmatter% numbering not overridden by frontmatter, etc.
5128     \bool_if:NTF \l__document_structure_sect_num_bool {
5129       \omgroup@num{#2}{#3}
5130     }{
5131       \omgroup@nonum{#2}{#3}
5132     }
5133     \def\current@section@level{\omdoc@sect@name}
5134   \else
5135     \omgroup@nonum{#2}{#3}
5136   \fi
5137 }% 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.

```

5138 \newcommand\omgroup@redefine@addtocontents[1]{%
5139 %\edef\l__document_structureimport{#1}%
5140 %\@for\@I:=\l__document_structureimport\do{%
5141 %\edef\@path{\csname module@\@I @path\endcsname}%
5142 %\@ifundefined{tf@toc}\relax%
5143 %   {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
5144 %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
5145 %\def\addcontentsline##1##2##3{%
5146 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
5147 %\else% hyperref.sty not loaded
5148 %\def\addcontentsline##1##2##3{%
5149 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
5150 %\fi
5151 }% 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.

```

5152 \int_new:N \l__document_structure_omgroup_level_int
5153 \newenvironment{omgroup}[2][]{% keys, title
5154 {
5155   \l__document_structure_omgroup_args:n { #1 }% \sref@target%

```

If the `loadmodules` key is set on `\begin{omgroup}`, we redefine the `\addcontetsline` macro that determines how the sectioning commands below construct the entries for the table of contents.

```

5156 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
5157   \omgroup@redefine@addtocontents{
5158     %\@ifundefined{module@id}\used@modules%
5159     %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}

```



```

5160     }
5161 }

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

5162 \int_incr:N \l_document_structure_omgroup_level_int
5163 \int_incr:N \l_document_structure_section_level_int
5164 \ifcase\l_document_structure_section_level_int
5165   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
5166   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
5167   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
5168   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
5169   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
5170   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
5171   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
5172 \fi
5173 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
5174 \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
5175 }% for customization
5176 {}

```

and finally, we localize the sections

```

5177 \newcommand\omdoc@part@kw{Part}
5178 \newcommand\omdoc@chapter@kw{Chapter}
5179 \newcommand\omdoc@section@kw{Section}
5180 \newcommand\omdoc@subsection@kw{Subsection}
5181 \newcommand\omdoc@subsubsection@kw{Subsubsection}
5182 \newcommand\omdoc@paragraph@kw{paragraph}
5183 \newcommand\omdoc@subparagraph@kw{subparagraph}

```

## 38.7 Front and Backmatter

Index markup is provided by the `omtext` package [Koh20c], so in the `document-structure` package we only need to supply the corresponding `\printindex` command, if it is not already defined

`\printindex`

```

5184 \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).

```

5185 \cs_if_exist:NTF\frontmatter{
5186   \let\__document_structure_orig_frontmatter\frontmatter
5187   \let\frontmatter\relax
5188 }{
5189   \tl_set:Nn\__document_structure_orig_frontmatter{
5190     \clearpage
5191     \@mainmatterfalse
5192     \pagenumbering{roman}
5193   }
5194 }

```

```

5195 \cs_if_exist:NTF\backmatter{
5196   \let\__document_structure_orig_backmatter\backmatter
5197   \let\backmatter\relax
5198 }{
5199   \tl_set:Nn\__document_structure_orig_backmatter{
5200     \clearpage
5201     \@mainmatterfalse
5202     \pagenumbering{roman}
5203   }
5204 }

```

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.

```
5205 \newenvironment{frontmatter}{
5206   \__document_structure_orig_frontmatter
5207 }{
5208   \cs_if_exist:NTF\mainmatter{
5209     \mainmatter
5210   }{
5211     \clearpage
5212     \@mainmattertrue
5213     \pagenumbering{arabic}
5214   }
5215 }
```

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

```

5216 \newenvironment{backmatter}{
5217   \__document_structure_orig_backmatter
5218 }{
5219   \cs_if_exist:NTF\mainmatter{
5220     \mainmatter
5221   }{
5222     \clearpage
5223     \@mainmattertrue
5224     \pagenumbering{arabic}
5225   }
5226 }

```

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

5227 \@mainmattertrue\pagenumbering{arabic}

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

```

5228 \def \c__document_structure_document_str{document}
5229 \newcommand\afterprematurestop{}
5230 \def\prematurestop@endomgroup{
5231   \unless\ifx\@currentvir\c__document_structure_document_str
5232     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
5233     \expandafter\prematurestop@endomgroup
5234   \fi
5235 }

```

```

5236 \providecommand\prematurestop{
5237   \message{Stopping~sTeX~processing~prematurely}
5238   \prematurestop@endomgroup
5239   \afterprematurestop
5240   \end{document}
5241 }

```

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

## 38.8 Global Variables

**\setSGvar** set a global variable

```

5242 \RequirePackage{etoolbox}
5243 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

**\useSGvar** use a global variable

```

5244 \newrobustcmd\useSGvar[1]{%
5245   \@ifundefined{sTeX@Gvar@#1}
5246   {\PackageError{document-structure}
5247     {The sTeX Global variable #1 is undefined}
5248     {set it with \protect\setSGvar}}
5249   \@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.

```

5250 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
5251   \@ifundefined{sTeX@Gvar@#1}
5252   {\PackageError{document-structure}
5253     {The sTeX Global variable #1 is undefined}
5254     {set it with \protect\setSGvar}}
5255   {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}

```

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

## Chapter 39

# NotesSlides – Implementation

### 39.1 Class and Package Options

We define some Package Options and switches for the `notesslides` class and activate them by passing them on to `beamer.cls` and `omdoc.cls` and the `notesslides` 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.

```
5256 \*cls)
5257 \@@=notesslides)
5258 \ProvidesExplClass{notesslides}{2022/02/10}{3.0}{notesslides Class}
5259 \RequirePackage{l3keys2e,expl-keystr-compatible}
5260
5261 \keys_define:nn{notesslides / cls}{
5262   class .code:n = {
5263     \PassOptionsToClass{\CurrentOption}{omdoc}
5264     \str_if_eq:nnT{#1}{book}{
5265       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5266     }
5267     \str_if_eq:nnT{#1}{report}{
5268       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5269     }
5270   },
5271   notes .bool_set:N = \c__notesslides_notes_bool ,
5272   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
5273   unknown .code:n = {
5274     \PassOptionsToClass{\CurrentOption}{omdoc}
5275     \PassOptionsToClass{\CurrentOption}{beamer}
5276     \PassOptionsToPackage{\CurrentOption}{notesslides}
5277   }
5278 }
5279 \ProcessKeysOptions{ notesslides / cls }
5280 \bool_if:NTF \c__notesslides_notes_bool {
5281   \PassOptionsToPackage{notes=true}{notesslides}
5282 }{
5283   \PassOptionsToPackage{notes=false}{notesslides}
5284 }
5285 \</cls)
```

now we do the same for the notesslides package.

```

5286 <*package>
5287 \ProvidesExplPackage{notesslides}{2022/02/10}{3.0}{notesslides Package}
5288 \RequirePackage{13keys2e,expl-keystr-compat}
5289
5290 \keys_define:nn{notesslides / pkg}{
5291   topsect          .str_set_x:N = \c__notesslides_topsect_str,
5292   defaulttopsect   .str_set_x:N = \c__notesslides_defaulttopsec_str,
5293   notes            .bool_set:N = \c__notesslides_notes_bool ,
5294   slides           .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
5295   sectocframes     .bool_set:N = \c__notesslides_sectocframes_bool ,
5296   frameimages      .bool_set:N = \c__notesslides_frameimages_bool ,
5297   fiboxed          .bool_set:N = \c__notesslides_fiboxed_bool ,
5298   noproblems       .bool_set:N = \c__notesslides_noproblems_bool,
5299   unknown          .code:n      = {
5300     \PassOptionsToClass{\CurrentOption}{stex}
5301     \PassOptionsToClass{\CurrentOption}{tikzinput}
5302   }
5303 }
5304 \ProcessKeysOptions{ notesslides / pkg }
5305 \newif\ifnotes
5306 \bool_if:NTF \c__notesslides_notes_bool {
5307   \notesttrue
5308 }{
5309   \notesfalse
5310 }
5311

```

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

```

5312 \str_if_empty:NTF \c__notesslides_topsect_str {
5313   \str_set_eq:NN \__notesslidestopsect \c__notesslides_defaulttopsec_str
5314 }{
5315   \str_set_eq:NN \__notesslidestopsect \c__notesslides_topsect_str
5316 }
5317 </package>

```

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

```

5318 <*cls>
5319 \bool_if:NTF \c__notesslides_notes_bool {
5320   \LoadClass{document-structure}
5321 }{
5322   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
5323   \newcounter{Item}
5324   \newcounter{paragraph}
5325   \newcounter{subparagraph}
5326   \newcounter{Hfootnote}
5327   \RequirePackage{document-structure}
5328 }

```

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

```

5329 \RequirePackage{notesslides}
5330 </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 `notesslides.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.

```

5331 \*package>
5332 \bool_if:NT \c__notesslides_notes_bool {
5333   \RequirePackage{a4wide}
5334   \RequirePackage{marginnote}
5335   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
5336   \RequirePackage{mdframed}
5337   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
5338   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
5339 }
5340 \RequirePackage{stex-tikzinput}
5341 \RequirePackage{etoolbox}
5342 \RequirePackage{amssymb}
5343 \RequirePackage{amsmath}
5344 \RequirePackage{comment}
5345 \RequirePackage{textcomp}
5346 \RequirePackage{url}
5347 \RequirePackage{graphicx}
5348 \RequirePackage{pgf}

```

## 39.2 Notes and Slides

For the lecture notes cases, we also provide the `\usetheme` macro that would otherwise come from the the `beamer` class. While the latter loads `beamertheme<theme>.sty`, the notes version loads `beamernotestheme<theme>.sty`.<sup>20</sup>

```

5349 \bool_if:NT \c__notesslides_notes_bool {
5350   \renewcommand\usetheme[2][\]{\usepackage[#1]{beamernotestheme#2}}
5351 }

```

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

```

5352 \newcounter{slide}
5353 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
5354 \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.

```

5355 \bool_if:NTF \c__notesslides_notes_bool {
5356   \renewenvironment{note}{\ignorespaces}{\ignorespaces}{}
5357 }{
5358   \excludecomment{note}
5359 }

```

---

<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.

```
5360 \bool_if:NT \c__notesslides_notes_bool {
5361   \newlength{\slideframewidth}
5362   \setlength{\slideframewidth}{1.5pt}
```

**frame** We first define the keys.

```
5363 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
5364   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
5365     \bool_set_true:N #1
5366   }{
5367     \bool_set_false:N #1
5368   }
5369 }
5370 \keys_define:nn{notesslides / frame}{
5371   label .str_set_x:N = \l__notesslides_frame_label_str,
5372   allowframebreaks .code:n = {
5373     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
5374   },
5375   allowdisplaybreaks .code:n = {
5376     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
5377   },
5378   fragile .code:n = {
5379     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
5380   },
5381   shrink .code:n = {
5382     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
5383   },
5384   squeeze .code:n = {
5385     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
5386   },
5387   t .code:n = {
5388     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
5389   },
5390 }
5391 \cs_new_protected:Nn \__notesslides_frame_args:n {
5392   \str_clear:N \l__notesslides_frame_label_str
5393   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
5394   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
5395   \bool_set_true:N \l__notesslides_frame_fragile_bool
5396   \bool_set_true:N \l__notesslides_frame_shrink_bool
5397   \bool_set_true:N \l__notesslides_frame_squeeze_bool
5398   \bool_set_true:N \l__notesslides_frame_t_bool
5399   \keys_set:nn { notesslides / frame }{ #1 }
5400 }
```

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

```
5401 \renewenvironment{frame}[1][]{
5402   \__notesslides_frame_args:n{#1}
5403   \sffamily
5404   \stepcounter{slide}
5405   \def\@currentlabel{\theslide}
5406   \str_if_empty:NF \l__notesslides_frame_label_str {
5407     \label{\l__notesslides_frame_label_str}
```

5408 }

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

5409 \def\itemize@level{outer}  
5410 \def\itemize@outer{outer}  
5411 \def\itemize@inner{inner}  
5412 \renewcommand\newpage{\addtocounter{framenum}{1}}  
5413 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}  
5414 \renewenvironment{itemize}{  
5415 \ifx\itemize@level\itemize@outer  
5416 \def\itemize@label{\\$ \rhd\$}  
5417 \fi  
5418 \ifx\itemize@level\itemize@inner  
5419 \def\itemize@label{\\$ \scriptstyle \rhd\$}  
5420 \fi  
5421 \begin{list}  
5422 {\itemize@label}  
5423 {\setlength{\labelsep}{.3em}  
5424 \setlength{\labelwidth}{.5em}  
5425 \setlength{\leftmargin}{1.5em}  
5426 }  
5427 \edef\itemize@level{\itemize@inner}  
5428 }{  
5429 \end{list}  
5430 }

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

5431 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth=100pt]  
5432 }{  
5433 \medskip\miko@slidelabel\end{mdframed}  
5434 }

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

\frametitle

5435 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip  
5436 }

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

EdN:21

\pause 21

5437 \bool\_if:NT \c\_\_notesslides\_notes\_bool {  
5438 \newcommand\pause{  
5439 }

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

nparagraph

5440 \bool\_if:NTF \c\_\_notesslides\_notes\_bool {  
5441 \newenvironment{nparagraph}[1] []{\begin{sparagraph}[#1]}{\end{sparagraph}}  
5442 }{  
5443 \excludecomment{nparagraph}  
5444 }

---

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



```

nomgroup
5445 \bool_if:NTF \c__notesslides_notes_bool {
5446   \newenvironment{nomgroup}[2] [] {\begin{omgroup}[#1]{#2}}{\end{omgroup}}
5447 }{
5448   \excludecomment{nomgroup}
5449 }

ntheorem
5450 \bool_if:NTF \c__notesslides_notes_bool {
5451   \newenvironment{ntheorem}[1] [] {\begin{sdefinition}[#1]}{\end{sdefinition}}
5452 }{
5453   \excludecomment{ntheorem}
5454 }

nassertion
5455 \bool_if:NTF \c__notesslides_notes_bool {
5456   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}{\end{sassertion}}
5457 }{
5458   \excludecomment{nassertion}
5459 }

nsproof
5460 \bool_if:NTF \c__notesslides_notes_bool {
5461   \newenvironment{nsproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
5462 }{
5463   \excludecomment{nsproof}
5464 }

nexample
5465 \bool_if:NTF \c__notesslides_notes_bool {
5466   \newenvironment{nexample}[1] [] {\begin{example}[#1]}{\end{example}}
5467 }{
5468   \excludecomment{nexample}
5469 }

\inputref@*skip We customize the hooks for in \inputref.
5470 \def\inputref@preskip{\smallskip}
5471 \def\inputref@postskip{\medskip}

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

\inputref*
5472 \let\orig@inputref\inputref
5473 \def\inputref{\@ifstar\ninputref\orig@inputref}
5474 \newcommand\ninputref[2] [] {
5475   \bool_if:NT \c__notesslides_notes_bool {
5476     \orig@inputref[#1]{#2}
5477   }
5478 }

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

```

5479 \newlength{\slidelogoheight}
5480
5481 \bool_if:NTF \c__notesslides_notes_bool {
5482   \setlength{\slidelogoheight}{.4cm}
5483 }{
5484   \setlength{\slidelogoheight}{1cm}
5485 }
5486 \newsavebox{\slidelogo}
5487 \sbox{\slidelogo}{\TeX}
5488 \newrobustcmd{\setslidelogo}[1]{
5489   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
5490 }
```

(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.

```

5491 \def\source{Michael Kohlhase}% customize locally
5492 \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.

```

5493 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
5494 \newsavebox{\cclogo}
5495 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
5496 \newif\ifcchref\cchreffalse
5497 \AtBeginDocument{
5498   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
5499 }
5500 \def\licensing{
5501   \ifcchref
5502     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
5503   \else
5504     {\usebox{\cclogo}}
5505   \fi
5506 }
5507 \newrobustcmd{\setlicensing}[2][]{
5508   \def@url{#1}
5509   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
5510   \ifx@url@empty
5511     \def\licensing{{\usebox{\cclogo}}}
5512   \else
5513     \def\licensing{
```

```

5514     \ifcchref
5515     \href{#1}{\usebox{\cclogo}}
5516   \else
5517     {\usebox{\cclogo}}
5518   \fi
5519 }
5520 \fi
5521 }

```

(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>

```

5522 \newrobustcmd\miko@slidelabel{
5523   \vbox to \slidelogoheight{
5524     \vss\hbox to \slidewidth
5525     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
5526   }
5527 }

```

(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.

```

5528 \def\Gin@mhrepos{}
5529 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
5530 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
5531 \newrobustcmd\frameimage[2][{}]{
5532   \stepcounter{slide}
5533   \bool_if:NT \c__notesslides_frameimages_bool {
5534     \def\Gin@ewidth{}\setkeys{Gin}{#1}
5535     \bool_if:NF \c__notesslides_notes_bool { \vfill }
5536     \begin{center}
5537       \bool_if:NTF \c__notesslides_fiboxed_bool {
5538         \fbox{
5539           \ifx\Gin@ewidth\@empty
5540             \ifx\Gin@mhrepos\@empty
5541               \mhgraphics[width=\slidewidth,#1]{#2}
5542             \else
5543               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5544             \fi
5545           \else% Gin@ewidth empty
5546             \ifx\Gin@mhrepos\@empty
5547               \mhgraphics[#1]{#2}
5548             \else
5549               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5550             \fi
5551           \fi% Gin@ewidth empty
5552         }
5553       }{
5554         \ifx\Gin@ewidth\@empty

```

---

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

```

5555         \ifx\Gin@mhrepos\@empty
5556             \mhgraphics[width=\slidewidth,#1]{#2}
5557         \else
5558             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5559         \fi
5560         \ifx\Gin@mhrepos\@empty
5561             \mhgraphics[#1]{#2}
5562         \else
5563             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5564         \fi
5565     \fi% Gin@ewidth empty
5566 }
5567 \end{center}
5568 \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
5569 \bool_if:NF \c__notesslides_notes_bool { \vfill }
5570 }
5571 } % 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.

```

5572 \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

```

5573 \AddToHook{begindocument}{
5574     \definecolor{green}{rgb}{0,.5,0}
5575     \definecolor{purple}{cmyk}{.3,1,0,.17}
5576 }

```

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`.

```

5577 % \def\STpresent#1{\textcolor{blue}{#1}}
5578 \def\defemph#1{\textcolor{magenta}{#1}}
5579 \def\symrefemph#1{\textcolor{cyan}{#1}}
5580 \def\compemph#1{\textcolor{blue}{#1}}
5581 \def\titleemph#1{\textcolor{blue}{#1}}
5582 \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.

```

5583 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
5584 \def\smalltextwarning{
5585     \pgfuseimage{miko@small@dbend}
5586     \xspace
5587 }
5588 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}

```

```

5589 \newrobustcmd\textwarning{
5590   \raisebox{-0.05cm}{\pgfuseimage{miko@dbend}}
5591   \xspace
5592 }
5593 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}
5594 \newrobustcmd\bigtextwarning{
5595   \raisebox{-0.05cm}{\pgfuseimage{miko@big@dbend}}
5596   \xspace
5597 }

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

5598 \newrobustcmd\putgraphicsat[3]{
5599   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
5600 }
5601 \newrobustcmd\putat[2]{
5602   \begin{picture}(0,0)\put(#1){#2}\end{picture}
5603 }

```

## 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`.

```

5604 \bool_if:NT \c__notesslides_sectocframes_bool {
5605   \str_if_eq:VnTF \__notesslidesstopsect{part}{
5606     \newcounter{chapter}\counterwithin*{section}{chapter}
5607   }{
5608     \str_if_eq:VnT\__notesslidesstopsect{chapter}{
5609       \newcounter{chapter}\counterwithin*{section}{chapter}
5610     }
5611   }
5612 }

```

`\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
5613 \def\part@prefix{}
5614 \@ifpackageloaded{document-structure}{
5615   \str_case:VnF \__notesslidesstopsect {
5616     {part}{
5617       \int_set:Nn \l_document_structure_section_level_int {0}
5618       \def\thesection{\arabic{chapter}.\arabic{section}}
5619       \def\part@prefix{\arabic{chapter}.}
5620     }
5621     {chapter}{
5622       \int_set:Nn \l_document_structure_section_level_int {1}
5623       \def\thesection{\arabic{chapter}.\arabic{section}}
5624       \def\part@prefix{\arabic{chapter}.}
5625     }
5626   }{
5627     \int_set:Nn \l_document_structure_section_level_int {2}
5628     \def\part@prefix{}

```

```

5629 }
5630 }
5631
5632 \bool_if:NF \c__notesslides_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

```

5633 \renewenvironment{omgroup}[2][]{
5634   \__document_structure_omgroup_args:n { #1 }
5635   \int_incr:N \l_document_structure_omgroup_level_int
5636   \int_incr:N \l_document_structure_section_level_int
5637   \bool_if:NT \c__notesslides_sectocframes_bool {
5638     \stepcounter{slide}
5639     \begin{frame}[noframenumbering]
5640     \vfill\Large\centering
5641     \red{
5642       \ifcase\l_document_structure_section_level_int\or
5643         \stepcounter{part}
5644         \def\__notesslideslabel{\omdoc@part@kw~\Roman{part}}
5645         \def\currentsectionlevel{\omdoc@part@kw}
5646       \or
5647         \stepcounter{chapter}
5648         \def\__notesslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
5649         \def\currentsectionlevel{\omdoc@chapter@kw}
5650       \or
5651         \stepcounter{section}
5652         \def\__notesslideslabel{\part@prefix\arabic{section}}
5653         \def\currentsectionlevel{\omdoc@section@kw}
5654       \or
5655         \stepcounter{subsection}
5656         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
5657         \def\currentsectionlevel{\omdoc@subsection@kw}
5658       \or
5659         \stepcounter{subsubsection}
5660         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
5661         \def\currentsectionlevel{\omdoc@subsubsection@kw}
5662       \or
5663         \stepcounter{paragraph}
5664         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
5665         \def\currentsectionlevel{\omdoc@paragraph@kw}
5666       \else
5667         \def\__notesslideslabel{}
5668         \def\currentsectionlevel{\omdoc@paragraph@kw}
5669       \fi% end ifcase
5670       \__notesslideslabel%\sref@label@id\__notesslideslabel
5671       \quad #2%
5672     }%
5673     \vfill%
5674     \end{frame}%
5675   }
5676   \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str%

```

```

5677 }{}
5678 }

```

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

```

5679 \def\inserttheorembodyfont{\normalfont}
5680 %\bool_if:NF \c__notesslides_notes_bool {
5681 % \defbeamertemplate{theorem begin}{miko}
5682 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
5683 % \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
5684 % \inserttheorempunctuation\inserttheorembodyfont\hspace}
5685 % \defbeamertemplate{theorem end}{miko}{}}

```

and we set it as the default one.

```

5686 % \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.

```

5687 % \expandafter\def\csname Parent2\endcsname{}
5688 %}
5689
5690 \AddToHook{begindocument}{% this does not work for some reasons
5691 \setbeamertemplate{theorems}[ams style]
5692 }
5693 \bool_if:NT \c__notesslides_notes_bool {
5694 \renewenvironment{columns}[1][{}]{%
5695 \par\noindent%
5696 \begin{minipage}%
5697 \slidewidth\centering\leavevmode%
5698 }{}%
5699 \end{minipage}\par\noindent%
5700 }%
5701 \newsavebox\columnbox%
5702 \renewenvironment<>{column}[2][{}]{%
5703 \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
5704 }{}%
5705 \end{minipage}\end{lrbox}\usebox\columnbox%
5706 }%
5707 }
5708 \bool_if:NTF \c__notesslides_noproblems_bool {
5709 \newenvironment{problems}{}{}
5710 }{
5711 \excludecomment{problems}
5712 }

```

## 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.

```

5713 \gdef\printexcursions{}
5714 \newcommand\excursionref[2]{% label, text
5715 \bool_if:NT \c__notesslides_notes_bool {

```

```

5716 \begin{sparagraph}[title=Excursion]
5717 #2 \sref[fallback=the appendix]{#1}.
5718 \end{sparagraph}
5719 }
5720 }
5721 \newcommand\activate@excursion[2][]{
5722 \gappto\printexcursions{\inputref{#1}{#2}}
5723 }
5724 \newcommand\excursion[4][]{% repos, label, path, text
5725 \bool_if:NT \c__notesslides_notes_bool {
5726 \activate@excursion{#1}{#3}\excursionref{#2}{#4}
5727 }
5728 }

```

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

\excursiongroup

```

5729 \keys_define:nn{notesslides / excursiongroup }{
5730 id .str_set_x:N = \l__notesslides_excursion_id_str,
5731 intro .tl_set:N = \l__notesslides_excursion_intro_tl,
5732 mhrepos .str_set_x:N = \l__notesslides_excursion_mhrepos_str
5733 }
5734 \cs_new_protected:Nn \__notesslides_excursion_args:n {
5735 \tl_clear:N \l__notesslides_excursion_intro_tl
5736 \str_clear:N \l__notesslides_excursion_id_str
5737 \str_clear:N \l__notesslides_excursion_mhrepos_str
5738 \keys_set:nn {notesslides / excursiongroup }{ #1 }
5739 }
5740 \newcommand\excursiongroup[1][]{
5741 \__notesslides_excursion_args:n{ #1 }
5742 \ifdefempty\printexcursions{}% only if there are excursions
5743 {\begin{note}
5744 \begin{omgroup}[#1]{Excursions}%
5745 \ifdefempty\l__notesslides_excursion_intro_tl}{
5746 \inputref[\l__notesslides_excursion_mhrepos_str]{
5747 \l__notesslides_excursion_intro_tl
5748 }
5749 }
5750 \printexcursions%
5751 \end{omgroup}
5752 \end{note}}
5753 }
5754 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
5755 \end{package}

```

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



## Chapter 40

# The Implementation

### 40.1 Package Options

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

```
5756 <*package>
5757 <@@=problems>
5758 \ProvidesExplPackage{problem}{2019/03/20}{1.3}{Semantic Markup for Problems}
5759 \RequirePackage{l3keys2e,expl-keystr-compat}
5760
5761 \keys_define:nn { problem / pkg }{
5762   notes      .default:n    = { true },
5763   notes      .bool_set:N   = \c__problems_notes_bool,
5764   gnotes     .default:n    = { true },
5765   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
5766   hints      .default:n    = { true },
5767   hints      .bool_set:N   = \c__problems_hints_bool,
5768   solutions  .default:n    = { true },
5769   solutions  .bool_set:N   = \c__problems_solutions_bool,
5770   pts        .default:n    = { true },
5771   pts        .bool_set:N   = \c__problems_pts_bool,
5772   min        .default:n    = { true },
5773   min        .bool_set:N   = \c__problems_min_bool,
5774   boxed      .default:n    = { true },
5775   boxed      .bool_set:N   = \c__problems_boxed_bool,
5776   unknown    .code:n       = {}
5777 }
5778 \newif\ifsolutions
5779
5780 \ProcessKeysOptions{ problem / pkg }
5781 \bool_if:NTF \c__problems_solutions_bool {
5782   \solutionstrue
5783 }{
5784   \solutionsfalse
5785 }
```

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

```
5786 \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.

```
5787 \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.

```
5788 \def\prob@problem@kw{Problem}
5789 \def\prob@solution@kw{Solution}
5790 \def\prob@hint@kw{Hint}
5791 \def\prob@note@kw{Note}
5792 \def\prob@gnote@kw{Grading}
5793 \def\prob@pt@kw{pt}
5794 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
5795 \AddToHook{begindocument}{
5796   \ltx@ifpackageloaded{babel}{
5797     \makeatletter
5798     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5799     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5800       \input{problem-ngerman.ldf}
5801     }
5802     \clist_if_in:NnT \l_tmpa_clist {finnish}{
5803       \input{problem-finnish.ldf}
5804     }
5805     \clist_if_in:NnT \l_tmpa_clist {french}{
5806       \input{problem-french.ldf}
5807     }
5808     \clist_if_in:NnT \l_tmpa_clist {russian}{
5809       \input{problem-russian.ldf}
5810     }
5811     \makeatother
5812   }{ }
5813 }
```

## 40.2 Problems and Solutions

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

```
5814 \keys_define:nn{ problem / problem }{
5815   id      .str_set:x:N = \l__problems_prob_id_str,
5816   pts     .tl_set:N    = \l__problems_prob_pts_tl,
5817   min     .tl_set:N    = \l__problems_prob_min_tl,
5818   title   .tl_set:N    = \l__problems_prob_title_tl,
5819   type    .tl_set:N    = \l__problems_prob_type_tl,
5820   refnum  .int_set:N   = \l__problems_prob_refnum_int
5821 }
5822 \cs_new_protected:Nn \__problems_prob_args:n {
```

```

5823 \str_clear:N \l__problems_prob_id_str
5824 \tl_clear:N \l__problems_prob_pts_tl
5825 \tl_clear:N \l__problems_prob_min_tl
5826 \tl_clear:N \l__problems_prob_title_tl
5827 \tl_clear:N \l__problems_prob_type_tl
5828 \int_zero_new:N \l__problems_prob_refnum_int
5829 \keys_set:nn { problem / problem }{ #1 }
5830 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
5831   \let\l__problems_prob_refnum_int\undefined
5832 }
5833 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

5834 \newcounter{problem}
5835 \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.

```

5836 \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

```

5837 \newcommand\prob@number{
5838   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
5839     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
5840   }{
5841     \int_if_exist:NTF \l__problems_prob_refnum_int {
5842       \prob@label{\int_use:N \l__problems_prob_refnum_int }
5843     }{
5844       \prob@label\theproblem
5845     }
5846   }
5847 }

```

(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.

```

5848 \newcommand\prob@title[3]{%
5849   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
5850     #2 \l__problems_inclprob_title_tl #3
5851   }{
5852     \tl_if_exist:NTF \l__problems_prob_title_tl {
5853       #2 \l__problems_prob_title_tl #3
5854     }{
5855       #1
5856     }
5857   }
5858 }

```

(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.

```
5859 \def\prob@heading{
5860   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
5861   %\sref@label{id{\prob@problem@kw~\prob@number}}{~}
5862 }
```

(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.

`sproblem`

```
5863 \newenvironment{sproblem}[1][{}]{
5864   \__problems_prob_args:n{#1}%\sref@target%
5865   \@in@omtexttrue% we are in a statement (for inline definitions)
5866   \stepcounter{problem}\record@problem
5867   \def\current@section@level{\prob@problem@kw}
5868   \tl_if_exist:NTF \l__problems_inclprob_type_tl {
5869     \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
5870   }{
5871     \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
5872   }
5873   \str_if_exist:NTF \l__problems_inclprob_id_str {
5874     \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
5875   }{
5876     \str_set_eq:NN \sproblemid \l__problems_prob_id_str
5877   }
5878
5879
5880   \clist_set:No \l_tmpa_clist \sproblemtype
5881   \tl_clear:N \l_tmpa_tl
5882   \clist_map_inline:Nn \l_tmpa_clist {
5883     \tl_if_exist:cT {\__problems_sproblem_##1_start:}{
5884       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_start:}}
5885     }
5886   }
5887   \tl_if_empty:NTF \l_tmpa_tl {
5888     \__problems_sproblem_start:
5889   }{
5890     \l_tmpa_tl
5891   }
5892   \stex_ref_new_doc_target:n \sproblemid
5893 }{
5894   \clist_set:No \l_tmpa_clist \sproblemtype
5895   \tl_clear:N \l_tmpa_tl
5896   \clist_map_inline:Nn \l_tmpa_clist {
5897     \tl_if_exist:cT {\__problems_sproblem_##1_end:}{
5898       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_end:}}
5899     }
5900   }
```

```

5900 }
5901 \tl_if_empty:NTF \l_tmpa_tl {
5902   \__problems_sproblem_end:
5903 }{
5904   \l_tmpa_tl
5905 }
5906
5907
5908 \smallskip
5909 }
5910
5911
5912 \cs_new_protected:Nn \__problems_sproblem_start: {
5913   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
5914 }
5915 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
5916
5917 \newcommand\stexpatchproblem[3][] {
5918   \str_set:Nx \l_tmpa_str{ #1 }
5919   \str_if_empty:NTF \l_tmpa_str {
5920     \tl_set:Nn \__problems_sproblem_start: { #2 }
5921     \tl_set:Nn \__problems_sproblem_end: { #3 }
5922   }{
5923     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
5924     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
5925   }
5926 }
5927
5928
5929 \bool_if:NT \c__problems_boxed_bool {
5930   \surroundwithmdframed{problem}
5931 }

```

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

```

5932 \def\record@problem{
5933   \protected@write\@auxout{}
5934   {
5935     \string\@problem{\prob@number}
5936     {
5937       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
5938         \l__problems_inclprob_pts_tl
5939       }{
5940         \l__problems_prob_pts_tl
5941       }
5942     }%
5943     {
5944       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
5945         \l__problems_inclprob_min_tl
5946       }{
5947         \l__problems_prob_min_tl
5948       }
5949     }
5950   }
5951 }

```

(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).

```
5952 \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.

```
5953 \keys_define:nn { problem / solution }{
5954   id                .str_set_x:N = \l__problems_solution_id_str ,
5955   for               .tl_set:N   = \l__problems_solution_for_tl ,
5956   height            .dim_set:N  = \l__problems_solution_height_dim ,
5957   creators          .clist_set:N = \l__problems_solution_creators_clist ,
5958   contributors      .clist_set:N = \l__problems_solution_contributors_clist ,
5959   srccite           .tl_set:N   = \l__problems_solution_srccite_tl
5960 }
5961 \cs_new_protected:Nn \__problems_solution_args:n {
5962   \str_clear:N \l__problems_solution_id_str
5963   \tl_clear:N \l__problems_solution_for_tl
5964   \tl_clear:N \l__problems_solution_srccite_tl
5965   \clist_clear:N \l__problems_solution_creators_clist
5966   \clist_clear:N \l__problems_solution_contributors_clist
5967   \dim_zero:N \l__problems_solution_height_dim
5968   \keys_set:nn { problem / solution }{ #1 }
5969 }
```

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

```
5970 \newcommand\@startsolution[1][{}]{
5971   \__problems_solution_args:n { #1 }
5972   \@in@omtexttrue% we are in a statement.
5973   \bool_if:NF \c__problems_boxed_bool { \hrule }
5974   \smallskip\noindent
5975   {\textbf\prob@solution@kw : \enspace}
5976   \begin{small}
5977   \def\current@section@level{\prob@solution@kw}
5978   \ignorespacesandpars
5979 }
```

**\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.

```
5980 \newcommand\startsolutions{
5981   \specialcomment{solution}{\@startsolution}{
5982     \bool_if:NF \c__problems_boxed_bool {
5983       \hrule\medskip
5984     }
5985     \end{small}%
5986   }
5987   \bool_if:NT \c__problems_boxed_bool {
5988     \surroundwithmdframed{solution}
5989   }
5990 }
```

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

\stopsolutions

```
5991 \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.

```
5992 \ifsolutions
5993   \startsolutions
5994 \else
5995   \stopsolutions
5996 \fi
```

exnote

```
5997 \bool_if:NTF \c__problems_notes_bool {
5998   \newenvironment{exnote}[1][]{
5999     \par\smallskip\hrule\smallskip
6000     \noindent\textbf{\prob@note@kw : }\small
6001   }{
6002     \smallskip\hrule
6003   }
6004 }{
6005   \excludecomment{exnote}
6006 }
```

hint

```
6007 \bool_if:NTF \c__problems_notes_bool {
6008   \newenvironment{hint}[1][]{
6009     \par\smallskip\hrule\smallskip
6010     \noindent\textbf{\prob@hint@kw :~ }\small
6011   }{
6012     \smallskip\hrule
6013   }
6014 \newenvironment{exhint}[1][]{
6015   \par\smallskip\hrule\smallskip
6016   \noindent\textbf{\prob@hint@kw :~ }\small
6017 }{
6018   \smallskip\hrule
6019 }
6020 }{
6021   \excludecomment{hint}
6022   \excludecomment{exhint}
6023 }
```

gnote

```
6024 \bool_if:NTF \c__problems_notes_bool {
6025   \newenvironment{gnote}[1][]{
6026     \par\smallskip\hrule\smallskip
6027     \noindent\textbf{\prob@gnote@kw : }\small
6028   }{
6029     \smallskip\hrule
6030   }
6031 }{
6032   \excludecomment{gnote}
6033 }
```

## 40.3 Multiple Choice Blocks

```

6034 \newenvironment{mcb}{
6035   \begin{enumerate}
6036 }{
6037   \end{enumerate}
6038 }

```

we define the keys for the mcc macro

```

6039 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
6040   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
6041     \bool_set_true:N #1
6042   }{
6043     \bool_set_false:N #1
6044   }
6045 }
6046 \keys_define:nn { problem / mcc }{
6047   id          .str_set_x:N = \l__problems_mcc_id_str ,
6048   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
6049   T           .default:n    = { true } ,
6050   T           .bool_set:N   = \l__problems_mcc_t_bool ,
6051   F           .default:n    = { true } ,
6052   F           .bool_set:N   = \l__problems_mcc_f_bool ,
6053   Ttext       .code:n       = {
6054     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
6055   } ,
6056   Ftext       .code:n       = {
6057     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
6058   }
6059 }
6060 \cs_new_protected:Nn \l__problems_mcc_args:n {
6061   \str_clear:N \l__problems_mcc_id_str
6062   \tl_clear:N \l__problems_mcc_feedback_tl
6063   \bool_set_true:N \l__problems_mcc_t_bool
6064   \bool_set_true:N \l__problems_mcc_f_bool
6065   \bool_set_true:N \l__problems_mcc_Ttext_bool
6066   \bool_set_false:N \l__problems_mcc_Ftext_bool
6067   \keys_set:nn { problem / mcc }{ #1 }
6068 }

```

\mcc

```

6069 \newcommand\mcc[2][]{
6070   \l__problems_mcc_args:n{ #1 }
6071   \item #2
6072   \ifsolutions
6073     \\\
6074     \bool_if:NT \l__problems_mcc_t_bool {
6075       % TODO!
6076       % \ifcsstring{mcc@T}{T}{\mcc@Ttext}%
6077     }
6078     \bool_if:NT \l__problems_mcc_f_bool {

```

---

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



```

6079      % TODO!
6080      % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
6081    }
6082    \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
6083      !
6084    }{
6085      \l__problems_mcc_feedback_tl
6086    }
6087    \fi
6088  } %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.

```

6089
6090 \keys_define:nn{ problem / inclproblem }{
6091   id      .str_set:N = \l__problems_inclprob_id_str,
6092   pts     .tl_set:N  = \l__problems_inclprob_pts_tl,
6093   min     .tl_set:N  = \l__problems_inclprob_min_tl,
6094   title   .tl_set:N  = \l__problems_inclprob_title_tl,
6095   refnum  .int_set:N  = \l__problems_inclprob_refnum_int,
6096   type    .tl_set:N  = \l__problems_inclprob_type_tl,
6097   mhrepos .str_set:N  = \l__problems_inclprob_mhrepos_str
6098 }
6099 \cs_new_protected:Nn \l__problems_inclprob_args:n {
6100   \str_clear:N \l__problems_prob_id_str
6101   \tl_clear:N \l__problems_inclprob_pts_tl
6102   \tl_clear:N \l__problems_inclprob_min_tl
6103   \tl_clear:N \l__problems_inclprob_title_tl
6104   \tl_clear:N \l__problems_inclprob_type_tl
6105   \int_zero_new:N \l__problems_inclprob_refnum_int
6106   \str_clear:N \l__problems_inclprob_mhrepos_str
6107   \keys_set:nn { problem / inclproblem }{ #1 }
6108   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
6109     \let\l__problems_inclprob_pts_tl\undefined
6110   }
6111   \tl_if_empty:NT \l__problems_inclprob_min_tl {
6112     \let\l__problems_inclprob_min_tl\undefined
6113   }
6114   \tl_if_empty:NT \l__problems_inclprob_title_tl {
6115     \let\l__problems_inclprob_title_tl\undefined
6116   }
6117   \tl_if_empty:NT \l__problems_inclprob_type_tl {
6118     \let\l__problems_inclprob_type_tl\undefined
6119   }
6120   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
6121     \let\l__problems_inclprob_refnum_int\undefined
6122   }
6123 }

```

```

6124
6125 \cs_new_protected:Nn \__problems_inclprob_clear: {
6126   \let\l__problems_inclprob_id_str\undefined
6127   \let\l__problems_inclprob_pts_tl\undefined
6128   \let\l__problems_inclprob_min_tl\undefined
6129   \let\l__problems_inclprob_title_tl\undefined
6130   \let\l__problems_inclprob_type_tl\undefined
6131   \let\l__problems_inclprob_refnum_int\undefined
6132   \let\l__problems_inclprob_mhrepos_str\undefined
6133 }
6134 \__problems_inclprob_clear:
6135
6136 \newcommand\includeproblem[2][ ]{
6137   \__problems_inclprob_args:n{ #1 }
6138   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
6139     \input{#2}
6140   }{
6141     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
6142       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
6143     }
6144   }
6145   \__problems_inclprob_clear:
6146 }

```

(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.

```

6147 \AddToHook{enddocument}{
6148   \bool_if:NT \c__problems_pts_bool {
6149     \message{Total:~\arabic{pts}~points}
6150   }
6151   \bool_if:NT \c__problems_min_bool {
6152     \message{Total:~\arabic{min}~minutes}
6153   }
6154 }

```

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

```

6155 \def\pts#1{
6156   \bool_if:NT \c__problems_pts_bool {
6157     \marginpar{#1~\prob@pt@kw}
6158   }
6159 }
6160 \def\min#1{
6161   \bool_if:NT \c__problems_min_bool {
6162     \marginpar{#1~\prob@min@kw}
6163   }
6164 }

```

`\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.

```

6165 \newcounter{pts}
6166 \def\show@pts{
6167   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
6168     \bool_if:NT \c__problems_pts_bool {
6169       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
6170       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
6171     }
6172   }{
6173     \tl_if_exist:NT \l__problems_prob_pts_tl {
6174       \bool_if:NT \c__problems_pts_bool {
6175         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
6176         \addtocounter{pts}{\l__problems_prob_pts_tl}
6177       }
6178     }
6179   }
6180 }

```

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

`\show@min`

```

6181 \newcounter{min}
6182 \def\show@min{
6183   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
6184     \bool_if:NT \c__problems_min_bool {
6185       \marginpar{\l__problems_inclprob_min_tl\ min}
6186       \addtocounter{min}{\l__problems_inclprob_min_tl}
6187     }
6188   }{
6189     \tl_if_exist:NT \l__problems_prob_min_tl {
6190       \bool_if:NT \c__problems_min_bool {
6191         \marginpar{\l__problems_prob_min_tl\ min}
6192         \addtocounter{min}{\l__problems_prob_min_tl}
6193       }
6194     }
6195   }
6196 }
6197 </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.

```
6198 <@@=hwexam>
6199 <*cls>
6200 \ProvidesExplClass{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
6201 \RequirePackage{l3keys2e,expl-keystr-compatible}
6202 \DeclareOption*{
6203   \PassOptionsToClass{\CurrentOption}{document-structure}
6204   \PassOptionsToPackage{\CurrentOption}{stex}
6205   \PassOptionsToPackage{\CurrentOption}{hwexam}
6206   \PassOptionsToPackage{\CurrentOption}{tikzinput}
6207 }
6208 \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.

```
6209 \LoadClass{document-structure}
6210 \RequirePackage{stex}
6211 \RequirePackage{hwexam}
6212 \RequirePackage{tikzinput}
6213 \RequirePackage{graphicx}
6214 \RequirePackage{a4wide}
6215 \RequirePackage{amssymb}
6216 \RequirePackage{amstext}
6217 \RequirePackage{amsmath}
```

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

```

6218 \newcommand\assig@default@type{\hwexam@assignment@kw}
6219 \def\document@hwexamtype{\assig@default@type}
6220 <@@=document_structure>
6221 \keys_define:nn { document-structure / document }{
6222 id .str_set_x:N = \c_document_structure_document_id_str,
6223 hwexamtype .tl_set:N = \document@hwexamtype
6224 }
6225 <@@=hwexam>
6226 </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.

```
6227 \*package>
6228 \ProvidesExplPackage{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
6229 \RequirePackage{l3keys2e,expl-keystr-compat}
6230
6231 \newif\iftest\testfalse
6232 \DeclareOption{test}{\testtrue}
6233 \newif\ifmultiple\multiplefalse
6234 \DeclareOption{multiple}{\multipletrue}
6235 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
6236 \ProcessOptions
```

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

```
6237 \RequirePackage{keyval}[1997/11/10]
6238 \RequirePackage{problem}
```

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

```
6239 \newcommand\hwexam@assignment@kw{Assignment}
6240 \newcommand\hwexam@given@kw{Given}
6241 \newcommand\hwexam@due@kw{Due}
6242 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
6243 blank~for~extra~space}
6244 \def\hwexam@minutes@kw{minutes}
6245 \newcommand\correction@probs@kw{prob.}
6246 \newcommand\correction@pts@kw{total}
6247 \newcommand\correction@reached@kw{reached}
6248 \newcommand\correction@sum@kw{Sum}
6249 \newcommand\correction@grade@kw{grade}
6250 \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

```

6251 \AddToHook{begindocument}{
6252 \ltx@ifpackageloaded{babel}{
6253 \makeatletter
6254 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6255 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6256 \input{hwexam-ngerman.ldf}
6257 }
6258 \clist_if_in:NnT \l_tmpa_clist {finnish}{
6259 \input{hwexam-finnish.ldf}
6260 }
6261 \clist_if_in:NnT \l_tmpa_clist {french}{
6262 \input{hwexam-french.ldf}
6263 }
6264 \clist_if_in:NnT \l_tmpa_clist {russian}{
6265 \input{hwexam-russian.ldf}
6266 }
6267 \makeatother
6268 }{}
6269 }
6270

```

## 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.

```

6271 \newcounter{assignment}
6272 \numberproblemsin{assignment}
6273 \renewcommand\prob@label[1]{\assignment@number.#1}

```

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

```

6274 \keys_define:nn { hwexam / assignment } {
6275 id .str_set:x:N = \l__hwexam_assign_id_str,
6276 number .int_set:N = \l__hwexam_assign_number_int,
6277 title .tl_set:N = \l__hwexam_assign_title_tl,
6278 type .tl_set:N = \l__hwexam_assign_type_tl,
6279 given .tl_set:N = \l__hwexam_assign_given_tl,
6280 due .tl_set:N = \l__hwexam_assign_due_tl,
6281 loadmodules .code:n = {
6282 \bool_set_true:N \l__hwexam_assign_loadmodules_bool
6283 }
6284 }
6285 \cs_new_protected:Nn \__hwexam_assignment_args:n {
6286 \str_clear:N \l__hwexam_assign_id_str
6287 \int_set:Nn \l__hwexam_assign_number_int {-1}
6288 \tl_clear:N \l__hwexam_assign_title_tl
6289 \tl_clear:N \l__hwexam_assign_type_tl
6290 \tl_clear:N \l__hwexam_assign_given_tl
6291 \tl_clear:N \l__hwexam_assign_due_tl
6292 \bool_set_false:N \l__hwexam_assign_loadmodules_bool

```

```

6293 \keys_set:nn { hwexam / assignment }{ #1 }
6294 }

```

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.

```

6295 \newcommand\given@due[2]{
6296 \bool_lazy_all:nF {
6297 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
6298 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
6299 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
6300 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
6301 }{ #1 }
6302
6303 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
6304 \tl_if_empty:NF \l__hwexam_assign_given_tl {
6305 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
6306 }
6307 }{
6308 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
6309 }
6310
6311 \bool_lazy_or:nnF {
6312 \bool_lazy_and_p:nn {
6313 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
6314 }{
6315 \tl_if_empty_p:V \l__hwexam_assign_due_tl
6316 }
6317 }{
6318 \bool_lazy_and_p:nn {
6319 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
6320 }{
6321 \tl_if_empty_p:V \l__hwexam_assign_due_tl
6322 }
6323 }{ ,~ }
6324
6325 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
6326 \tl_if_empty:NF \l__hwexam_assign_due_tl {
6327 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
6328 }
6329 }{
6330 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
6331 }
6332
6333 \bool_lazy_all:nF {
6334 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
6335 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
6336 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
6337 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
6338 }{ #2 }
6339 }

```

`\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.

```

6340 \newcommand\assignment@title[3]{
6341 \tl_if_empty:NTF \l__hwexam_inclasssign_title_tl {
6342 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
6343 #1
6344 }{
6345 #2\l__hwexam_assign_title_tl#3
6346 }
6347 }{
6348 #2\l__hwexam_inclasssign_title_tl#3
6349 }
6350 }

```

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

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

```

6351 \newcommand\assignment@number{
6352 \int_compare:nNnTF \l__hwexam_inclasssign_number_int = {-1} {
6353 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
6354 \arabic{assignment}
6355 } {
6356 \int_use:N \l__hwexam_assign_number_int
6357 }
6358 }{
6359 \int_use:N \l__hwexam_inclasssign_number_int
6360 }
6361 }

```

(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.

```

6362 \newenvironment{assignment}[1][]{
6363 \__hwexam_assignment_args:n { #1 }
6364 %\sref@target
6365 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
6366 \global\stepcounter{assignment}
6367 }{
6368 \global\setcounter{assignment}{\int_use:N\l__hwexam_assign_number_int}
6369 }
6370 \setcounter{problem}{0}
6371 \def\current@section@level{\document@hwexamtype}
6372 %\sref@label@id{\document@hwexamtype \thesection}
6373 \begin{@assignment}
6374 }{
6375 \end{@assignment}
6376 }

```

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

```

6377 \def\ass@title{
6378 \protect\document@hwexamtype~\arabic{assignment}
6379 \assignment@title{}\{;\}\{;\} -- \given@due{}\{;\}
6380 }
6381 \ifmultiple
6382 \newenvironment{@assignment}{
6383 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
6384 \begin{omgroup}[loadmodules]{\ass@title}
6385 }{
6386 \begin{omgroup}{\ass@title}
6387 }
6388 }{
6389 \end{omgroup}
6390 }

```

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

```

6391 \else
6392 \newenvironment{@assignment}{
6393 \begin{center}\bf
6394 \Large@title\strut\\
6395 \document@hwexamtype~\arabic{assignment}\assignment@title{}\{;\}\{;\}
6396 \large\given@due{--;\}\{;\}
6397 \end{center}
6398 }{}
6399 \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.

```

6400 \keys_define:nn { hwexam / inclassignment } {
6401 %id .str_set_x:N = \l__hwexam_assign_id_str,
6402 number .int_set:N = \l__hwexam_inclassign_number_int,
6403 title .tl_set:N = \l__hwexam_inclassign_title_tl,
6404 type .tl_set:N = \l__hwexam_inclassign_type_tl,
6405 given .tl_set:N = \l__hwexam_inclassign_given_tl,
6406 due .tl_set:N = \l__hwexam_inclassign_due_tl,
6407 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
6408 }
6409 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
6410 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
6411 \tl_clear:N \l__hwexam_inclassign_title_tl
6412 \tl_clear:N \l__hwexam_inclassign_type_tl
6413 \tl_clear:N \l__hwexam_inclassign_given_tl
6414 \tl_clear:N \l__hwexam_inclassign_due_tl
6415 \str_clear:N \l__hwexam_inclassign_mhrepos_str
6416 \keys_set:nn { hwexam / inclassignment }{ #1 }
6417 }
6418 \__hwexam_inclassignment_args:n {}
6419
6420 \newcommand\inputassignment[2][{}]{

```

```

6421 \_hwexam_inclassnment_args:n { #1 }
6422 \str_if_empty:NTF \l__hwexam_inclassn_mhrepos_str {
6423 \input{#2}
6424 }{
6425 \stex_in_repository:nn{\l__hwexam_inclassn_mhrepos_str}{
6426 \input{\mhp{path}\l__hwexam_inclassn_mhrepos_str}{#2}}
6427 }
6428 }
6429 \_hwexam_inclassnment_args:n {}
6430 }
6431 \newcommand\includeassignment[2][ ]{
6432 \newpage
6433 \inputassignment[#1]{#2}
6434 }

```

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

## 42.4 Typesetting Exams

\quizheading

```

6435 \ExplSyntaxOff
6436 \newcommand\quizheading[1]{%
6437 \def\@tas{#1}%
6438 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
6439 \ifx\@tas\@empty\else%
6440 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
6441 \fi%
6442 }
6443 \ExplSyntaxOn

```

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

\testheading

```

6444
6445 \def\hwexamheader{\input{hwexam-default.header}}
6446
6447 \def\hwexamminutes{
6448 \tl_if_empty:NTF \testheading@duration {
6449 {\testheading@min}~\hwexam@minutes@kw
6450 }{
6451 \testheading@duration
6452 }
6453 }
6454
6455 \keys_define:nn { hwexam / testheading } {
6456 min .tl_set:N = \testheading@min,
6457 duration .tl_set:N = \testheading@duration,
6458 reqpts .tl_set:N = \testheading@reqpts,
6459 tools .tl_set:N = \testheading@tools
6460 }
6461 \cs_new_protected:Nn \_hwexam_testheading_args:n {
6462 \tl_clear:N \testheading@min
6463 \tl_clear:N \testheading@duration

```

```

6464 \tl_clear:N \testheading@reqpts
6465 \tl_clear:N \testheading@tools
6466 \keys_set:nn { hwexam / testheading }{ #1 }
6467 }
6468 \newenvironment{testheading}[1][]{
6469   \_hwexam_testheading_args:n{ #1 }
6470   \newcount\check@time\check@time=\testheading@min
6471   \advance\check@time by -\theassignment@totalmin
6472   \newif\if@bonuspoints
6473   \tl_if_empty:NTF \testheading@reqpts {
6474     \@bonuspointsfalse
6475   }{
6476     \newcount\bonus@pts
6477     \bonus@pts=\theassignment@totalpts
6478     \advance\bonus@pts by -\testheading@reqpts
6479     \edef\bonus@pts{\the\bonus@pts}
6480     \@bonuspointstrue
6481   }
6482   \edef\check@time{\the\check@time}
6483
6484   \makeatletter\hwexamheader\makeatother
6485 }{
6486   \newpage
6487 }

```

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

\testspace

```

6488 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

6489 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

6490 \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.

```

6491 \@@=problems
6492 \renewcommand\@problem[3]{
6493   \stepcounter{assignment@probs}
6494   \def\__problemspts{#2}
6495   \ifx\__problemspts\@empty\else
6496     \addtocounter{assignment@totalpts}{#2}
6497   \fi
6498   \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
6499   \xdef\correction@probs{\correction@probs & #1}%
6500   \xdef\correction@pts{\correction@pts & #2}
6501   \xdef\correction@reached{\correction@reached &}

```

```

6502 }
6503 <@@=hwexam>

```

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

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

```

6504 \newcounter{assignment@probs}
6505 \newcounter{assignment@totalpts}
6506 \newcounter{assignment@totalmin}
6507 \def\correction@probs{\correction@probs@kw}
6508 \def\correction@pts{\correction@pts@kw}
6509 \def\correction@reached{\correction@reached@kw}
6510 \stepcounter{assignment@probs}
6511 \newcommand\correction@table{
6512 \resizebox{\textwidth}{!}{%
6513 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
6514 &\multicolumn{\theassignment@probs}{c|}||%|
6515 {\footnotesize\correction@forgrading@kw} &\\ \hline
6516 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
6517 \correction@pts & \theassignment@totalpts & \\ \hline
6518 \correction@reached & & \[.7cm]\hline
6519 \end{tabular}}
6520 </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\bierfont=../assignments/bierglas
\font\denkerfont=../assignments/denker
\font\uhrfont=../assignments/uhr
\font\warnschildfont=../assignments/achtung

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

```