

# The sTeX3 Package \*

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

2022-02-16

## 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 2022-02-16)

# 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 . . . . .	112
<b>30</b>	<b>STeX-Symbols Implementation</b>	<b>117</b>
30.1	Symbol Declarations . . . . .	117
30.2	Notations . . . . .	124
<b>31</b>	<b>STeX-Terms Implementation</b>	<b>134</b>
31.1	Symbol Invocations . . . . .	134
31.2	Terms . . . . .	137
31.3	Notation Components . . . . .	144
<b>32</b>	<b>STeX-Structural Features Implementation</b>	<b>147</b>
32.1	Imports with modification . . . . .	147
32.2	The feature environment . . . . .	154
32.3	Features . . . . .	156
<b>33</b>	<b>STeX-Statements Implementation</b>	<b>161</b>
33.1	Definitions . . . . .	161
33.2	Assertions . . . . .	166
33.3	Examples . . . . .	169
33.4	Logical Paragraphs . . . . .	171

<b>34 The Implementation</b>	<b>176</b>
34.1 Package Options . . . . .	176
34.2 Proofs . . . . .	176
34.3 Justifications . . . . .	182
<b>35 <math>\text{\LaTeX}</math>-Others Implementation</b>	<b>184</b>
<b>36 <math>\text{\LaTeX}</math>-Metatheory Implementation</b>	<b>185</b>
<b>37 Tikzinput Implementation</b>	<b>188</b>
<b>38 document-structure.sty Implementation</b>	<b>190</b>
38.1 The document-structure Class . . . . .	190
38.2 Class Options . . . . .	190
38.3 Beefing up the <code>document</code> environment . . . . .	191
38.4 Implementation: document-structure Package . . . . .	191
38.5 Package Options . . . . .	191
38.6 Document Structure . . . . .	193
38.7 Front and Backmatter . . . . .	196
38.8 Global Variables . . . . .	198
<b>39 NotesSlides – Implementation</b>	<b>199</b>
39.1 Class and Package Options . . . . .	199
39.2 Notes and Slides . . . . .	201
39.3 Header and Footer Lines . . . . .	205
39.4 Frame Images . . . . .	206
39.5 Colors and Highlighting . . . . .	207
39.6 Sectioning . . . . .	208
39.7 Excursions . . . . .	210
<b>40 The Implementation</b>	<b>212</b>
40.1 Package Options . . . . .	212
40.2 Problems and Solutions . . . . .	213
40.3 Multiple Choice Blocks . . . . .	219
40.4 Including Problems . . . . .	220
40.5 Reporting Metadata . . . . .	221
<b>41 Implementation: The hwexam Class</b>	<b>223</b>
41.1 Class Options . . . . .	223
<b>42 Implementation: The hwexam Package</b>	<b>225</b>
42.1 Package Options . . . . .	225
42.2 Assignments . . . . .	226
42.3 Including Assignments . . . . .	229
42.4 Typesetting Exams . . . . .	230
42.5 Leftovers . . . . .	232

**Part I**  
**Manual**

# Chapter 1

## What is sTeX?

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

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

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

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

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

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



# Chapter 2

## Quickstart

### 2.1 Setup

#### 2.1.1 The sTeX IDE

TODO: VSCode Plugin

#### 2.1.2 Manual Setup

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

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

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

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

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

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

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

---

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

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

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

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

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

The document we will consider is the following:

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

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

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

Compiling this document with `pdflatex` should yield the output

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

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

### \usemodule

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

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

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

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

---

`\symref`  
`\symname`

---

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

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

---

`\importmodule`

---

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

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

## Chapter 3

# Using Semantic Macros

TODO

## Chapter 4

# TeX Archives

### 4.1 The Local MathHub-Directory

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

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

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

### 4.2 The Structure of TeX Archives

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

Each such archive needs two subdirectories:

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

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

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

### 4.3 MANIFEST.MF-Files

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

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

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

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

`source-base` or

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

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

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

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

## Chapter 5

# Creating New Modules and Symbols

TODO

### Example 1

```
\begin{smodule}{assoctest}
\symdef[ args=1 a ]{foo}{\comp{a:}#1\comp{; b:}#2\comp{; c:}#3}{\comp[#1\comp{;}# #1\comp{##2\comp{;#2\comp{}}]
$ \foo {w_1}{w_2}{x,y,z}$
\end{smodule}
```

Module 1:  $a : w_1; b : w_2; c : [w_1; x + [w_1; y + z; w_2]; w_2]$

## 5.1 Advanced Structuring Mechanisms

Given modules:

### Example 2

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

Module 2:  
Module 3:  
Module 4:

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 3

```
\begin{smodule}{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}{\compl}
Test: $\rtimes a{\rplus c{\rtimes de}}$
\end{smodule}
```

Module 5:

Test:  $a \circ a$

TODO: explain donotclone

### Example 4

```
\begin{smodule}{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{smodule}
```

Module 6:

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

---

### 8.1.1 Semantic Macros and Notations

Semantic macros invoke a formally declared symbol.

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

**Module 7:** 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 5

```
\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 6

```
\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 7

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

```
\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 9

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

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

<sup>4</sup>EdNOTE: TODO

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

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

### Example 10

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

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

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

### Example 11

`\mult![\comp{Multiplication}]` (denoted by `$\mult*![\comp\cdot]$\`) 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}
```

**Module 8:** 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 12

```
\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 13

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

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

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

5 6

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

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

## Precedences

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

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

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

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

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

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

For example:

**Module 9:**

### Example 14

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

```
 $a + b \cdot c$  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.

**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 <b>debug</b> 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> X <sub>2</sub> e and L <sup>A</sup> T <sub>E</sub> X <sub>3</sub> conditionals for L <sup>A</sup> T <sub>E</sub> X <sub>ML</sub> .
<code>\latexml_if_p:</code>	
<code>\latexml_if:T</code>	
<code>\latexml_if:F</code>	
<code>\latexml_if:TF</code>	

---

We have four macros for annotating generated HTML (via L<sup>A</sup>T<sub>E</sub>X<sub>ML</sub> or 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://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{smodule}{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{smodule}
\ExplSyntaxOff

```

```

Module 10:  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{smodule}[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{smodule}
\ExplSyntaxOff
```

Module 11: 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{smodule}{STEXModuleTest1}
\symdecl{foo}
\end{smodule}
\begin{smodule}{STEXModuleTest2}
\importmodule{STEXModuleTest1}
\symdecl{foo}
\end{smodule}
\begin{smodule}{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{smodule}
```

Module 12:  
 Module 13:  
 Module 14: file://stextest?STEXModuleTest1  
 file://stextest?STEXModuleTest2  
 file://stextest?STEXModuleTest3  
 foo1  
 foo2  
 foo3

---

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

```
Module 15:    Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?Foo?foo}<
              Meaning: >macro:->\protect \bar <
Module 16:    Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?Foo?foo}<
Module 17:    Meaning: >macro:->\stex_invoke_symbol:n {file://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{smodule}{UseTest1}
\symdecl{foo}
\end{smodule}
\begin{smodule}{UseTest2}
\usemodule{UseTest1}
\symdecl{bar}
Meaning:- \present\foo\
\end{smodule}
\begin{smodule}{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{smodule}

```

Module 18:  
Module 19:      Meaning: >macro:->\stex\_invoke\_symbol:n {file://stextest?UseTest1?foo}<  
Module 20:      Meaning: >undefined<  
Meaning: >macro:->\stex\_invoke\_symbol:n {file://stextest?UseTest2?bar}<

All modules: <http://mathhub.info/sTeX?Metatheory, file://stextest?UseTest3, file://stextest?UseTest2>  
All symbols: <http://mathhub.info/sTeX?Metatheory?isa, http://mathhub.info/sTeX?Metatheory?bind, http://mathhub.info/sTeX?Metatheory?collection, http://mathhub.info/sTeX?Metatheory?fromto, http://mathhub.info/sTeX?Metatheory?apply, http://mathhub.info/sTeX?Metatheory?collection, http://mathhub.info/sTeX?Metatheory?seqtype, http://mathhub.info/sTeX?Metatheory?sequence-index, http://mathhub.info/sTeX?Metatheory?seqfromto, 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?collection, http://mathhub.info/sTeX?Metatheory?seqtype, http://mathhub.info/sTeX?Metatheory?sequence-index, http://mathhub.info/sTeX?Metatheory?seqfromto, http://mathhub.info/sTeX?Metatheory?module-type, http://mathhub.info/sTeX?Metatheory?mathematical-structure, file://stextest?UseTest2?bar>

## Test 10

```

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

```

Circular dependencies:  
Module 21:      >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}<

<hr/> <hr/>	<hr/>
<code>\stex_import_module_uri:nn</code>	<code>\stex_import_module_uri:nn {\langle archive-ID \rangle} {\langle module-path \rangle}</code>
	<p>Determines the URI of a module by splitting <math>\langle module-path \rangle</math> into <math>\langle path \rangle ? \langle name \rangle</math>. If <math>\langle module-path \rangle</math> does <i>not</i> contain a ?-character, we consider it to be the <math>\langle name \rangle</math>, and <math>\langle path \rangle</math> to be empty.</p> <p>If <math>\langle archive-ID \rangle</math> is empty, it is automatically set to the ID of the current archive (if one exists).</p> <ol style="list-style-type: none"> <li>1. If <math>\langle archive-ID \rangle</math> is empty: <ol style="list-style-type: none"> <li>(a) If <math>\langle path \rangle</math> is empty, then <math>\langle name \rangle</math> must have been declared earlier in the same file and retrievable from <code>\g_stex_modules_in_file_seq</code>, or a file with name <math>\langle name \rangle . \langle lang \rangle . \text{tex}</math> must exist in the same folder, containing a module <math>\langle name \rangle</math>. That module should have the same namespace as the current one.</li> <li>(b) If <math>\langle path \rangle</math> is not empty, it must point to the relative path of the containing file as well as the namespace.</li> </ol> </li> <li>2. Otherwise: <ol style="list-style-type: none"> <li>(a) If <math>\langle path \rangle</math> is empty, then <math>\langle name \rangle</math> must have been declared earlier in the same file and retrievable from <code>\g_stex_modules_in_file_seq</code>, or a file with name <math>\langle name \rangle . \langle lang \rangle . \text{tex}</math> must exist in the top <b>source</b> folder of the archive, containing a module <math>\langle name \rangle</math>. That module should lie directly in the namespace of the archive.</li> <li>(b) If <math>\langle path \rangle</math> 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 <code>\stex_require_module:nn</code> on the <b>source</b> directory of the archive to find the file.</li> </ol> </li> </ol>

---

---

<code>\stex_import_require_module:nnnn</code>	<code>{\langle ns \rangle} {\langle archive-ID \rangle} {\langle path \rangle} {\langle name \rangle}</code>
---	--

Checks whether a module with URI  $\langle ns \rangle ? \langle name \rangle$  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

---

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

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

- **name**: An (OMDOC) name. By default equal to `⟨macroname⟩`.
- **type**: An (ideally semantic) term. Not used by TeX, but passed on to MMT for semantic services.
- **local**: A boolean (by default false). If set, this declaration will not be added to the module content, i.e. importing the current module will not make this declaration available.
- **args**: Specifies the “signature” of the semantic macro. Can be either an integer  $0 \leq n \leq 9$ , or a (more precise) sequence of the following characters:
  - i a “normal” argument, e.g. `\symdecl[args=ii]{plus}` allows for `\plus{2}{2}`.
  - a an *associative* argument; i.e. a sequence of arbitrarily many arguments provided as a comma-separated list, e.g. `\symdecl[args=a]{plus}` allows for `\plus{2,2,2}`.
  - b a *variable* argument. Is treated by TeX like an i-argument, but an application is turned into an OMBind in OMDoc, binding the provided variable in the subsequent arguments of the operator; e.g. `\symdecl[args=bi]{forall}` allows for `\forall{x\in\Nat}{x\geq0}`.

---

---

`\stex_symdecl_do:n`

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

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

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

### Test 11

```
\begin{smodule}{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{smodule}
```

```
Module 22:      Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?SymdeclTest?foo}<
Result: file://stextest?SymdeclTest?foo
Meaning: >macro:->\stex_invoke_symbol:n {file://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{smodule}{NotationTest}
\importmodule{Foo}
\notation[foo, prec=500;20x20x20]{bar}{\comp\langle {#1} ^ {#2} _ {#3} \comp\rangle }
\notation[foo, prec=500;20x20x20]{foobar}{\comp\langle #1 \comp\mid [ #2 ] ^ {#3} \comp\rangle }{ {#1}_{\comp
```

Module 23:

---

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

---

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

### Test 13

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

Module 24:  $a + b + c$

# 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> .

---

`\infprec`  
`\neginfprec`

---

Maximal and minimal notation precedences.

---

`\dobrackets`

---

`\dobrackets {⟨body⟩}`

Puts  $\langle body \rangle$  in parentheses; scaled if in display mode unscaled otherwise. Uses the current  $\text{\TeX}$  brackets (by default ( and )), which can be changed temporarily using `\withbrackets`.

---

`\withbrackets`

---

`\withbrackets ⟨left⟩ ⟨right⟩ {⟨body⟩}`

Temporarily (i.e. within  $\langle body \rangle$ ) sets the brackets used by  $\text{\TeX}$  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 `\left` and `\right` in display-mode.

### Test 14

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

Module 25:  $\langle a^b_c \rangle$  and  $\langle a^b_c \rangle$ .

### Test 15

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

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

Module 26:  $\langle a \mid [b;c,d,e,f]^g \rangle$  and  $\langle a \mid [b;c]^g \rangle$  and  $\langle a \mid [b]^c \rangle$

$$a + (b \cdot c) \text{ and } a \cdot \frac{a}{b} + \frac{a}{c}$$

$$a + (b \cdot c) \text{ and } a \cdot \frac{a}{b} + \frac{a}{c}$$

$$a + (b \cdot c) \text{ and } a \cdot \frac{a}{b} + \frac{a}{c}$$

---

`\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{smodule}{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{smodule}
```

**Module 27:**  
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

# STEX-Structural Features

Code related to structural features

### 16.1 Macros and Environments

#### 16.1.1 Structures

`mathstructure` TODO

# Chapter 17

## TeX-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 do, or what we have achieved so far. As such, it cannot be the target of a <b>\premise</b> .

18.2.5 Proof End Markers

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

`\sproofend`

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

`\sProofEndSymbol`

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

18.2.6 Configuration of the Presentation

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

EdN:8

Environment	configuration macro	value
<code>sproof</code>	<code>\spf@proof@kw</code>	Proof
<code>sketchproof</code>	<code>\spf@sketchproof@kw</code>	ProofSketch

Figure 1: Configuration Hooks for Semantic Proof Markup

`\pstlabelstyle`

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

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

Figure 2: Configuration Proof Step Label Styles

18.3 Limitations

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

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

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

## Chapter 19

# sTeX-Metatheory

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

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

### 19.1 Symbols



**Part III**  
**Extensions**

## Chapter 20

# Tikzinput

### 20.1 Macros and Environments

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

## Chapter 21

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

The `document-structure` package is part of the  $\text{\S T E X}$  collection, a version of  $\text{T E X/L A T E X}$  that allows to markup  $\text{T E X/L A T E X}$  documents semantically without leaving the document format, essentially turning  $\text{T E X/L A T E X}$  into a document format for mathematical knowledge management (MKM).

This package supplies an infrastructure for writing OMDOC documents in  $\text{L A T E X}$ . This includes a simple structure sharing mechanism for  $\text{\S T E X}$  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  $\text{\S T E X}$  sources, or after translation.

### 21.1 Introduction

$\text{\S T E X}$  is a version of  $\text{T E X/L A T E X}$  that allows to markup  $\text{T E X/L A T E X}$  documents semantically without leaving the document format, essentially turning  $\text{T E X/L A T E X}$  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  $\text{\S T E X}$  sources, or after translation. Currently, trans-document referencing provided by this package can only be used in the  $\text{\S T E X}$  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{smodule}{foo}
\symdef{bar}{B^a_r}
...
\begin{omgroup}[id=sec.barderv,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  $\text{\TeX}$  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,  $\text{\TeX}$  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[⟨URL⟩]{⟨label⟩}`, which expands to the previously stored content. If the `\STRlabel` macro was in a different file, then we can give a URL `⟨URL⟩` that lets L<sup>A</sup>T<sub>E</sub>XML 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 L<sup>A</sup>T<sub>E</sub>X. 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  $\text{\TeX}$  preamble of the course notes file. `\setSGvar{⟨vname⟩}{⟨text⟩}` to set the global variable `⟨vname⟩` to `⟨text⟩` and `\useSGvar{⟨vname⟩}` 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><br><code>notes</code> | <ul style="list-style-type: none"><li>• The options <code>slides</code> and <code>notes</code> switch between slides mode and notes mode (see Section 22.2.2).</li></ul> |
|---|--|



<code>sectocframes</code>	<ul style="list-style-type: none"> <li>If the option <code>sectocframes</code> is given, then for the <code>omgroups</code>, special frames with the <code>omgroup</code> title (and number) are generated.</li> </ul>
<code>showmeta</code>	<ul style="list-style-type: none"> <li><code>showmeta</code>. If this is set, then the metadata keys are shown (see [Koh20b] for details and customization options).</li> </ul>
<code>frameimages</code> <code>fiboxed</code>	<ul style="list-style-type: none"> <li>If the option <code>frameimages</code> is set, then slide mode also shows the <code>\frameimage</code>-generated frames (see section 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.

### 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.

### 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>

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.



Part IV

# Implementation

## Chapter 25

# STEX -Basics Implementation

### 25.1 The STEXDocument Class

The `stex` document class is pretty straight-forward: It largely extends the `standalone` package and loads the `stex` package, passing all provided options on to the package.

```
1 <*cls>
2
3 %%%%%%%%% basics.dtx %%%%%%%%%
4
5 \RequirePackage{expl3,l3keys2e}
6 \ProvidesExplClass{stex}{2021/08/01}{1.9}{bla}
7 \LoadClass[border=1px,varwidth]{standalone}
8 \setlength\textwidth{15cm}
9
10 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
11 \ProcessOptions
12
13 \RequirePackage{stex}
14 </cls>
```

### 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-compat}
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   lang       .clist_set:N = \c_stex_languages_clist ,
29   mathhub    .tl_set_x:N = \mathhub ,
30   sms        .bool_set:N = \c_stex_persist_mode_bool ,
31   image      .bool_set:N = \c_tikzinput_image_bool ,
32   unknown    .code:n      = {}
33 }
34 \ProcessKeysOptions { stex }

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

```

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

## 25.3 Messages and logging

```

42 <@@=stex_log>

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

\stex_debug:nn A simple macro issuing package messages with subpath.
53 \cs_new_protected:Nn \stex_debug:nn {
54   \clist_if_in:NnTF \c_stex_debug_clist { all } {
55     \exp_args:Nnnx\msg_set:nnn{stex}{debug / #1}{
56       \\Debug~#1:~#2\\
57     }
58     \msg_none:nn{stex}{debug / #1}
59   }{
60     \clist_if_in:NnT \c_stex_debug_clist { #1 } {
61       \exp_args:Nnnx\msg_set:nnn{stex}{debug / #1}{
62         \\Debug~#1:~#2\\
63       }
64       \msg_none:nn{stex}{debug / #1}
65     }
66   }
67 }

```

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

Redirecting messages:

```

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

```

## 25.4 Persistence

77 `<@=stex_persist>`

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

```

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

```

(End definition for `\c__stex_persist_sms_iow`.)

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

```

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

```

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

## 25.5 HTML Annotations

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

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

```

98 \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
99 \ifcsname if@latexml\endcsname\else

```



```

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

```

(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`

```

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

```

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

`\_stex_annotate_checkempty:n`

```

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

```

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

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

```

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

```

(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

```

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

```

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

`\stex_annotate:env`

`\stex_annotate_invisible:n`

`\stex_annotate_invisible:nnn`

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

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

```

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

```

```

186   }{
187     \par\rustex_annotate_HTML_end:
188   }
189 }{
190   \latexml_if:TF {
191     \cs_new_protected:Nn \stex_annotate:nnn {
192       \__stex_annotate_checkempty:n { #3 }
193       \mode_if_math:TF {
194         \cs:w latexml@annotate@math\cs_end:{#1}{#2}{
195           \tl_use:N \l__stex_annotate_arg_tl
196         }
197       }{
198         \cs:w latexml@annotate@text\cs_end:{#1}{#2}{
199           \tl_use:N \l__stex_annotate_arg_tl
200         }
201       }
202     }
203     \cs_new_protected:Nn \stex_annotate_invisible:n {
204       \__stex_annotate_checkempty:n { #1 }
205       \mode_if_math:TF {
206         \cs:w latexml@invisible@math\cs_end:{
207           \tl_use:N \l__stex_annotate_arg_tl
208         }
209       } {
210         \cs:w latexml@invisible@text\cs_end:{
211           \tl_use:N \l__stex_annotate_arg_tl
212         }
213       }
214     }
215     \cs_new_protected:Nn \stex_annotate_invisible:nnn {
216       \__stex_annotate_checkempty:n { #3 }
217       \cs:w latexml@annotate@invisible\cs_end:{#1}{#2}{
218         \tl_use:N \l__stex_annotate_arg_tl
219       }
220     }
221     \NewDocumentEnvironment{stex_annotate_env} { m m } {
222       \par\begin{latexml@annotateenv}{#1}{#2}
223     }{
224       \par\end{latexml@annotateenv}
225     }
226   }{
227     \cs_new_protected:Nn \stex_annotate:nnn {#3}
228     \cs_new_protected:Nn \stex_annotate_invisible:n {}
229     \cs_new_protected:Nn \stex_annotate_invisible:nnn {}
230     \NewDocumentEnvironment{stex_annotate_env} { m m } {}{}
231   }
232 }

```

(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

```

233 <@=stex_language>

```

`\c_stex_languages_prop`  
`\c_stex_language_abbrevs_prop`

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

```

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

```

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

```

## 25.7 Activating/Deactivating Macros

`\stex_deactivate_macro:Nn`

```

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

```

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

`\stex_reactivate_macro:N`

```

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

```

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

`\stex_do_aftergroup:nn`

```

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

```

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

```

300 </package>

```

## Chapter 26

# STEX -MathHub Implementation

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

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

```

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

```

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

```

(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`

```

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

```

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

```

```

368         }{
369         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
370         }
371     }
372     }{
373     \str_if_empty:NF \l_tmpa_tl {
374     \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
375     }
376     }
377 }
378 }
379 \seq_gset_eq:NN #1 \l_tmpa_seq
380 }
381 }

```

(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 $\underline{TF}$`

```

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

```

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

## 26.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

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

```

(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`

```

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

```



```

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

```

409 <@@=stex_files>

```

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

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

`\g_stex_files_stack` keeps track of file changes

```

410 \seq_gclear_new:N\g_stex_files_stack

```

(End definition for `\g_stex_files_stack`.)

`\c_stex_mainfile_seq`  
`\c_stex_mainfile_str`

```

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

```

414 \seq_gclear_new:N\g_stex_currentfile_seq
415 \cs_new_protected:Nn \stex_filestack_push:n {
416   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
417   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
418     \stex_path_from_string:Nn\g_stex_currentfile_seq{
419       \c_stex_pwd_str/#1
420     }
421   }
422   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
423   \exp_args:NNo\seq_gpush:Nn\g_stex_files_stack\g_stex_currentfile_seq
424 }
425 \cs_new_protected:Nn \stex_filestack_pop: {
426   \seq_if_empty:NF\g_stex_files_stack{
427     \seq_gpop:NN\g_stex_files_stack\l_tmpa_seq
428   }
429   \seq_if_empty:NTF\g_stex_files_stack{
430     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
431   }{
432     \seq_get:NN\g_stex_files_stack\l_tmpa_seq
433     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
434   }
435 }
436

```

```

437 \AddToHook{file/before}{
438   \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
439 }
440 \AddToHook{file/after}{
441   \stex_filestack_pop:
442 }

```

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

## 26.4 MathHub Repositories

```

443 <@@=stex_mathhub>

\mathhub
\c_stex_mathhub_seq
\c_stex_mathhub_str
444 \str_if_empty:NTF\mathhub{
445   \stex_kpsewhich:n{-var-value~MATHHUB}
446   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
447
448   \str_if_empty:NTF\c_stex_mathhub_str{
449     \msg_warning:nn{stex}{warning/nomathhub}
450   }{
451     \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
452     \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
453   }
454 }{
455   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
456   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
457     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
458       \c_stex_pwd_str/\mathhub
459     }
460   }
461   \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
462   \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
463 }

```

(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
464 \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
465   \str_set:Nx \l_tmpa_str { #1 }
466   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
467     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
468     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
469     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
470     \__stex_mathhub_find_manifest:N \l_tmpa_seq
471     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
472       \msg_error:nnxx{stex}{error/norepository}{#1}{
473         \stex_path_to_string:N \c_stex_mathhub_str
474       }
475     } {
476       \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
477     }
478   }
479 }

```

(End definition for \\_stex\_mathhub\_do\_manifest:n.)

\l\_stex\_mathhub\_manifest\_file\_seq

480 \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:

```

481 \cs_new_protected:Nn \_stex_mathhub_find_manifest:N {
482   \seq_set_eq:NN\l_tmpa_seq #1
483   \bool_set_true:N\l_tmpa_bool
484   \bool_while_do:Nn \l_tmpa_bool {
485     \seq_if_empty:NTF \l_tmpa_seq {
486       \bool_set_false:N\l_tmpa_bool
487     }{
488       \file_if_exist:nTF{
489         \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
490       }{
491         \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
492         \bool_set_false:N\l_tmpa_bool
493       }{
494         \file_if_exist:nTF{
495           \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
496         }{
497           \seq_put_right:Nn\l_tmpa_seq{META-INF}
498           \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
499           \bool_set_false:N\l_tmpa_bool
500         }{
501           \file_if_exist:nTF{
502             \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
503           }{
504             \seq_put_right:Nn\l_tmpa_seq{meta-inf}
505             \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
506             \bool_set_false:N\l_tmpa_bool
507           }{
508             \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
509           }
510         }
511       }
512     }
513   }
514   \seq_set_eq:NN\l_stex_mathhub_manifest_file_seq\l_tmpa_seq
515 }

```

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

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

516 \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:

```

517 \cs_new_protected:Nn \_stex_mathhub_parse_manifest:n {
518   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
519   \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
520   \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
521     \str_set:Nn \l_tmpa_str {##1}
522     \exp_args:NNoo \seq_set_split:Nnn
523       \l_tmpb_seq \c_colon_str \l_tmpa_str
524     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
525       \exp_args:NNe \str_set:Nn \l_tmpb_tl {
526         \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
527       }
528       \exp_args:No \str_case:nnTF \l_tmpa_tl {
529         {id} {
530           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
531             { id } \l_tmpb_tl
532         }
533         {narration-base} {
534           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
535             { narr } \l_tmpb_tl
536         }
537         {url-base} {
538           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
539             { docurl } \l_tmpb_tl
540         }
541         {source-base} {
542           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
543             { ns } \l_tmpb_tl
544         }
545         {ns} {
546           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
547             { ns } \l_tmpb_tl
548         }
549         {dependencies} {
550           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
551             { deps } \l_tmpb_tl
552         }
553       }{}{}
554     }{}
555   }
556   \ior_close:N \c__stex_mathhub_manifest_ior
557 }

```

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

`\stex_set_current_repository:n`

```

558 \cs_new_protected:Nn \stex_set_current_repository:n {
559   \stex_require_repository:n { #1 }
560   \prop_set_eq:Nc \l_stex_current_repository_prop {
561     c_stex_mathhub_#1_manifest_prop
562   }
563 }

```

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

`\stex_require_repository:n`

```

564 \cs_new_protected:Nn \stex_require_repository:n {
565   \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
566     \stex_debug:nn{mathhub}{Opening~archive:~#1}
567     \__stex_mathhub_do_manifest:n { #1 }
568     \exp_args:Nx \stex_add_to_sms:n {
569       \prop_const_from_keyval:cn { c_stex_mathhub_#1_manifest_prop } {
570         id   = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { id } ,
571         ns   = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { ns } ,
572         narr = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { narr } ,
573         deps = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { deps }
574       }
575     }
576   }
577 }

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

578 %\prop_new:N \l_stex_current_repository_prop
579
580 \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
581 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
582   \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
583 } {
584   \__stex_mathhub_parse_manifest:n { main }
585   \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
586   \l_tmpa_str
587   \prop_set_eq:cN { c_stex_mathhub_#1_manifest_prop }
588   \c_stex_mathhub_main_manifest_prop
589   \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
590   \stex_debug:nn{mathhub}{Current~repository:~
591     \prop_item:Nn \l_stex_current_repository_prop {id}
592   }
593 }

```

(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.

```

594 \cs_new_protected:Nn \stex_in_repository:nn {
595   \str_set:Nx \l_tmpa_str { #1 }
596   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
597   \str_if_empty:NTF \l_tmpa_str {
598     \prop_if_exist:NTF \l_stex_current_repository_prop {
599       \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
600       \exp_args:Ne \l_tmpa_cs{
601         \prop_item:Nn \l_stex_current_repository_prop { id }
602       }
603     }{
604       \l_tmpa_cs{}
605     }
606   }{
607     \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}

```

```

608 \stex_require_repository:n \l_tmpa_str
609 \str_set:Nx \l_tmpa_str { #1 }
610 \exp_args:Nne \use:nn {
611   \stex_set_current_repository:n \l_tmpa_str
612   \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
613 }{
614   \stex_debug:nn{mathhub}{switching~back~to:~
615     \prop_if_exist:NTF \l_stex_current_repository_prop {
616       \prop_item:Nn \l_stex_current_repository_prop { id }::~
617       \meaning\l_stex_current_repository_prop
618     }{
619       no~repository
620     }
621   }
622   \prop_if_exist:NTF \l_stex_current_repository_prop {
623     \stex_set_current_repository:n {
624       \prop_item:Nn \l_stex_current_repository_prop { id }
625     }
626   }{
627     \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
628   }
629 }
630 }
631 }

```

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

```

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

```

```

658     \stex_annotate_invisible:nnn{inputref}{
659       \l_tmpa_str / #2
660     }{}
661   }{
662     \begingroup
663       \inputreftrue
664       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
665     \endgroup
666   }
667 }
668 }
669
670 \NewDocumentCommand \inputref { 0{ } m }{
671   \stex_inputref:nn{ #1 }{ #2 }
672 }
673
674 \cs_new_protected:Nn \stex_mhbibresource:nn {
675   \stex_in_repository:nn {#1} {
676     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
677   }
678 }
679 \newcommand\addmhbibresource[2][]{
680   \stex_mhbibresource:nn{ #1 }{ #2 }
681 }

```

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

### `\mhpath`

```

682 \def \mhpath #1 #2 {
683   \exp_args:Ne \str_if_eq:nnTF{#1}{#2}{
684     \c_stex_mathhub_str /
685     \prop_item:Nn \l_stex_current_repository_prop { id }
686     / source / #2
687   }{
688     \c_stex_mathhub_str / #1 / source / #2
689   }
690 }

```

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

### `\libinput`

```

691 \cs_new_protected:Npn \libinput #1 {
692   \prop_if_exist:NF \l_stex_current_repository_prop {
693     \msg_error:nnn{stex}{error/notinarchive}\libinput
694   }
695   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
696     \msg_error:nnn{stex}{error/notinarchive}\libinput
697   }
698   \bool_set_false:N \l_tmpa_bool
699   \tl_clear:N \l_tmpa_tl
700   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
701   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
702   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str
703   \seq_pop_left:NNT \l_tmpb_seq \l_tmpb_str {

```

```

704 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
705 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
706 / meta-inf / lib / #1.tex}{
707 \bool_set_true:N \l_tmpa_bool
708 \tl_put_right:Nx \l_tmpa_tl {
709 \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
710 / meta-inf / lib / #1.tex}
711 }
712 }{}
713 }
714 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
715 / \l_tmpa_str / lib / #1.tex
716 }{
717 \bool_set_true:N \l_tmpa_bool
718 \tl_put_right:Nx \l_tmpa_tl {
719 \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
720 / \l_tmpa_str / lib / #1.tex}
721 }
722 }{}
723 \bool_if:NF \l_tmpa_bool {
724 \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
725 }
726 \l_tmpa_tl
727 }

```

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

`\libusepackage`

```

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

```



```

754 \bool_if:NT \l_libusepackage_bool {
755   \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
756 }
757 \bool_set_true:N \l_libusepackage_bool
758 \tl_put_right:Nx \l_tmpa_tl {
759   \exp_not:N \usepackage[#1] { \stex_path_to_string:N \l_tmpa_seq
760     / \l_tmpa_str / lib / #2}
761 }
762 }{}
763 \bool_if:NF \l_libusepackage_bool {
764   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
765 }
766 \l_tmpa_tl
767 }

```

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

```

768
769 \AddToHook{begindocument}{
770 \ltx@ifpackageloaded{graphicx}{
771   \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
772   \newcommand\mhgraphics[2][]{%
773     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
774     \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}}
775   \newcommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
776 }{}
777 \ltx@ifpackageloaded{listings}{
778   \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
779   \newcommand\lstinputmhlstlisting[2][]{%
780     \def\lst@mhrepos{}\setkeys{lst}{#1}%
781     \lstinputlisting[#1]{\mhp\lst@mhrepos{#2}}}
782   \newcommand\clstinputmhlstlisting[2][]{\begin{center}\lstinputmhlstlisting[#1]{#2}\end{center}}
783 }{}
784 }
785
786
787 \end{package}

```

## Chapter 27

# STEX -References Implementation

```
788 <*package>
789
790 %%%%%%%%%% references.dtx %%%%%%%%%%
791
792 %\RequirePackage{hyperref}
793 %\RequirePackage{cleveref}
794 <@@=stex_refs>
795
796 Warnings and error messages
797
798 \iow_new:N \c__stex_refs_refs_iow
799 \AddToHook{begindocument}{
800   \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
801 }
802 \AddToHook{enddocument}{
803   \iow_close:N \c__stex_refs_refs_iow
804 }
805
806 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
807
808 \NewDocumentCommand \STEXreftitle { m } {
809   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
810 }
811
```

### 27.1 Document URIs and URLs

```
809 \seq_new:N \g__stex_refs_all_refs_seq
810
811 \str_new:N \l_stex_current_docns_str
812
813 \cs_new_protected:Nn \stex_get_document_uri: {
814   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
815   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
816   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
817   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
818 }
819
```

```

818 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
819
820 \str_clear:N \l_tmpa_str
821 \prop_if_exist:NT \l_stex_current_repository_prop {
822   \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
823     \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
824   }
825 }
826
827 \str_if_empty:NTF \l_tmpa_str {
828   \str_set:Nx \l_stex_current_docns_str {
829     file:/\stex_path_to_string:N \l_tmpa_seq
830   }
831 }{
832   \bool_set_true:N \l_tmpa_bool
833   \bool_while_do:Nn \l_tmpa_bool {
834     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
835     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
836       {source} { \bool_set_false:N \l_tmpa_bool }
837     }{}{
838       \seq_if_empty:NT \l_tmpa_seq {
839         \bool_set_false:N \l_tmpa_bool
840       }
841     }
842   }
843
844   \seq_if_empty:NTF \l_tmpa_seq {
845     \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
846   }{
847     \str_set:Nx \l_stex_current_docns_str {
848       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
849     }
850   }
851 }
852 }
853
854 \str_new:N \l_stex_current_docurl_str
855 \cs_new_protected:Nn \stex_get_document_url: {
856   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
857   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
858   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
859   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
860   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
861
862   \str_clear:N \l_tmpa_str
863   \prop_if_exist:NT \l_stex_current_repository_prop {
864     \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
865       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
866         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
867       }
868     }
869   }
870
871   \str_if_empty:NTF \l_tmpa_str {
872     \str_set:Nx \l_stex_current_docurl_str {

```

```

872     file:/\stex_path_to_string:N \l_tmpa_seq
873   }
874 }{
875   \bool_set_true:N \l_tmpa_bool
876   \bool_while_do:Nn \l_tmpa_bool {
877     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
878     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
879       {source} { \bool_set_false:N \l_tmpa_bool }
880     }{}{
881       \seq_if_empty:NT \l_tmpa_seq {
882         \bool_set_false:N \l_tmpa_bool
883       }
884     }
885   }
886
887   \seq_if_empty:NTF \l_tmpa_seq {
888     \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
889   }{
890     \str_set:Nx \l_stex_current_docurl_str {
891       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
892     }
893   }
894 }
895 }

```

## 27.2 Setting Reference Targets

```

896 \str_const:Nn \c__stex_refs_url_str{URL}
897 \str_const:Nn \c__stex_refs_ref_str{REF}
898 % @currentlabel -> number
899 % @currentlabelname -> title
900 % @currentHref -> name.number <- id of some kind
901 % \theH# -> \arabic{section}
902 % \the# -> number
903 % \hyper@makecurrent{#}
904 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
905   \stex_get_document_uri:
906   \str_set:Nx \l_tmpa_str { #1 }
907   \str_if_empty:NF {
908     \str_set:Nx \l_tmpa_str {
909       \l_stex_current_docns_str??\l_tmpa_str
910     }
911     \seq_if_in:NoF \g__stex_refs_all_refs_seq \l_tmpa_str {
912       \seq_gput_right:No \g__stex_refs_all_refs_seq \l_tmpa_str
913     }
914     \stex_if_smsmode:TF {
915       \stex_get_document_url:
916       \str_gset_eq:cN {sref_url_\l_tmpa_str_str}\l_stex_current_docurl_str
917       \str_gset_eq:cN {sref_\l_tmpa_str_type}\c__stex_refs_url_str
918     }{
919       \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter\unexpanded\expandafter
920         \exp_args:Nx\label{sref_\l_tmpa_str}
921         \exp_args:NNNx\immediate\write\@auxout{\stexauxadddocref{\l_tmpa_str}}
922         \str_gset:cx {sref_\l_tmpa_str_type}\c__stex_refs_ref_str

```

```

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

```

## 27.3 Using References

```

947 \str_new:N \l__stex_refs_indocument_str
948 \keys_define:nn { stex / sref } {
949   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
950   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
951   pre           .tl_set:N = \l__stex_refs_pre_tl ,
952   post          .tl_set:N = \l__stex_refs_post_tl ,
953   %indoc        .str_set_x:N = \l__stex_refs_repo_str ,
954 }
955
956 \bool_new:N \c__stex_refs_hyperref_bool
957 \bool_set_false:N \c__stex_refs_hyperref_bool
958 \AddToHook{begindocument}{
959   \@ifpackageloaded{hyperref}{
960     \bool_set_true:N \c__stex_refs_hyperref_bool
961   }{}
962 }
963
964
965 \cs_new_protected:Nn \__stex_refs_args:n {
966   \tl_clear:N \l__stex_refs_linktext_tl
967   \tl_clear:N \l__stex_refs_fallback_tl
968   \tl_clear:N \l__stex_refs_pre_tl
969   \tl_clear:N \l__stex_refs_post_tl
970   \str_clear:N \l__stex_refs_repo_str
971   \keys_set:nn { stex / sref } { #1 }
972 }
973

```

```

974 \NewDocumentCommand \sref { 0{} m}{
975   \__stex_refs_args:n { #1 }
976   \str_if_empty:NTF \l__stex_refs_indocument_str {
977     \str_set:Nn \l_tmpa_str { #2 }
978     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
979     \tl_set:Nn \l_tmpa_tl {
980       \l__stex_refs_fallback_tl
981     }
982     \seq_map_inline:Nn \g__stex_refs_all_refs_seq {
983       \str_set:Nn \l_tmpb_str { ##1 }
984       \str_if_eq:eeT { \l_tmpa_str } {
985         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int }{-1 }
986       } {
987         \seq_map_break:n {
988           \tl_set:Nn \l_tmpa_tl {
989             % doc uri in \l_tmpb_str
990             \str_set:Nx \l_tmpa_str {\use:c{sref_\l_tmpb_str _type}}
991             \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
992               % reference
993               \cs_if_exist:cTF{autoref}{
994                 \l__stex_refs_pre_tl\autoref{sref_\l_tmpb_str}\l__stex_refs_post_tl
995               }{
996                 \l__stex_refs_pre_tl\ref{sref_\l_tmpb_str}\l__stex_refs_post_tl
997               }
998             }{
999               % URL
1000               \if_bool:N \c__stex_refs_hyperref_bool {
1001                 \exp_args:Nx \href{\use:c{sref_url_\l_tmpb_str _str}}{\l__stex_refs_fallback
1002               }{
1003                 \l__stex_refs_fallback_tl
1004               }
1005             }
1006           }
1007         }
1008       }
1009     }
1010     \l_tmpa_tl
1011   }{
1012     % TODO
1013   }
1014 }
1015
1016 \NewDocumentCommand \srefsym { 0{} m}{
1017   \stex_get_symbol:n { #2 }
1018   \__stex_refs_args:n { #1 }
1019   \str_if_empty:NTF \l__stex_refs_indocument_str {
1020     \tl_set:Nn \l_tmpa_tl {
1021       \l__stex_refs_fallback_tl
1022     }
1023     \tl_if_exist:cT{sref_sym_\l_stex_get_symbol_uri_str _type}{
1024       \tl_set:Nn \l_tmpa_tl {
1025         % doc uri in \l_tmpb_str
1026         \str_set:Nx \l_tmpa_str {\use:c{sref_sym_\l_stex_get_symbol_uri_str _type}}
1027         \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {

```

```

1028         % reference
1029         \cs_if_exist:cTF{autoref}{
1030             \l__stex_refs_pre_tl\autoref{sref_sym_\l_stex_get_symbol_uri_str}\l__stex_refs_p
1031         }{
1032             \l__stex_refs_pre_tl\ref{sref_sym_\l_stex_get_symbol_uri_str}\l__stex_refs_post_
1033         }
1034     }{
1035         % URL
1036         \if_bool:N \c__stex_refs_hyperref_bool {
1037             \exp_args:Nx \href{\use:c{sref_sym_url_\l_stex_get_symbol_uri_str _str}}{\l__ste
1038         }{
1039             \l__stex_refs_fallback_tl
1040         }
1041     }
1042 }
1043 }
1044 \l_tmpa_tl
1045 }{
1046     % TODO
1047 }
1048 }
1049
1050 \cs_new_protected:Npn \srefsymuri #1 #2 {
1051     \hyperref[sref_sym_#1]{#2}
1052 }
1053
1054 </package>

```

## Chapter 28

# STEX -Modules Implementation

```
1055 <*package>
1056
1057 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1058
1059 <@@=stex_modules>
1060
1061   Warnings and error messages
1062   \msg_new:nnn{stex}{error/unknownmodule}{
1063     No~module~#1~found
1064   }
1065   \msg_new:nnn{stex}{error/syntax}{
1066     Syntax~error:~#1
1067   }
1068   \msg_new:nnn{stex}{error/siglanguage}{
1069     Module~#1~declares~signature~#2,~but~does~not~
1070     declare~its~language
1071   }
1072   \msg_new:nnn{stex}{error/conflictingmodules}{
1073     Conflicting~imports~for~module~#1
1074   }
1075
1076 \l_stex_current_module_str The current module:
1077 \str_new:N \l_stex_current_module_str
1078
1079 (End definition for \l_stex_current_module_str. This variable is documented on page 26.)
1080
1081 \l_stex_all_modules_seq Stores all available modules
1082 \seq_new:N \l_stex_all_modules_seq
1083
1084 (End definition for \l_stex_all_modules_seq. This variable is documented on page 26.)
1085
1086 \stex_if_in_module_p:
1087 \stex_if_in_module:TF
1088 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1089   \str_if_empty:NTF \l_stex_current_module_str
1090     \prg_return_false: \prg_return_true:
1091 }
```



(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`

```
1080 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1081   \prop_if_exist:cTF { c_stex_module_#1_prop }
1082   \prg_return_true: \prg_return_false:
1083 }
```

(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:

```
1084 \cs_new_protected:Nn \stex_add_to_current_module:n {
1085   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1086 }
1087 \cs_new_protected:Npn \STEXexport {
1088   \beginngroup
1089   \newlinechar=-1\relax
1090   \endlinechar=-1\relax
1091   %\catcode'\ = 9\relax
1092   \expandafter\endgroup\STEXexport:n
1093 }
1094 \cs_new_protected:Nn \STEXexport:n {
1095   \ignorespaces #1
1096   \stex_add_to_current_module:n { \ignorespaces #1 }
1097   \stex_smsmode_do:
1098 }
1099 \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`

```
1100 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1101   \str_set:Nx \l_tmpa_str { #1 }
1102   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1103 }
1104
1105 %\cs_new_protected:Nn \stex_add_field_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 _fields} { \l_tmpa_str }
1108 %}
```

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

`\stex_collect_imports:n`

```
1109 \cs_new_protected:Nn \stex_collect_imports:n {
1110   \seq_clear:N \l_stex_collect_imports_seq
1111   \__stex_modules_collect_imports:n {#1}
1112 }
1113 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1114   \seq_map_inline:cn {c_stex_module_#1_imports} {
1115     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1116       \__stex_modules_collect_imports:n { ##1 }
1117     }
1118 }
```

```

1118 }
1119 \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1120   \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1121 }
1122 }

```

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

`\stex_add_import_to_current_module:n`

```

1123 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1124   \str_set:Nx \l_tmpa_str { #1 }
1125   \exp_args:Nno
1126   \seq_if_in:cnF{c_stex_module\_l_stex_current_module_str_imports}\l_tmpa_str{
1127     \seq_gput_right:co{c_stex_module\_l_stex_current_module_str_imports}\l_tmpa_str
1128   }
1129 }

```

(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).

```

1130 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
1131   \str_set:Nx \l_tmpa_str { #1 }
1132   \seq_set_eq:NN \l_tmpa_seq #2
1133   % split off file extension
1134   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1135   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1136   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1137   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1138
1139   \bool_set_true:N \l_tmpa_bool
1140   \bool_while_do:Nn \l_tmpa_bool {
1141     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1142     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1143       {source} { \bool_set_false:N \l_tmpa_bool }
1144     }{}{
1145       \seq_if_empty:NT \l_tmpa_seq {
1146         \bool_set_false:N \l_tmpa_bool
1147       }
1148     }
1149   }
1150
1151   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
1152   \str_if_empty:NTF \l_stex_modules_subpath_str {
1153     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
1154   }{
1155     \str_set:Nx \l_stex_modules_ns_str {
1156       \l_tmpa_str/\l_stex_modules_subpath_str
1157     }
1158   }
1159 }

```

(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
1160 \str_new:N \l_stex_modules_ns_str
1161 \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.

```

1162 \cs_new_protected:Nn \stex_modules_current_namespace: {
1163   \str_clear:N \l_stex_modules_subpath_str
1164   \prop_if_exist:NTF \l_stex_current_repository_prop {
1165     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1166     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1167   }{
1168     % split off file extension
1169     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1170     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1171     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1172     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1173     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1174     \str_set:Nx \l_stex_modules_ns_str {
1175       file:/\stex_path_to_string:N \l_tmpa_seq
1176     }
1177   }
1178 }

```

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

## 28.1 The module environment

module arguments:

```

1179 \keys_define:nn { stex / module } {
1180   title      .tl_set:N      = \smodulename ,
1181   type       .str_set_x:N   = \smodulename ,
1182   id         .str_set_x:N   = \smoduleid ,
1183   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1184   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1185   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1186   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1187   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1188   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1189   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1190 }
1191
1192 \cs_new_protected:Nn \__stex_modules_args:n {
1193   \str_clear:N \smodulename
1194   \str_clear:N \smodulename
1195   \str_clear:N \smoduleid
1196   \str_clear:N \l_stex_module_ns_str
1197   \str_clear:N \l_stex_module_lang_str
1198   \str_clear:N \l_stex_module_sig_str
1199   \str_clear:N \l_stex_module_creators_str

```

```

1200 \str_clear:N \l_stex_module_contributors_str
1201 \str_clear:N \l_stex_module_meta_str
1202 \str_clear:N \l_stex_module_srccite_str
1203 \keys_set:nn { stex / module } { #1 }
1204 }
1205
1206 % module parameters here? In the body?
1207

```

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

```

1208 \cs_new_protected:Nn \stex_module_setup:nn {
1209   \str_set:Nx \l_stex_module_name_str { #2 }
1210   \__stex_modules_args:n { #1 }

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

1211   \stex_if_in_module:TF {
1212     % Nested module
1213     \prop_get:cnN {c_stex_module\_l_stex_current_module_str _prop}
1214       { ns } \l_stex_module_ns_str
1215     \str_set:Nx \l_stex_module_name_str {
1216       \prop_item:cn {c_stex_module\_l_stex_current_module_str _prop}
1217         { name } / \l_stex_module_name_str
1218     }
1219   }{
1220     % not nested:
1221     \str_if_empty:NT \l_stex_module_ns_str {
1222       \stex_modules_current_namespace:
1223       \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1224       \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1225         / { \l_stex_module_ns_str }
1226       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1227       \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1228         \str_set:Nx \l_stex_module_ns_str {
1229           \stex_path_to_string:N \l_tmpa_seq
1230         }
1231       }
1232     }
1233   }

```

Next, we determine the language of the module:

```

1234 \str_if_empty:NT \l_stex_module_lang_str {
1235   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1236   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1237   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1238   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1239   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1240     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1241       inferred~from~file~name}
1242     \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1243   }
1244 }
1245
1246 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {

```

```

1247 \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1248 \l_tmpa_str {
1249   \ltx@ifpackageloaded{babel}{
1250     \exp_args:Nx \selectlanguage { \l_tmpa_str }
1251   }{}
1252 } {
1253   \msg_error:nxx{stex}{error/unknownlanguage}{\l_tmpa_str}
1254 }
1255 }}

```

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

```

1256 \str_if_empty:NTF \l_stex_module_sig_str {
1257   \exp_args:Nnx \prop_gset_from_keyval:cn {
1258     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1259   } {
1260     name      = \l_stex_module_name_str ,
1261     ns        = \l_stex_module_ns_str ,
1262     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1263     lang      = \l_stex_module_lang_str ,
1264     sig       = \l_stex_module_sig_str ,
1265     meta      = \l_stex_module_meta_str
1266   }
1267   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1268   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _fields}
1269   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1270   \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1271   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1272 \str_if_empty:NT \l_stex_module_meta_str {
1273   \str_set:Nx \l_stex_module_meta_str {
1274     \c_stex_metatheory_ns_str ? Metatheory
1275   }
1276 }
1277 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1278   \bool_set_true:N \l_stex_in_meta_bool
1279   \exp_args:Nx \stex_add_to_current_module:n {
1280     \bool_set_true:N \l_stex_in_meta_bool
1281     \stex_activate_module:n {\l_stex_module_meta_str}
1282     \bool_set_false:N \l_stex_in_meta_bool
1283   }
1284   \stex_activate_module:n {\l_stex_module_meta_str}
1285   \bool_set_false:N \l_stex_in_meta_bool
1286 }
1287 }{
1288   \str_if_empty:NT \l_stex_module_lang_str {
1289     \msg_error:nxxx{stex}{error/siglanguage}{
1290       \l_stex_module_ns_str?\l_stex_module_name_str
1291     }\l_stex_module_sig_str}
1292   }
1293
1294   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1295   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str

```

```

1296 \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1297 \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1298 \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1299 \str_set:Nx \l_tmpa_str {
1300   \stex_path_to_string:N \l_tmpa_seq /
1301   \l_tmpa_str . \l_stex_module_sig_str .tex
1302 }
1303 \IfFileExists \l_tmpa_str {
1304   \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1305     \str_clear:N \l_stex_current_module_str
1306     \seq_clear:N \l_stex_all_modules_seq
1307     \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1308   }
1309 }{
1310   \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1311 }
1312 \stex_if_smsmode:F {
1313   \stex_activate_module:n {
1314     \l_stex_module_ns_str ? \l_stex_module_name_str
1315   }
1316 }
1317 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1318 }
1319 }

```

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

**module** The module environment.

```

\__stex_modules_begin_module: implements \begin{smodule}

1320 \int_new:N \l_stex_module_group_depth_int
1321 \cs_new_protected:Nn \__stex_modules_begin_module: {
1322   \stex_reactivate_macro:N \STEXexport
1323   \stex_reactivate_macro:N \importmodule
1324   \stex_reactivate_macro:N \symdecl
1325   \stex_reactivate_macro:N \notation
1326   \stex_reactivate_macro:N \symdef
1327
1328   \stex_debug:nn{modules}{
1329     New~module:\\
1330     Namespace:~\l_stex_module_ns_str\\
1331     Name:~\l_stex_module_name_str\\
1332     Language:~\l_stex_module_lang_str\\
1333     Signature:~\l_stex_module_sig_str\\
1334     Metatheory:~\l_stex_module_meta_str\\
1335     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1336   }
1337
1338   \seq_put_right:Nx \l_stex_all_modules_seq {
1339     \l_stex_module_ns_str ? \l_stex_module_name_str
1340   }
1341
1342   % \seq_gput_right:Nx \g_stex_modules_in_file_seq
1343   % { \l_stex_module_ns_str ? \l_stex_module_name_str }

```

```

1344
1345
1346 \stex_if_smsmode:F{
1347   \begin{stex_annotate_env} {theory} {
1348     \l_stex_module_ns_str ? \l_stex_module_name_str
1349   }
1350
1351   \stex_annotate_invisible:nnn{header}{} {
1352     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1353     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1354     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1355       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1356     }
1357     \str_if_empty:NF \smoduletype {
1358       \stex_annotate:nnn{type}{\smoduletype}{}
1359     }
1360   }
1361 }
1362 \int_set:Nn \l_stex_module_group_depth_int {\currentgrouplevel}
1363 % TODO: Inherit metatheory for nested modules?
1364 }
1365 \iffalse \end{stex_annotate_env} \fi %^^A make syntax highlighting work again

```

(End definition for \\_stex\_modules\_begin\_module:.)

```

\_stex_modules_end_module: implements \end{module}

1366 \cs_new_protected:Nn \_stex_modules_end_module: {
1367   % \str_set:Nx \l_tmpa_str {
1368   %   c_stex_module_
1369   %   \prop_item:Nn \l_stex_current_module_prop { ns } ?
1370   %   \prop_item:Nn \l_stex_current_module_prop { name }
1371   %   _prop
1372   % }
1373   %^^A \prop_new:c { \l_tmpa_str }
1374   % \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
1375   \stex_debug:nn{modules}{Closing module~\prop_item:cn {c_stex_module_\l_stex_current_module_
1376   }

```

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

**smodule** The core environment, with no header

```

1377 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1378 \NewDocumentEnvironment { smodule } { 0 } { m } {
1379   \stex_module_setup:nn{#1}{#2}
1380   \par
1381   \stex_if_smsmode:F{
1382     \tl_clear:N \l_tmpa_tl
1383     \clist_map_inline:Nn \smoduletype {
1384       \tl_if_exist:cT {\_stex_modules_smodule_##1_start:}{
1385         \tl_set:Nn \l_tmpa_tl {\use:c{\_stex_modules_smodule_##1_start:}}
1386       }
1387     }
1388     \tl_if_empty:NTF \l_tmpa_tl {
1389       \_stex_modules_smodule_start:

```

```

1390     }{
1391         \l_tmpa_tl
1392     }
1393 }
1394 \__stex_modules_begin_module:
1395 \stex_ref_new_doc_target:n \smoduleid
1396 \stex_smsmode_do:
1397 } {
1398 \__stex_modules_end_module:
1399 \stex_if_smsmode:TF {
1400 %     \exp_args:Nx \stex_add_to_sms:n {
1401 %         \prop_gset_from_keyval:cn {
1402 %             c_stex_module_
1403 %             \prop_item:Nn \l_stex_current_module_prop { ns } ?
1404 %             \prop_item:Nn \l_stex_current_module_prop { name }
1405 %             _prop
1406 %         } {
1407 %             name      = \prop_item:cn { \l_tmpa_str } { name } ,
1408 %             ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
1409 %             file      = \prop_item:cn { \l_tmpa_str } { file } ,
1410 %             lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
1411 %             sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
1412 %             meta      = \prop_item:cn { \l_tmpa_str } { meta }
1413 %         }
1414 %     }
1415 }{
1416     \end{stex_annotate_env}
1417     \clist_set:No \l_tmpa_clist \smoduletype
1418     \tl_clear:N \l_tmpa_tl
1419     \clist_map_inline:Nn \l_tmpa_clist {
1420         \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1421             \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1422         }
1423     }
1424     \tl_if_empty:NTF \l_tmpa_tl {
1425         \__stex_modules_smodule_end:
1426     }{
1427         \l_tmpa_tl
1428     }
1429 }
1430 }
1431
1432 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1433 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1434
1435 \newcommand\stexpatchmodule[3] [] {
1436     \str_set:Nx \l_tmpa_str{ #1 }
1437     \str_if_empty:NTF \l_tmpa_str {
1438         \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1439         \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1440     }{
1441         \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1442         \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1443     }

```



```
1444 }
1445
```

## 28.2 Invoking modules

```
\STEXModule
\stex_invoke_module:n
```

```
1446 \NewDocumentCommand \STEXModule { m } {
1447   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1448   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1449   \tl_set:Nn \l_tmpa_tl {
1450     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1451   }
1452   \seq_map_inline:Nn \l_stex_all_modules_seq {
1453     \str_set:Nn \l_tmpb_str { ##1 }
1454     \str_if_eq:eeT { \l_tmpa_str } {
1455       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1456     } {
1457       \seq_map_break:n {
1458         \tl_set:Nn \l_tmpa_tl {
1459           \stex_invoke_module:n { ##1 }
1460         }
1461       }
1462     }
1463   }
1464   \l_tmpa_tl
1465 }
1466
1467 \cs_new_protected:Nn \stex_invoke_module:n {
1468   \stex_debug:nn{modules}{Invoking~module~#1}
1469   \peek_charcode_remove:NTF ! {
1470     \__stex_modules_invoke_uri:nN { #1 }
1471   } {
1472     \peek_charcode_remove:NTF ? {
1473       \__stex_modules_invoke_symbol:nn { #1 }
1474     } {
1475       \msg_error:nnx{stex}{error/syntax}{
1476         ?~or~!~expected~after~
1477         \c_backslash_str STEXModule{#1}
1478       }
1479     }
1480   }
1481 }
1482
1483 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1484   \str_set:Nn #2 { #1 }
1485 }
1486
1487 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1488   \stex_invoke_symbol:n{#1?#2}
1489 }
```

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

`\stex_activate_module:n`

```
1490 \bool_new:N \l_stex_in_meta_bool
1491 \bool_set_false:N \l_stex_in_meta_bool
1492 \cs_new_protected:Nn \stex_activate_module:n {
1493   \stex_debug:nn{modules}{Activating~module~#1}
1494   \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1495     \msg_error:nnn{stex}{error/conflictingmodules}{ #1 }
1496   }
1497   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1498     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1499     \use:c{ c_stex_module_#1_code }
1500   }
1501 }
```

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

```
1502 </package>
```

## Chapter 29

# STEX -Module Inheritance Implementation

```
1503 <*package>
1504
1505 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1506
```

### 29.1 SMS Mode

```
1507 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1508 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1509 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1510 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1511
1512 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1513   \makeatletter
1514   \makeatother
1515   \ExplSyntaxOn
1516   \ExplSyntaxOff
1517   \rustexBREAK
1518 }
1519
1520 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1521   \symdef
1522   \importmodule
1523   \notation
1524   \symdecl
1525   \STEXexport
1526   \inlineass
1527   \inlinedef
1528   \inlineex
1529   \endinput
1530   \setnotation
```

```

1531 \copynotation
1532 }
1533
1534 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1535   \tl_to_str:n {
1536     smodule,
1537     copymodule,
1538     interpretmodule
1539     sdefinition,
1540     sexample,
1541     sassertion,
1542     sparagraph
1543   }
1544 }

```

(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`

```

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

```

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

`\stex_in_smsmode:nn`

```

1550 \cs_new_protected:Nn \stex_in_smsmode:nn {
1551   \vbox_set:Nn \l_tmpa_box {
1552     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1553     \bool_gset_true:N \g__stex_smsmode_bool
1554     #2
1555     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1556   }
1557   \box_clear:N \l_tmpa_box
1558 }
1559
1560 \quark_new:N \q__stex_smsmode_break
1561
1562 %\ior_new:N \c__stex_smsmode_ior
1563 %\tl_new:N \l__stex_smsmode_filecontent_tl
1564 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1565   % \tl_clear:N \l__stex_smsmode_filecontent_tl
1566   % \ior_open:Nn \c__stex_smsmode_ior {#1}
1567   % \ior_map_inline:Nn \c__stex_smsmode_ior {
1568     % \tl_put_right:Nn \l__stex_smsmode_filecontent_tl { ##1 }
1569   % }
1570   % \ior_close:N \c__stex_smsmode_ior
1571   \stex_filestack_push:n{#1}
1572   \stex_in_smsmode:nn{#1} {
1573     #2
1574     \everyeof{\q__stex_smsmode_break\noexpand}
1575     \expandafter\expandafter\expandafter
1576     \stex_smsmode_do:

```

```

1577 \csname @ @ input\endcsname "#1"\relax
1578 %\expandafter \stex_smsmode_do: \l__stex_smsmode_filecontent_tl
1579 }
1580 \stex_filestack_pop:
1581 }

```

(End definition for \stex\_in\_smsmode:nn. This function is documented on page 32.)

**\stex\_smsmode\_do:** is executed on encountering \ in smsmode. It checks whether the corresponding command is allowed and executes or ignores it accordingly:

```

1582 \cs_new_protected:Npn \stex_smsmode_do: {
1583   \stex_if_smsmode:T {
1584     \__stex_smsmode_do:w
1585   }
1586 }
1587 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1588   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
1589     \expandafter\if\expandafter\relax\noexpand#1
1590     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1591   } \else\expandafter\__stex_smsmode_do:w\fi
1592 }{
1593   \__stex_smsmode_do:w % #1
1594 }
1595 }
1596 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1597   \cs_if_eq:NNTF #1 \q__stex_smsmode_break {
1598     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1599       #1\__stex_smsmode_do:w
1600     }{
1601       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1602         #1
1603       }{
1604         \cs_if_eq:NNTF \begin #1 {
1605           \__stex_smsmode_check_begin:n
1606         }{
1607           \cs_if_eq:NNTF \end #1 {
1608             \__stex_smsmode_check_end:n
1609           }{
1610             \__stex_smsmode_do:w
1611           }
1612         }
1613       }
1614     }
1615   }
1616 }
1617
1618 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1619   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1620     \begin{#1}
1621   }{
1622     \__stex_smsmode_do:w
1623   }
1624 }
1625 \cs_new_protected:Nn \__stex_smsmode_check_end:n {

```

```

1626 \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1627   \end{#1}\__stex_smsmode_do:w
1628 }{
1629   \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1630 }
1631 }

```

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

## 29.2 Inheritance

```

1632 <@@=stex_importmodule>

```

`\stex_import_module_uri:nn`

```

1633 \cs_new_protected:Nn \stex_import_module_uri:nn {
1634   \str_set:Nx \l_stex_import_archive_str { #1 }
1635   \str_set:Nn \l_stex_import_path_str { #2 }
1636
1637   \exp_args:NNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1638   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1639   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1640
1641   \stex_modules_current_namespace:
1642   \bool_lazy_all:nTF {
1643     {\str_if_empty_p:N \l_stex_import_archive_str}
1644     {\str_if_empty_p:N \l_stex_import_path_str}
1645     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1646   }{
1647     \str_set_eq:NN \l_stex_import_path_str \l_stex_modules_subpath_str
1648     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1649   }{
1650     \str_if_empty:NT \l_stex_import_archive_str {
1651       \prop_if_exist:NT \l_stex_current_repository_prop {
1652         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1653       }
1654     }
1655     \str_if_empty:NTF \l_stex_import_archive_str {
1656       \str_if_empty:NF \l_stex_import_path_str {
1657         \str_set:Nx \l_stex_import_ns_str {
1658           \l_stex_module_ns_str / \l_stex_import_path_str
1659         }
1660       }
1661     }{
1662       \stex_require_repository:n \l_stex_import_archive_str
1663       \prop_get:cnN { c_stex_mathhub_\l_stex_import_archive_str_manifest_prop } { ns }
1664       \l_stex_import_ns_str
1665       \str_if_empty:NF \l_stex_import_path_str {
1666         \str_set:Nx \l_stex_import_ns_str {
1667           \l_stex_import_ns_str / \l_stex_import_path_str
1668         }
1669       }
1670     }
1671   }
1672 }

```

(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 1673 \str_new:N \l_stex_import_name_str
\l_stex_import_path_str 1674 \str_new:N \l_stex_import_archive_str
\l_stex_import_ns_str 1675 \str_new:N \l_stex_import_path_str
1676 \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)}
1677 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1678   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1679
1680     % archive
1681     \str_set:Nx \l_tmpa_str { #2 }
1682     \str_if_empty:NTF \l_tmpa_str {
1683       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1684     } {
1685       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1686       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1687       \seq_put_right:Nn \l_tmpa_seq { source }
1688     }
1689
1690     % path
1691     \str_set:Nx \l_tmpb_str { #3 }
1692     \str_if_empty:NTF \l_tmpb_str {
1693       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1694
1695       \ltx@ifpackageloaded{babel} {
1696         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1697           { \languagename } \l_tmpb_str {
1698           \msg_error:nnx{stex}{error/unknownlanguage}{\languagename}
1699         }
1700       } {
1701         \str_clear:N \l_tmpb_str
1702       }
1703
1704       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1705       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1706         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1707       }{
1708         \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1709         \IfFileExists{ \l_tmpa_str.tex }{
1710           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1711         }{
1712           % try english as default
1713           \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1714           \IfFileExists{ \l_tmpa_str.en.tex }{
1715             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1716           }{
1717             \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1718           }
1719         }

```

```

1720     }
1721
1722   } {
1723     \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1724     \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1725
1726     \ltx@ifpackageloaded{babel} {
1727       \exp_args:NnX \prop_get:NnNF \c_stex_language_abbrevs_prop
1728         { \language } \l_tmpb_str {
1729         \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1730       }
1731     } {
1732       \str_clear:N \l_tmpb_str
1733     }
1734
1735     \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1736
1737     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1738     \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1739       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1740     }{
1741       \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1742       \IfFileExists{ \l_tmpa_str/#4.tex }{
1743         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1744       }{
1745         % try english as default
1746         \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1747         \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1748           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1749         }{
1750           \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1751           \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1752             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1753           }{
1754             \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1755             \IfFileExists{ \l_tmpa_str.tex }{
1756               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1757             }{
1758               % try english as default
1759               \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1760               \IfFileExists{ \l_tmpa_str.en.tex }{
1761                 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1762               }{
1763                 \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1764               }
1765             }
1766           }
1767         }
1768       }
1769     }
1770   }
1771
1772   \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
1773     \seq_clear:N \l_stex_all_modules_seq

```



```

1774     \str_clear:N \l_stex_current_module_str
1775     \str_set:Nx \l_tmpb_str { #2 }
1776     \str_if_empty:NF \l_tmpb_str {
1777         \stex_set_current_repository:n { #2 }
1778     }
1779     \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1780 }
1781
1782 \stex_if_module_exists:nF { #1 ? #4 } {
1783     \msg_error:nnx{stex}{error/unknownmodule}{
1784         #1?#4~(in~file~\g__stex_importmodule_file_str)
1785     }
1786 }
1787 }
1788 \stex_activate_module:n { #1 ? #4 }
1789 }

```

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

### `\importmodule`

```

1790 \NewDocumentCommand \importmodule { 0{} m } {
1791     \stex_import_module_uri:nn { #1 } { #2 }
1792     \stex_debug:nn{modules}{Importing~module:~
1793         \l_stex_import_ns_str ? \l_stex_import_name_str
1794     }
1795     \stex_if_smsmode:F {
1796         \stex_import_require_module:nnnn
1797         { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1798         { \l_stex_import_path_str } { \l_stex_import_name_str }
1799         \stex_annotate_invisible:nnn
1800         {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1801     }
1802     \exp_args:Nx \stex_add_to_current_module:n {
1803         \stex_import_require_module:nnnn
1804         { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1805         { \l_stex_import_path_str } { \l_stex_import_name_str }
1806     }
1807     \exp_args:Nx \stex_add_import_to_current_module:n {
1808         \l_stex_import_ns_str ? \l_stex_import_name_str
1809     }
1810     \stex_smsmode_do:
1811 }
1812 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

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

### `\usemodule`

```

1813 \NewDocumentCommand \usemodule { 0{} m } {
1814     \stex_if_smsmode:F {
1815         \stex_import_module_uri:nn { #1 } { #2 }
1816         \stex_import_require_module:nnnn
1817         { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1818         { \l_stex_import_path_str } { \l_stex_import_name_str }
1819         \stex_annotate_invisible:nnn
1820         {usemodule} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}

```

```

1821 }
1822 \stex_smsmode_do:
1823 }

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

1824 </package>

```

## Chapter 30

# STEX -Symbols Implementation

```
1825 <*package>
1826
1827 %%%%%%%%%%%%% symbols.dtx %%%%%%%%%%%%%
1828
```

Warnings and error messages

```
1829
```

### 30.1 Symbol Declarations

```
1830 <@@=stex_symdecl>
```

`\l_stex_all_symbols_seq` Stores all available symbols

```
1831 \seq_new:N \l_stex_all_symbols_seq
```

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

`\STEXsymbol`

```
1832 \NewDocumentCommand \STEXsymbol { m } {
1833   \stex_get_symbol:n { #1 }
1834   \exp_args:No
1835   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1836 }
```

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

symdecl arguments:

```
1837 \keys_define:nn { stex / symdecl } {
1838   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1839   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1840   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1841   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
1842   align     .str_set:N    = \l_stex_symdecl_align_str , % TODO(?)
1843   gfc       .str_set:N    = \l_stex_symdecl_gfc_str , % TODO(?)
1844   specializes .str_set:N  = \l_stex_symdecl_specializes_str , % TODO(?)
1845   def       .tl_set:N    = \l_stex_symdecl_definiens_tl
1846 }
```

```

1847
1848 \bool_new:N \l_stex_symdecl_make_macro_bool
1849
1850 \cs_new_protected:Nn \__stex_symdecl_args:n {
1851   \str_clear:N \l_stex_symdecl_name_str
1852   \str_clear:N \l_stex_symdecl_args_str
1853   \bool_set_false:N \l_stex_symdecl_local_bool
1854   \tl_clear:N \l_stex_symdecl_type_tl
1855   \tl_clear:N \l_stex_symdecl_definiens_tl
1856
1857   \keys_set:nn { stex / symdecl } { #1 }
1858 }

```

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

```

1859
1860 \NewDocumentCommand \symdecl { s O{} m } {
1861   \__stex_symdecl_args:n { #2 }
1862   \IfBooleanTF #1 {
1863     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1864   } {
1865     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1866   }
1867   \stex_symdecl_do:n { #3 }
1868   \stex_smsmode_do:
1869 }
1870 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

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

```

1871 \cs_new_protected:Nn \stex_symdecl_do:n {
1872   \stex_if_in_module:F {
1873     % TODO throw error? some default namespace?
1874   }
1875
1876   \str_if_empty:NT \l_stex_symdecl_name_str {
1877     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1878   }
1879
1880   \prop_if_exist:cT { l_stex_symdecl_
1881     \l_stex_current_module_str ?
1882     \l_stex_symdecl_name_str
1883     _prop
1884   }{
1885     % TODO throw error (beware of circular dependencies)
1886   }
1887
1888   \prop_clear:N \l_tmpa_prop
1889   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
1890   \seq_clear:N \l_tmpa_seq
1891   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1892   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1893

```

```

1894 \exp_args:No \stex_add_constant_to_current_module:n {
1895   \l_stex_symdecl_name_str
1896 }
1897
1898 % arity/args
1899 \int_zero:N \l_tmpb_int
1900
1901 \bool_set_true:N \l_tmpa_bool
1902 \str_map_inline:Nn \l_stex_symdecl_args_str {
1903   \token_case_meaning:NnF ##1 {
1904     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1905     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1906     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1907     {\tl_to_str:n a} {
1908       \bool_set_false:N \l_tmpa_bool
1909       \int_incr:N \l_tmpb_int
1910     }
1911     {\tl_to_str:n B} {
1912       \bool_set_false:N \l_tmpa_bool
1913       \int_incr:N \l_tmpb_int
1914     }
1915   }{
1916     \msg_set:nnn{stex}{error/wrongargs}{
1917       args~value~in~symbol~declaration~for~
1918       \l_stex_current_module_str ?
1919       \l_stex_symdecl_name_str ~
1920       needs~to~be~
1921       i,~a,~b~or~B,~but~##1~given
1922     }
1923     \msg_error:nn{stex}{error/wrongargs}
1924   }
1925 }
1926 \bool_if:NTF \l_tmpa_bool {
1927   % possibly numeric
1928   \str_if_empty:NTF \l_stex_symdecl_args_str {
1929     \prop_put:Nnn \l_tmpa_prop { args } {}
1930     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1931   }{
1932     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1933     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1934     \str_clear:N \l_tmpa_str
1935     \int_step_inline:nn \l_tmpa_int {
1936       \str_put_right:Nn \l_tmpa_str i
1937     }
1938     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1939   }
1940 } {
1941   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1942   \prop_put:Nnx \l_tmpa_prop { arity }
1943     { \str_count:N \l_stex_symdecl_args_str }
1944 }
1945 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
1946
1947

```

```

1948 % semantic macro
1949
1950 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1951   \exp_args:Nx \stex_do_aftergroup:n {
1952     \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1953       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1954     }}
1955   }
1956
1957   \bool_if:NF \l_stex_symdecl_local_bool {
1958     \exp_args:Nx \stex_add_to_current_module:n {
1959       \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1960         \l_stex_current_module_str ? \l_stex_symdecl_name_str
1961       } }
1962     }
1963   }
1964 }
1965
1966 % add to all symbols
1967
1968 \bool_if:NF \l_stex_symdecl_local_bool {
1969   \exp_args:Nx \stex_add_to_current_module:n {
1970     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1971       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1972     }
1973   }
1974 %   \exp_args:Nx \stex_add_field_to_current_module:n {
1975 %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1976 %   }
1977 }
1978
1979 \stex_debug:nn{symbols}{New~symbol:~
1980   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
1981   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
1982   Args:~\prop_item:Nn \l_tmpa_prop { args }
1983 }
1984
1985 % circular dependencies require this:
1986
1987 \prop_if_exist:cF {
1988   l_stex_symdecl_
1989   \l_stex_current_module_str ? \l_stex_symdecl_name_str
1990   _prop
1991 } {
1992   \prop_set_eq:cN {
1993     l_stex_symdecl_
1994     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1995     _prop
1996   } \l_tmpa_prop
1997 }
1998
1999 \seq_clear:c {
2000   l_stex_symdecl_
2001   \l_stex_current_module_str ? \l_stex_symdecl_name_str

```

```

2002   _notations
2003 }
2004
2005 \bool_if:NF \l_stex_symdecl_local_bool {
2006   \exp_args:Nx
2007   \stex_add_to_current_module:n {
2008     \seq_clear:c {
2009       l_stex_symdecl_
2010       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2011       _notations
2012     }
2013     \prop_set_from_keyval:cn {
2014       l_stex_symdecl_
2015       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2016       _prop
2017     } {
2018       name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2019       module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2020       type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2021       args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2022       arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2023       assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
2024     }
2025   }
2026 }
2027
2028 \stex_if_smsmode:TF {
2029   \bool_if:NF \l_stex_symdecl_local_bool {
2030     % \exp_args:Nx \stex_add_to_sms:n {
2031     %   \prop_set_from_keyval:cn {
2032     %     l_stex_symdecl_
2033     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2034     %     _prop
2035     %   } {
2036     %     name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2037     %     module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2038     %     local     = \prop_item:Nn \l_tmpa_prop { local }     ,
2039     %     type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2040     %     args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2041     %     arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2042     %     assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
2043     %   }
2044     %   \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2045     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2046     %   }
2047     % }
2048   }
2049 }{
2050   \exp_args:Nx \stex_do_aftergroup:n {
2051     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2052       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2053     }
2054   }
2055   \stex_if_do_html:T {

```

```

2056 \stex_annotate_invisible:nnn {symdecl} {
2057   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2058 } {
2059   \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}{}}{ $\l_st
2060   \stex_annotate_invisible:nnn{args}{}{
2061     \prop_item:Nn \l_tmpa_prop { args }
2062   }
2063   \stex_annotate_invisible:nnn{macroname}{#1}{}
2064   \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2065     \stex_annotate_invisible:nnn{definiens}{}
2066     { $\l_stex_symdecl_definiens_tl$ }
2067   }
2068 }
2069 }
2070 }
2071 }

```

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

`\stex_get_symbol:n`

```

2072 \str_new:N \l_stex_get_symbol_uri_str
2073
2074 \cs_new_protected:Nn \stex_get_symbol:n {
2075   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2076     \__stex_symdecl_get_symbol_from_cs:n { #1 }
2077   }{
2078     % argument is a string
2079     % is it a command name?
2080     \cs_if_exist:cTF { #1 }{
2081       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2082       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2083       \str_if_empty:NNTF \l_tmpa_str {
2084         \exp_args:Nx \cs_if_eq:NNTF {
2085           \tl_head:N \l_tmpa_tl
2086         } \stex_invoke_symbol:n {
2087           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
2088         }{
2089           \__stex_symdecl_get_symbol_from_string:n { #1 }
2090         }
2091       } {
2092         \__stex_symdecl_get_symbol_from_string:n { #1 }
2093       }
2094     }{
2095       % argument is not a command name
2096       \__stex_symdecl_get_symbol_from_string:n { #1 }
2097       % \l_stex_all_symbols_seq
2098     }
2099   }
2100 }
2101
2102 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2103   \str_set:Nn \l_tmpa_str { #1 }
2104   \bool_set_false:N \l_tmpa_bool
2105   \stex_if_in_module:T {

```



```

2106 \exp_args:Nno \seq_if_in:cnT {c_stex_module_\l_stex_current_module_str _constants} { \l_
2107 \bool_set_true:N \l_tmpa_bool
2108 \str_set:Nx \l_stex_get_symbol_uri_str {
2109 \l_stex_current_module_str ? #1
2110 }
2111 }
2112 }
2113 \bool_if:NF \l_tmpa_bool {
2114 \tl_set:Nn \l_tmpa_tl {
2115 \msg_set:nnn{stex}{error/unknownsymbol}{
2116 No~symbol~#1~found!
2117 }
2118 \msg_error:nn{stex}{error/unknownsymbol}
2119 }
2120 \str_set:Nn \l_tmpa_str { #1 }
2121 \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2122 \seq_map_inline:Nn \l_stex_all_symbols_seq {
2123 \str_set:Nn \l_tmpb_str { ##1 }
2124 \str_if_eq:eeT { \l_tmpa_str } {
2125 \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
2126 } {
2127 \seq_map_break:n {
2128 \tl_set:Nn \l_tmpa_tl {
2129 \str_set:Nn \l_stex_get_symbol_uri_str {
2130 ##1
2131 }
2132 }
2133 }
2134 }
2135 }
2136 \l_tmpa_tl
2137 }
2138 }
2139
2140 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
2141 \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2142 { \tl_tail:N \l_tmpa_tl }
2143 \tl_if_single:NTF \l_tmpa_tl {
2144 \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2145 \exp_after:wN \str_set:Nn \exp_after:wN
2146 \l_stex_get_symbol_uri_str \l_tmpa_tl
2147 }{
2148 % TODO
2149 % tail is not a single group
2150 }
2151 }{
2152 % TODO
2153 % tail is not a single group
2154 }
2155 }

```

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

## 30.2 Notations

```

2156 <@@=stex_notation>

notation arguments:
2157 \keys_define:nn { stex / notation } {
2158   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2159   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2160   prec .str_set_x:N = \l__stex_notation_prec_str ,
2161   op .tl_set:N = \l__stex_notation_op_tl ,
2162   primary .bool_set:N = \l__stex_notation_primary_bool ,
2163   primary .default:n = {true} ,
2164   unknown .code:n = \str_set:Nx
2165     \l__stex_notation_variant_str \l_keys_key_str
2166 }
2167
2168 \cs_new_protected:Nn \stex_notation_args:n {
2169   \str_clear:N \l__stex_notation_lang_str
2170   \str_clear:N \l__stex_notation_variant_str
2171   \str_clear:N \l__stex_notation_prec_str
2172   \tl_clear:N \l__stex_notation_op_tl
2173   \bool_set_false:N \l__stex_notation_primary_bool
2174
2175   \keys_set:nn { stex / notation } { #1 }
2176 }

```

**\notation**

```

2177 \NewDocumentCommand \notation { 0{ } m } {
2178   \stex_notation_args:n { #1 }
2179   \tl_clear:N \l_stex_symdecl_definiens_tl
2180   \stex_get_symbol:n { #2 }
2181   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
2182 }
2183 \stex_deactivate_macro:Nn \notation {module-environments}

```

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

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

```

2184 \seq_new:N \l__stex_notation_precedences_seq
2185 \tl_new:N \l__stex_notation_opprec_tl
2186 \int_new:N \l__stex_notation_currarg_int
2187
2188 \cs_new_protected:Nn \stex_notation_do:nn {
2189   \let\l_stex_current_symbol_str\relax
2190   \str_set:Nx \l__stex_notation_symbol_str { #1 }
2191   \seq_clear:N \l__stex_notation_precedences_seq
2192   \tl_clear:N \l__stex_notation_opprec_tl
2193   \prop_get:cnN {
2194     l_stex_symdecl_ #1 _prop
2195   } { args } \l__stex_notation_args_str
2196
2197   % precedences
2198   \prop_get:cnN {
2199     l_stex_symdecl_ #1 _prop
2200   } { arity } \l__stex_notation_arity_str

```

```

2201 \str_if_empty:NTF \l__stex_notation_prec_str {
2202   \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2203     \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2204   }{
2205     \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2206   }
2207 } {
2208   \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2209     \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2210     \int_step_inline:nn { \l__stex_notation_arity_str } {
2211       \exp_args:NNo
2212       \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2213     }
2214   }{
2215     \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2216     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2217       \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2218       \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2219         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2220         \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2221         \seq_map_inline:Nn \l_tmpa_seq {
2222           \seq_put_right:Nn \l_tmpb_seq { ##1 }
2223         }
2224       }
2225     }{
2226       \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2227         \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2228       }{
2229         \tl_set:No \l__stex_notation_opprec_tl { 0 }
2230       }
2231     }
2232   }
2233 }
2234
2235 \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2236 \int_step_inline:nn { \l__stex_notation_arity_str } {
2237   \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
2238     \exp_args:NNo
2239     \seq_put_right:No \l__stex_notation_precedences_seq {
2240       \l__stex_notation_opprec_tl
2241     }
2242   }
2243 }
2244
2245 \tl_clear:N \l__stex_notation_dummyargs_tl
2246
2247 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2248   \exp_args:NNe
2249   \cs_set:Npn \l__stex_notation_macrocode_cs {
2250     \stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2251     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2252     { \l__stex_notation_opprec_tl }
2253     { \exp_not:n { #2 } }
2254   }

```

```

2255   \_stex_notation_final:
2256 }{
2257   \str_if_in:NnTF \l__stex_notation_args_str b {
2258     \exp_args:Nne \use:nn
2259     {
2260       \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2261       \cs_set:Npn \l__stex_notation_arity_str } { {
2262         \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2263         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2264         { \l__stex_notation_opprec_tl }
2265         { \exp_not:n { #2 } }
2266       }}
2267 }{
2268   \str_if_in:NnTF \l__stex_notation_args_str B {
2269     \exp_args:Nne \use:nn
2270     {
2271       \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2272       \cs_set:Npn \l__stex_notation_arity_str } { {
2273         \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2274         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2275         { \l__stex_notation_opprec_tl }
2276         { \exp_not:n { #2 } }
2277       } }
2278 }{
2279   \exp_args:Nne \use:nn
2280   {
2281     \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2282     \cs_set:Npn \l__stex_notation_arity_str } { {
2283       \_stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2284       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2285       { \l__stex_notation_opprec_tl }
2286       { \exp_not:n { #2 } }
2287     } }
2288   }
2289 }
2290
2291 \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2292 \int_zero:N \l__stex_notation_currarg_int
2293 \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2294 \_stex_notation_arguments:
2295 }
2296 }

```

(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

```

2297 \cs_new_protected:Nn \_stex_notation_arguments: {
2298   \int_incr:N \l__stex_notation_currarg_int
2299   \str_if_empty:NnTF \l__stex_notation_remaining_args_str {
2300     \_stex_notation_final:
2301   }{
2302     \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2303     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2304     \str_if_eq:VnTF \l_tmpa_str a {

```

```

2305     \_stex_notation_argument_assoc:n
2306   }{
2307     \str_if_eq:VnTF \l_tmpa_str B {
2308       \_stex_notation_argument_assoc:n
2309     }{
2310       \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2311       \tl_put_right:Nx \l__stex_notation_dummyargs_tl {
2312         { \_stex_term_math_arg:nnn
2313           { \int_use:N \l__stex_notation_currarg_int }
2314           { \l_tmpa_str }
2315           { #####\int_use:N \l__stex_notation_currarg_int }
2316         }
2317       }
2318       \_stex_notation_arguments:
2319     }
2320   }
2321 }
2322 }

```

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

\\_stex\_notation\_argument\_assoc:n

```

2323 \cs_new_protected:Nn \_stex_notation_argument_assoc:n {
2324
2325   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2326     {\l__stex_notation_arity_str}{
2327       #1
2328     }
2329   \int_zero:N \l_tmpa_int
2330   \tl_clear:N \l_tmpa_tl
2331   \str_map_inline:Nn \l__stex_notation_args_str {
2332     \int_incr:N \l_tmpa_int
2333     \tl_put_right:Nx \l_tmpa_tl {
2334       \str_if_eq:nnTF {##1}{a}{ {} }{
2335         \str_if_eq:nnTF {##1}{B}{ {} }{
2336           {##### \int_use:N \l_tmpa_int}
2337         }
2338       }
2339     }
2340   }
2341   \exp_after:wN\exp_after:wN\exp_after:wN \def
2342   \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2343   \exp_after:wN\exp_after:wN\exp_after:wN ##
2344   \exp_after:wN\exp_after:wN\exp_after:wN 1
2345   \exp_after:wN\exp_after:wN\exp_after:wN ##
2346   \exp_after:wN\exp_after:wN\exp_after:wN 2
2347   \exp_after:wN\exp_after:wN\exp_after:wN {
2348     \exp_after:wN \exp_after:wN \exp_after:wN
2349     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2350       \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2351     }
2352   }
2353
2354   \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str

```

```

2355 \tl_put_right:Nx \l__stex_notation_dummyargs_tl { {
2356   \stex_term_math_assoc_arg:nnnn
2357   { \int_use:N \l__stex_notation_currarg_int }
2358   { \l_tmpa_str }
2359   { ####\int_use:N \l__stex_notation_currarg_int }
2360   { \l_tmpa_cs {####1} {####2} }
2361 } }
2362 %\cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
2363 %\tl_put_right:Nx \l_tmpa_tl {
2364 % { \stex_term_math_assoc_arg:nnnn
2365 % { \int_use:N \l_tmpa_int }
2366 % { \l_tmpb_str }
2367 % \exp_args:No \exp_not:n
2368 % {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
2369 % { ####\int_use:N \l_tmpa_int }
2370 % }
2371 %}
2372 \__stex_notation_arguments:
2373 }

```

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

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

```

2374 \cs_new_protected:Nn \__stex_notation_final: {
2375   \exp_args:Nne \use:nn
2376   {
2377     \cs_generate_from_arg_count:cNnn {
2378       stex_notation_ \l__stex_notation_symbol_str \c_hash_str
2379       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2380       _cs
2381     }
2382     \cs_set:Npn \l__stex_notation_arity_str { { {
2383       \exp_after:wN \exp_after:wN \exp_after:wN
2384       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2385       { \exp_after:wN \l__stex_notation_macrocode_cs \l__stex_notation_dummyargs_tl }
2386     } }
2387
2388     \tl_if_empty:NF \l__stex_notation_op_tl {
2389       \cs_set:cpx {
2390         stex_op_notation_ \l__stex_notation_symbol_str \c_hash_str
2391         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2392         _cs
2393       } {
2394         \stex_term_oms:nnn {
2395           \l__stex_notation_symbol_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2396           \l__stex_notation_lang_str
2397         }{
2398           \l__stex_notation_symbol_str
2399         }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2400       }
2401     }
2402
2403     \exp_args:Ne
2404     \stex_add_to_current_module:n {

```

```

2405 \cs_generate_from_arg_count:cNnn {
2406   stex_notation_ \l__stex_notation_symbol_str \c_hash_str
2407   \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2408   _cs
2409 } \cs_set:Npn {\l__stex_notation_arity_str} {
2410   \exp_after:wN \exp_after:wN \exp_after:wN
2411   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2412   { \exp_after:wN \l__stex_notation_macrocode_cs \l__stex_notation_dummyargs_tl }
2413 }
2414 \tl_if_empty:NF \l__stex_notation_op_tl {
2415   \cs_set:cpn {
2416     stex_op_notation_ \l__stex_notation_symbol_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__stex_notation_symbol_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2422       \l__stex_notation_lang_str
2423     }{
2424       \l__stex_notation_symbol_str
2425     }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2426   }
2427 }
2428 }
2429 \exp_args:Nx
2430 % \stex_do_aftergroup:n {
2431   \seq_put_right:cx {
2432     l_stex_symdecl_ \l__stex_notation_symbol_str
2433     _notations
2434   } {
2435     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2436   }
2437 % }
2438
2439 \stex_debug:nn{symbols}{
2440   Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2441   ~for~\l__stex_notation_symbol_str^^J
2442   Operator~precedence:~\l__stex_notation_opprec_tl^^J
2443   Argument~precedences:~
2444   \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2445   Notation: \cs_meaning:c {
2446     stex_notation_ \l__stex_notation_symbol_str \c_hash_str
2447     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2448     _cs
2449   }
2450 }
2451
2452 %\prop_set_eq:cN {
2453 %   l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2454 %   \c_hash_str \l__stex_notation_lang_str _prop
2455 %} \l_tmpb_prop
2456
2457 \exp_args:Ne
2458 \stex_add_to_current_module:n {

```

```

2459 \seq_put_right:cn {
2460   l_stex_symdecl_ \l__stex_notation_symbol_str
2461   _notations
2462 } {
2463   \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2464 }
2465 %\prop_set_from_keyval:cn {
2466 % l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2467 %   \c_hash_str \l__stex_notation_lang_str _prop
2468 %} {
2469 % symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2470 % language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2471 % variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,
2472 % opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2473 % argprec     = \prop_item:Nn \l_tmpb_prop { argprec }     ,
2474 %}
2475 }
2476
2477 \stex_if_smsmode:TF {
2478 %   \exp_args:Nx \stex_add_to_sms:n {
2479 %     \prop_set_from_keyval:cn {
2480 %       l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2481 %       \c_hash_str \l__stex_notation_lang_str _prop
2482 %     } {
2483 %       symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2484 %       language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2485 %       variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,
2486 %       opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2487 %       argprec     = \prop_item:Nn \l_tmpb_prop { argprec }     ,
2488 %     }
2489 %   }
2490 }{
2491
2492 % HTML annotations
2493 \stex_if_do_html:T {
2494   \stex_annotate_invisible:nnn { notation }
2495   { \l__stex_notation_symbol_str } {
2496     \stex_annotate_invisible:nnn { notationfragment }
2497     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
2498   \stex_annotate_invisible:nnn { precedence }
2499   { \l__stex_notation_prec_str }{}
2500
2501   \int_zero:N \l_tmpa_int
2502   \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2503   \tl_clear:N \l_tmpa_tl
2504   \int_step_inline:nn { \l__stex_notation_arity_str }{
2505     \int_incr:N \l_tmpa_int
2506     \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2507     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_r
2508     \str_if_eq:VnTF \l_tmpb_str a {
2509       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2510         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2511         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2512       } }

```



```

2513     }{
2514       \str_if_eq:VnTF \l_tmpb_str B {
2515         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2516           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2517           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2518         } }
2519       }{
2520         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2521           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2522         } }
2523       }
2524     }
2525   }
2526   \stex_annotate_invisible:nnn { notationcomp }{}{
2527     \str_set:Nx \l_stex_current_symbol_str { \l__stex_notation_symbol_str }
2528     $ \exp_args:Nno \use:nn { \use:c {
2529       stex_notation_ \l_stex_current_symbol_str
2530       \c_hash_str \l__stex_notation_variant_str
2531       \c_hash_str \l__stex_notation_lang_str _cs
2532     } } { \l_tmpa_tl } $
2533   }
2534 }
2535 }
2536 }
2537 \stex_smsmode_do:
2538 }

```

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

\setnotation

```

2539 \keys_define:nn { stex / setnotation } {
2540   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2541   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2542   unknown .code:n = \str_set:Nx
2543     \l__stex_notation_variant_str \l_keys_key_str
2544 }
2545
2546 \cs_new_protected:Nn \stex_setnotation_args:n {
2547   \str_clear:N \l__stex_notation_lang_str
2548   \str_clear:N \l__stex_notation_variant_str
2549   \keys_set:nn { stex / setnotation } { #1 }
2550 }
2551
2552 \NewDocumentCommand \setnotation {m m} {
2553   \stex_get_symbol:n { #1 }
2554   \_stex_setnotation_args:n { #2 }
2555   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl \l_stex_get_symbol_uri_str _notations }
2556     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2557     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl \l_stex_get_symbol_uri_str _notation
2558       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2559     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl \l_stex_get_symbol_uri_str _notation
2560       { \c_hash_str }
2561     \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl \l_stex_get_symbol_uri_str _notations
2562       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }

```

```

2563 \exp_args:Nx \stex_add_to_current_module:n {
2564 \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _notati
2565 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2566 \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _notation
2567 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2568 \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _notati
2569 { \c_hash_str }
2570 }
2571 \stex_debug:nn {notations}{
2572 Setting~default~notation~
2573 {\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str}~for~
2574 \l_stex_get_symbol_uri_str \
2575 \expandafter\meaning\csname
2576 l_stex_symdecl_\l_stex_get_symbol_uri_str _notations\endcsname
2577 }
2578 }{
2579 % todo throw error
2580 }
2581 \stex_smsmode_do:
2582 }
2583
2584 \cs_new_protected:Nn \stex_copy_notations:nn {
2585 \stex_debug:nn {notations}{
2586 Copying~notations~from~#2~to~#1\
2587 \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
2588 }
2589 \tl_clear:N \l_tmpa_tl
2590 \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2591 \tl_put_right:Nn \l_tmpa_tl { {## #1} }
2592 }
2593 \seq_map_inline:cn {l_stex_symdecl_#2_notations}{
2594 \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2595 \edef \l_tmpa_tl {
2596 \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2597 \exp_after:wN\exp_after:wN\exp_after:wN {
2598 \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2599 }
2600 }
2601 \exp_args:Nx
2602 \stex_do_aftergroup:n {
2603 \seq_put_right:cn{l_stex_symdecl_#1_notations}{##1}
2604 \cs_generate_from_arg_count:cNnn {
2605 stex_notation_ #1 \c_hash_str ##1 _cs
2606 } \cs_set:Npn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }{
2607 \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl}
2608 }
2609 }
2610 }
2611 }
2612
2613 \NewDocumentCommand \copynotation {m m} {
2614 \stex_get_symbol:n { #1 }
2615 \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2616 \stex_get_symbol:n { #2 }

```

```

2617 \exp_args:Noo
2618 \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2619 \exp_args:Nx \stex_add_import_to_current_module:n{
2620   \stex_copy_notations:nn {\l_tmpa_str} {\l_stex_get_symbol_uri_str}
2621 }
2622 \stex_smsmode_do:
2623 }
2624

```

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

**\symdef**

```

2625 \keys_define:nn { stex / symdef } {
2626   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2627   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2628   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2629   type      .tl_set:N   = \l_stex_symdecl_type_tl ,
2630   def       .tl_set:N   = \l_stex_symdecl_definiens_tl ,
2631   op        .tl_set:N   = \l__stex_notation_op_tl ,
2632   lang      .str_set_x:N = \l__stex_notation_lang_str ,
2633   variant   .str_set_x:N = \l__stex_notation_variant_str ,
2634   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2635   unknown   .code:n     = \str_set:Nx
2636             \l__stex_notation_variant_str \l_keys_key_str
2637 }
2638
2639 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2640   \str_clear:N \l_stex_symdecl_name_str
2641   \str_clear:N \l_stex_symdecl_args_str
2642   \bool_set_false:N \l_stex_symdecl_local_bool
2643   \tl_clear:N \l_stex_symdecl_type_tl
2644   \tl_clear:N \l_stex_symdecl_definiens_tl
2645   \str_clear:N \l__stex_notation_lang_str
2646   \str_clear:N \l__stex_notation_variant_str
2647   \str_clear:N \l__stex_notation_prec_str
2648   \tl_clear:N \l__stex_notation_op_tl
2649
2650   \keys_set:nn { stex / symdef } { #1 }
2651 }
2652
2653 \NewDocumentCommand \symdef { 0{} m } {
2654   \__stex_notation_symdef_args:n { #1 }
2655   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2656   \stex_symdecl_do:n { #2 }
2657   \exp_args:Nx \stex_notation_do:nn {
2658     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2659   }
2660 }
2661 \stex_deactivate_macro:Nn \symdef {module~environments}

```

(End definition for \symdef. This function is documented on page 37.)

```

2662 \</package>

```

## Chapter 31

# STEX -Terms Implementation

```
2663 <*package>
2664
2665 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2666
2667 <@@=stex_terms>
2668
2669 Warnings and error messages
2670 \msg_new:nnn{stex}{error/nonotation}{
2671   Symbol~#1~invoked,~but~has~no~notation~#2!
2672 }
2673 \msg_new:nnn{stex}{error/notationarg}{
2674   Error~in~parsing~notation~#1
2675 }
2676 \msg_new:nnn{stex}{error/noop}{
2677   Symbol~#1~has~no~operator~notation~for~notation~#2
2678 }
2679
```

### 31.1 Symbol Invocations

Arguments:

```
2678 \keys_define:nn { stex / terms } {
2679   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2680   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2681   unknown .code:n = \str_set:Nx
2682     \l__stex_terms_variant_str \l_keys_key_str
2683 }
2684
2685 \cs_new_protected:Nn \__stex_terms_args:n {
2686   \str_clear:N \l__stex_terms_lang_str
2687   \str_clear:N \l__stex_terms_variant_str
2688   \str_clear:N \l__stex_terms_prec_str
2689   \tl_clear:N \l__stex_terms_op_tl
2690
2691   \keys_set:nn { stex / terms } { #1 }
```

2692 }

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

```
2693 \cs_new_protected:Nn \stex_invoke_symbol:n {
2694   \if_mode_math:
2695     \exp_after:wN \__stex_terms_invoke_math:n
2696   \else:
2697     \exp_after:wN \__stex_terms_invoke_text:n
2698   \fi: { #1 }
2699 }
```

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

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

```
2700 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2701   \peek_charcode_remove:NTF ! {
2702     \peek_charcode:NTF [ {
2703       \__stex_terms_invoke_op:nw { #1 }
2704     }{
2705       \peek_charcode_remove:NTF ! {
2706         \peek_charcode:NTF [ {
2707           \__stex_terms_invoke_op_custom:nw
2708         }{
2709           % TODO throw error
2710         }
2711       }{
2712         \__stex_terms_invoke_op:nw { #1 } []
2713       }
2714     }
2715   }{
2716     \peek_charcode_remove:NTF * {
2717       \__stex_terms_invoke_text:n { #1 }
2718     }{
2719       \peek_charcode:NTF [ {
2720         \__stex_terms_invoke_math:nw { #1 }
2721       }{
2722         \__stex_terms_invoke_math:nw { #1 } []
2723       }
2724     }
2725   }
2726 }
```

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

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

```
2727 \cs_new_protected:Npn \__stex_terms_invoke_op_custom:nw #1 [#2] {
2728   \stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
2729     \stex_highlight_term:nn{#1}{#2}
2730   }
2731 }
```

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

\\_stex\_terms\_invoke\_op:nw

```

2732 \cs_new_protected:Npn \_stex_terms_invoke_op:nw #1 [#2] {
2733   \_stex_terms_args:n { #2 }
2734   \cs_if_exist:cTF {
2735     stex_op_notation_ #1 \c_hash_str
2736     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2737   }{
2738     \csname stex_op_notation_ #1 \c_hash_str
2739       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2740     \endcsname
2741   }{
2742     \msg_error:nnxx{stex}{error/noop}{#1}{\l__stex_terms_variant_str \c_hash_str \l__stex_te
2743   }
2744 }

```

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

\\_stex\_terms\_invoke\_math:nw

```

2745 \cs_new_protected:Npn \_stex_terms_invoke_math:nw #1 [#2] {
2746   \_stex_terms_args:n { #2 }
2747   \seq_if_empty:cTF {
2748     l_stex_symdecl_ #1 _notations
2749   } {
2750     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
2751   } {
2752     \seq_if_in:cxTF {
2753       l_stex_symdecl_ #1 _notations
2754     }
2755     { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
2756       \str_set:Nn \l_stex_current_symbol_str { #1 }
2757       \stex_debug:nn{terms}{Using~
2758         #1\c_hash_str\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str \\
2759         \expandafter\meaning\csname stex_notation_ #1 \c_hash_str
2760         \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2761         _cs\endcsname
2762       }
2763       \use:c{
2764         stex_notation_ #1 \c_hash_str
2765         \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2766         _cs
2767       }
2768     }{
2769       \str_if_empty:NTF \l__stex_terms_variant_str {
2770         \str_if_empty:NTF \l__stex_terms_lang_str {
2771           \seq_get_left:cN {
2772             l_stex_symdecl_ #1 _notations
2773           } \l_tmpa_str
2774           \str_set:Nn \l_stex_current_symbol_str { #1 }
2775           \stex_debug:nn{terms}{Using~
2776             #1\c_hash_str\l_tmpa_str \\
2777             \expandafter\meaning\csname stex_notation_ #1 \c_hash_str
2778             \l_tmpa_str
2779             _cs\endcsname
2780           }

```

```

2781         \use:c{
2782             stex_notation_ #1 \c_hash_str \l_tmpa_str
2783             _cs
2784         }
2785     }{
2786         \msg_error:nnxx{stex}{error/nonotation}{#1}{
2787             ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2788         }
2789     }
2790 }{
2791     \msg_error:nnxx{stex}{error/nonotation}{#1}{
2792         ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2793     }
2794 }
2795 }
2796 }
2797 }

```

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

`\__stex_terms_invoke_text:n`

```

2798 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
2799     \peek_charcode_remove:NTF ! {
2800         \stex_term_custom:nn { #1 } { }
2801     }{
2802         \prop_set_eq:Nc \l_tmpa_prop {
2803             l_stex_symdecl_ #1 _prop
2804         }
2805         \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2806         \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2807     }
2808 }

```

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

## 31.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
2809 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2810 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2811 \int_new:N \l__stex_terms_downprec
2812 \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
2813 \tl_set:Nn \l__stex_terms_left_bracket_str (
2814 \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

```
2815 \cs_new_protected:Nn \_stex_terms_maybe_brackets:nn {
2816   \bool_if:NTF \l__stex_terms_brackets_done_bool {
2817     \bool_set_false:N \l__stex_terms_brackets_done_bool
2818     #2
2819   } {
2820     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
2821       \bool_if:NTF \l_stex_inarray_bool { #2 }{
2822         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
2823         \dobrackets { #2 }
2824       }
2825     }{ #2 }
2826   }
2827 }
```

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

**`\dobrackets`**

```
2828 \bool_new:N \l__stex_terms_brackets_done_bool
2829 %\RequirePackage{scalerel}
2830 \cs_new_protected:Npn \dobrackets #1 {
2831   %\ThisStyle{\if D\m@switch
2832   %   \exp_args:Nnx \use:nn
2833   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
2834   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
2835   % \else
2836   \exp_args:Nnx \use:nn
2837   {
2838     \bool_set_true:N \l__stex_terms_brackets_done_bool
2839     \int_set:Nn \l__stex_terms_downprec \infprec
2840     \l__stex_terms_left_bracket_str
2841     #1
2842   }
2843   {
2844     \bool_set_false:N \l__stex_terms_brackets_done_bool
2845     \l__stex_terms_right_bracket_str
2846     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2847   }
2848   %\fi}
2849 }
```

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

**`\withbrackets`**

```
2850 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2851   \exp_args:Nnx \use:nn
2852   {
2853     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
2854     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
2855     #3
2856   }
2857   {
2858     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
2859     {\l__stex_terms_left_bracket_str}
```



```

2860 \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
2861 {\l__stex_terms_right_bracket_str}
2862 }
2863 }

```

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

## `\STEXinvisible`

```

2864 \cs_new_protected:Npn \STEXinvisible #1 {
2865   \stex_annotate_invisible:n { #1 }
2866 }

```

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

OMDoc terms:

## `\_stex_term_math_oms:nnnn`

```

2867 \cs_new_protected:Nn \_stex_term_oms:nnn {
2868   \stex_annotate:nnn{ OMID }{ #2 }{
2869     \stex_highlight_term:nn { #1 } { #3 }
2870   }
2871 }
2872
2873 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2874   \__stex_terms_maybe_brackets:nn { #3 }{
2875     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2876   }
2877 }

```

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

## `\_stex_term_math_oma:nnnn`

```

2878 \cs_new_protected:Nn \_stex_term_oma:nnn {
2879   \stex_annotate:nnn{ OMA }{ #2 }{
2880     \stex_highlight_term:nn { #1 } { #3 }
2881   }
2882 }
2883
2884 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2885   \__stex_terms_maybe_brackets:nn { #3 }{
2886     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2887   }
2888 }

```

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

## `\_stex_term_math_omb:nnnn`

```

2889 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2890   \stex_annotate:nnn{ OMBIND }{ #2 }{
2891     \stex_highlight_term:nn { #1 } { #3 }
2892   }
2893 }
2894
2895 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2896   \__stex_terms_maybe_brackets:nn { #3 }{
2897     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }

```

```

2898 }
2899 }

```

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

`\_stex_term_math_arg:nnn`

```

2900 \cs_new_protected:Nn \_stex_term_arg:nn {
2901   \stex_unhighlight_term:n {
2902     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2903   }
2904 }
2905 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
2906   \exp_args:Nnx \use:nn
2907   { \int_set:Nn \l__stex_terms_downprec { #2 }
2908     \_stex_term_arg:nn { #1 }{ #3 }
2909   }
2910   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2911 }

```

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

`\_stex_term_math_assoc_arg:nnnn`

```

2912 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
2913   % TODO sequences
2914   \clist_set:Nn \l_tmpa_clist{ #3 }
2915   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
2916     \tl_set:Nn \l_tmpa_tl { #3 }
2917   }{
2918     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
2919     \clist_reverse:N \l_tmpa_clist
2920     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
2921
2922     \clist_map_inline:Nn \l_tmpa_clist {
2923       \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
2924         \exp_args:Nno
2925         \l_tmpa_cs { ##1 } \l_tmpa_tl
2926       }
2927     }
2928   }
2929   \exp_args:Nnno
2930   \_stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2931 }

```

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

`\stex_term_custom:nn`

```

2932 \cs_new_protected:Nn \stex_term_custom:nn {
2933   \str_set:Nn \l__stex_terms_custom_uri { #1 }
2934   \str_set:Nn \l_tmpa_str { #2 }
2935   \tl_clear:N \l_tmpa_tl
2936   \int_zero:N \l_tmpa_int
2937   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2938   \__stex_terms_custom_loop:
2939 }

```

(End definition for \stex\_term\_custom:nn. This function is documented on page 39.)

\\_stex\_terms\_custom\_loop:

```

2940 \cs_new_protected:Nn \_stex_terms_custom_loop: {
2941   \bool_set_false:N \l_tmpa_bool
2942   \bool_while_do:nn {
2943     \str_if_eq_p:ee X {
2944       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2945     }
2946   }{
2947     \int_incr:N \l_tmpa_int
2948   }
2949
2950   \peek_charcode:NTF [ {
2951     % notation/text component
2952     \_stex_terms_custom_component:w
2953   } {
2954     \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2955       % all arguments read => finish
2956       \_stex_terms_custom_final:
2957     } {
2958       % arguments missing
2959       \peek_charcode_remove:NTF * {
2960         % invisible, specific argument position or both
2961         \peek_charcode:NTF [ {
2962           % visible specific argument position
2963           \_stex_terms_custom_arg:wn
2964         } {
2965           % invisible
2966           \peek_charcode_remove:NTF * {
2967             % invisible specific argument position
2968             \_stex_terms_custom_arg_inv:wn
2969           } {
2970             % invisible next argument
2971             \_stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2972           }
2973         }
2974       } {
2975         % next normal argument
2976         \_stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
2977       }
2978     }
2979   }
2980 }

```

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

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

```

2981 \cs_new_protected:Npn \_stex_terms_custom_arg_inv:wn [ #1 ] #2 {
2982   \bool_set_true:N \l_tmpa_bool
2983   \_stex_terms_custom_arg:wn [ #1 ] { #2 }
2984 }

```

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

\\_stex\_terms\_custom\_arg:wn

```

2985 \cs_new_protected:Npn \_stex_terms_custom_arg:wn [ #1 ] #2 {
2986   \str_set:Nx \l_tmpb_str {
2987     \str_item:Nn \l_tmpa_str { #1 }
2988   }
2989   \str_case:VnTF \l_tmpb_str {
2990     { X } {
2991       \msg_error:nnx{stex}{error/notationarg}{\l_stex_terms_custom_uri}
2992     }
2993     { i } { \_stex_terms_custom_set_X:n { #1 } }
2994     { b } { \_stex_terms_custom_set_X:n { #1 } }
2995     { a } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
2996     { B } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
2997   }{}{
2998     \msg_error:nnx{stex}{error/notationarg}{\l_stex_terms_custom_uri}
2999   }
3000
3001   \bool_if:nTF \l_tmpa_bool {
3002     \tl_put_right:Nx \l_tmpa_tl {
3003       \stex_annotate_invisible:n {
3004         \_stex_term_arg:nn { \int_eval:n { #1 } }
3005         \exp_not:n { { #2 } }
3006       }
3007     }
3008   } {
3009     \tl_put_right:Nx \l_tmpa_tl {
3010       \_stex_term_arg:nn { \int_eval:n { #1 } }
3011       \exp_not:n { { #2 } }
3012     }
3013   }
3014
3015   \_stex_terms_custom_loop:
3016 }

```

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

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

```

3017 \cs_new_protected:Nn \_stex_terms_custom_set_X:n {
3018   \str_set:Nx \l_tmpa_str {
3019     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
3020     X
3021     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
3022   }
3023 }

```

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

\\_stex\_terms\_custom\_component:

```

3024 \cs_new_protected:Npn \_stex_terms_custom_component:w [ #1 ] {
3025   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
3026   \_stex_terms_custom_loop:
3027 }

```

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

`\_stex_terms_custom_final:`

```

3028 \cs_new_protected:Nn \_stex_terms_custom_final: {
3029   \int_compare:nNnTF \l_tmpb_int = 0 {
3030     \exp_args:Nnno \_stex_term_oms:nnn
3031   }{
3032     \str_if_in:NnTF \l_tmpa_str {b} {
3033       \exp_args:Nnno \_stex_term_ombind:nnn
3034     } {
3035       \exp_args:Nnno \_stex_term_oma:nnn
3036     }
3037   }
3038   { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
3039 }

```

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

`\symref`

`\symname`

```

3040 \NewDocumentCommand \symref { m m }{
3041   \let\compemph_uri_prev:\compemph@uri
3042   \let\compemph@uri\symrefemph@uri
3043   \STEXsymbol{#1}![#2]
3044   \let\compemph@uri\compemph_uri_prev:
3045 }
3046
3047 \keys_define:nn { stex / symname } {
3048   post .str_set_x:N = \l_stex_symname_post_str
3049 }
3050
3051 \cs_new_protected:Nn \stex_symname_args:n {
3052   \str_clear:N \l_stex_symname_post_str
3053   \keys_set:nn { stex / symname } { #1 }
3054 }
3055
3056 \NewDocumentCommand \symname { 0{} m }{
3057   \stex_symname_args:n { #1 }
3058   \stex_get_symbol:n { #2 }
3059   \str_set:Nx \l_tmpa_str {
3060     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3061   }
3062   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3063
3064   \let\compemph_uri_prev:\compemph@uri
3065   \let\compemph@uri\symrefemph@uri
3066   \exp_args:NNx \use:nn
3067   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }![
3068     \l_tmpa_str \l_stex_symname_post_str
3069   ] }
3070   \let\compemph@uri\compemph_uri_prev:
3071 }

```

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

## 31.3 Notation Components

3072 <@@=stex\_notationcomps>

`\stex_highlight_term:nn`

```

3073
3074 \str_new:N \l_stex_current_symbol_str
3075 \cs_new_protected:Nn \stex_highlight_term:nn {
3076   \exp_args:Nnx
3077   \use:nn {
3078     \str_set:Nx \l_stex_current_symbol_str { #1 }
3079     #2
3080   } {
3081     \str_set:Nx \exp_not:N \l_stex_current_symbol_str
3082     { \l_stex_current_symbol_str }
3083   }
3084 }
3085
3086 \cs_new_protected:Nn \stex_unhighlight_term:n {
3087   % \latexml_if:TF {
3088   %   #1
3089   % } {
3090   %   \rustex_if:TF {
3091   %     #1
3092   %   } {
3093     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
3094   % }
3095   % }
3096 }
```

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

```

\comp
\compemph@uri 3097 \cs_new_protected:Npn \comp #1 {
\compemph      3098   \str_if_empty:NF \l_stex_current_symbol_str {
\defemph       3099     \rustex_if:TF {
\defemph@uri   3100       \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
\symrefemph    3101     }{
\symrefemph@uri 3102       \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3103     }
3104   }
3105 }
3106
3107 \cs_new_protected:Npn \compemph@uri #1 #2 {
3108   \compemph{ #1 }
3109 }
3110
3111
3112 \cs_new_protected:Npn \compemph #1 {
3113   #1
3114 }
3115
3116 \cs_new_protected:Npn \defemph@uri #1 #2 {
3117   \defemph{#1}
3118 }
```

```

3119
3120 \cs_new_protected:Npn \defemph #1 {
3121     \textbf{#1}
3122 }
3123
3124 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3125     \symrefemph{#1}
3126 }
3127
3128 \cs_new_protected:Npn \symrefemph #1 {
3129     \textbf{#1}
3130 }

```

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

### **\ellipses**

```

3131 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix
\parrayline
\parraylineh
\parraycell
3132 \bool_new:N \l_stex_inarray_bool
3133 \bool_set_false:N \l_stex_inarray_bool
3134 \NewDocumentCommand \parray { m m } {
3135     \begingroup
3136     \bool_set_true:N \l_stex_inarray_bool
3137     \begin{array}{#1}
3138         #2
3139     \end{array}
3140 \endgroup
3141 }
3142
3143 \NewDocumentCommand \prmatrix { m } {
3144     \begingroup
3145     \bool_set_true:N \l_stex_inarray_bool
3146     \begin{matrix}
3147         #1
3148     \end{matrix}
3149 \endgroup
3150 }
3151
3152 \def \maybepline {
3153     \bool_if:NT \l_stex_inarray_bool {\hline}
3154 }
3155
3156 \def \parrayline #1 #2 {
3157     #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3158 }
3159
3160 \def \pmrow #1 { \parrayline{}{ #1 } }
3161
3162 \def \parraylineh #1 #2 {
3163     #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3164 }
3165

```

```

3166 \def \parraycell #1 {
3167   #1 \bool_if:NT \l_stex_inarray_bool {&}
3168 }

```

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

```

3169 \endpackage

```



## Chapter 32

# STEX -Structural Features Implementation

```
3170 <*package>
3171
3172 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3173
3174 <@@=stex_features>
3175
3176 Warnings and error messages
3177 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3178   Symbol~#1~can~not~be~assigned~in~copymodule~#2
3179 }
3180 \msg_new:nnn{stex}{error/interpretmodule/noddefinens}{
3181   Symbol~#1~not~assigned~in~interpretmodule~#2
3182 }
3183
```

### 32.1 Imports with modification

```
3182 \cs_new_protected:Nn \stex_get_symbol_in_copymodule:n {
3183   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3184     \__stex_features_get_symbol_from_cs:n { #1 }
3185   }{
3186     % argument is a string
3187     % is it a command name?
3188     \cs_if_exist:cTF { #1 }{
3189       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3190       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3191       \str_if_empty:NNTF \l_tmpa_str {
3192         \exp_args:Nx \cs_if_eq:NNTF {
3193           \tl_head:N \l_tmpa_tl
3194         } \stex_invoke_symbol:n {
3195           \exp_args:No \__stex_features_get_symbol_from_cs:n { \use:c { #1 } }
3196         }{
3197           \__stex_features_get_symbol_from_string:n { #1 }
3198         }
3199       }
3200     }
3201   }
3202 }
```

```

3198     }
3199   } {
3200     \__stex_features_get_symbol_from_string:n { #1 }
3201   }
3202   ){
3203     % argument is not a command name
3204     \__stex_features_get_symbol_from_string:n { #1 }
3205     % \l_stex_all_symbols_seq
3206   }
3207 }
3208 }
3209
3210 \cs_new_protected:Nn \__stex_features_get_symbol_from_string:n {
3211   \str_set:Nn \l_tmpa_str { #1 }
3212   \bool_set_false:N \l_tmpa_bool
3213   \bool_if:NF \l_tmpa_bool {
3214     \tl_set:Nn \l_tmpa_tl {
3215       \msg_set:nnn{stex}{error/unknownsymbol}{
3216         No~symbol~#1~found!
3217       }
3218       \msg_error:nn{stex}{error/unknownsymbol}
3219     }
3220     \str_set:Nn \l_tmpa_str { #1 }
3221     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3222     \seq_map_inline:Nn \l__stex_features_copymodule_fields_seq {
3223       \str_set:Nn \l_tmpb_str { ##1 }
3224       \str_if_eq:eeT { \l_tmpa_str } {
3225         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3226       } {
3227         \seq_map_break:n {
3228           \tl_set:Nn \l_tmpa_tl {
3229             \str_set:Nn \l_stex_get_symbol_uri_str {
3230               ##1
3231             }
3232             \__stex_features_get_symbol_check:
3233           }
3234         }
3235       }
3236     }
3237     \l_tmpa_tl
3238   }
3239 }
3240
3241 \cs_new_protected:Nn \__stex_features_get_symbol_from_cs:n {
3242   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3243   { \tl_tail:N \l_tmpa_tl }
3244   \tl_if_single:NTF \l_tmpa_tl {
3245     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3246       \exp_after:wN \str_set:Nn \exp_after:wN
3247       \l_stex_get_symbol_uri_str \l_tmpa_tl
3248       \__stex_features_get_symbol_check:
3249     }{
3250       % TODO
3251       % tail is not a single group

```

```

3252     }
3253 }{
3254     % TODO
3255     % tail is not a single group
3256 }
3257 }
3258
3259 \cs_new_protected:Nn \__stex_features_get_symbol_check: {
3260     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq {?} \l_stex_get_symbol_uri_str
3261     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 3 {
3262         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
3263         \str_set:Nx \l_tmpa_str {\seq_use:Nn \l_tmpa_seq ?}
3264         \seq_if_in:Nof \l__stex_features_copymodule_modules_seq \l_tmpa_str {
3265             \msg_error:nxxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3266                 \l_stex_current_copymodule_name_str\Allowed:~\seq_use:Nn \l__stex_features_copymodule_modules_seq \l_tmpa_str
3267             }
3268         }
3269     }{
3270         \msg_error:nxxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3271             \l_stex_current_copymodule_name_str~(inexplicably)
3272         }
3273     }
3274 }
3275
3276 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
3277     \stex_import_module_uri:nn { #1 } { #2 }
3278     \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3279     \stex_import_require_module:nnnn
3280     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3281     { \l_stex_import_path_str } { \l_stex_import_name_str }
3282     \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3283     \seq_set_eq:NN \l__stex_features_copymodule_modules_seq \l_stex_collect_imports_seq
3284     \seq_clear:N \l__stex_features_copymodule_fields_seq
3285     \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3286         \seq_map_inline:cn {c_stex_module_###1_constants}{
3287             \exp_args:NNx \seq_put_right:Nn \l__stex_features_copymodule_fields_seq {
3288                 ###1 ? #####1
3289             }
3290         }
3291     }
3292     \seq_clear:N \l_tmpa_seq
3293     \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3294         name      = \l_stex_current_copymodule_name_str ,
3295         module    = \l_stex_current_module_str ,
3296         from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3297         includes  = \l_tmpa_seq ,
3298         fields    = \l_tmpa_seq
3299     }
3300     \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3301         as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3302     \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_features_copymodule_modules_seq
3303     \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_features_copymodule_fields_seq {,~}
3304     \stex_if_smsmode:F {
3305         \begin{stex_annotate_env} {#4} {

```

```

3306     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3307   }
3308   \stex_annotate_invisible:nnn{from}{\l_stex_import_ns_str ?\l_stex_import_name_str}{\}
3309 }
3310 \bool_set_eq:NN \l__stex_features_oldhtml_bool \l_stex_html_do_output_bool
3311 \bool_set_false:N \l_stex_html_do_output_bool
3312 }
3313 \cs_new_protected:Nn \stex_copymodule_end:n {
3314   \def \l_tmpa_cs ##1 ##2 {#1}
3315   \bool_set_eq:NN \l_stex_html_do_output_bool \l__stex_features_oldhtml_bool
3316   \tl_clear:N \l_tmpa_tl
3317   \tl_clear:N \l_tmpb_tl
3318   \prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3319   \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3320     \seq_map_inline:cn {c_stex_module_##1_constants}{
3321       \tl_clear:N \l_tmpc_tl
3322       \l_tmpa_cs{##1}{####1}
3323       \str_if_exist:cTF {l__stex_features_copymodule_##1?####1_name_str} {
3324         \tl_put_right:Nx \l_tmpa_tl {
3325           \prop_set_from_keyval:cn {
3326             l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_name_str}
3327           }{
3328             \exp_after:wN \prop_to_keyval:N \csname
3329               l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_name_str}
3330             \endcsname
3331           }
3332           \seq_clear:c {
3333             l_stex_symdecl_
3334             \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_name_str}
3335             _notations
3336           }
3337         }
3338         \tl_put_right:Nx \l_tmpc_tl {
3339           \stex_copy_notations:nn {\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_name_str}}
3340           \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_features_copymodule_##1?####1_name_str}}
3341         }
3342         \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_name_str}}
3343         \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3344           \tl_put_right:Nx \l_tmpc_tl {
3345             \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?####1_name_str}}
3346           }
3347           \tl_put_right:Nx \l_tmpa_tl {
3348             \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3349               \stex_invoke_symbol:n {
3350                 \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_name_str}
3351               }
3352             }
3353           }
3354         }
3355       }{
3356         \tl_put_right:Nx \l_tmpc_tl {
3357           \stex_copy_notations:nn {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str}
3358         }
3359         \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1_prop}

```

```

3360 \prop_put:Nnx \l_tmpa_prop { name }{ \l_stex_current_copymodule_name_str / ####1 }
3361 \prop_put:Nnx \l_tmpa_prop { module }{ \l_stex_current_module_str }
3362 \tl_put_right:Nx \l_tmpa_tl {
3363   \prop_set_from_keyval:cn {
3364     l_stex_symdecl_\l_stex_current_module_str ? \l_stex_current_copymodule_name_str
3365   }{
3366     \prop_to_keyval:N \l_tmpa_prop
3367   }
3368   \seq_clear:c {
3369     l_stex_symdecl_
3370     \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3371     _notations
3372   }
3373 }
3374 \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str}
3375 \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3376   \tl_put_right:Nx \l_tmpc_tl {
3377     \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}
3378   }
3379   \tl_put_right:Nx \l_tmpa_tl {
3380     \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3381       \stex_invoke_symbol:n {
3382         \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3383       }
3384     }
3385   }
3386 }
3387 }
3388 \tl_if_exist:cT {l__stex_features_copymodule_##1?####1_def_tl}{
3389   \tl_put_right:Nx \l_tmpc_tl {
3390     \stex_annotate_invisible:nnn{definiens}{\use:c{l__stex_features_copymodule_##1?####1_def_tl}}
3391   }
3392 }
3393 \tl_put_right:Nx \l_tmpb_tl {
3394   \stex_annotate:nnn{assignment} {##1?####1} { \l_tmpc_tl }
3395 }
3396 }
3397 }
3398 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3399 \tl_put_left:Nx \l_tmpa_tl {
3400   \prop_set_from_keyval:cn {
3401     l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _prop
3402   }{
3403     \prop_to_keyval:N \l_stex_current_copymodule_prop
3404   }
3405 }
3406 \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl
3407 \stex_debug:nn{copymodule}{result:\meaning \l_tmpa_tl}
3408 \exp_args:Nx \stex_do_aftergroup:n {
3409   \exp_args:No \exp_not:n \l_tmpa_tl
3410 }
3411 \l_tmpb_tl
3412 \stex_if_smsmode:F {
3413   \end{stex_annotate_env}

```

```

3414 }
3415 }
3416
3417 \NewDocumentEnvironment {copymodule} { 0{} m m}{
3418   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ structure }
3419   \stex_deactivate_macro:Nn \symdecl {module~environments}
3420   \stex_deactivate_macro:Nn \symdef {module~environments}
3421   \stex_deactivate_macro:Nn \notation {module~environments}
3422   \stex_reactivate_macro:N \assign
3423   \stex_reactivate_macro:N \renamedec1
3424   \stex_reactivate_macro:N \donotcopy
3425   \stex_smsmode_do:
3426 }{
3427   \stex_copymodule_end:n {}
3428 }
3429
3430 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
3431   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ realization }
3432   \stex_deactivate_macro:Nn \symdecl {module~environments}
3433   \stex_deactivate_macro:Nn \symdef {module~environments}
3434   \stex_deactivate_macro:Nn \notation {module~environments}
3435   \stex_reactivate_macro:N \assign
3436   \stex_reactivate_macro:N \renamedec1
3437   \stex_reactivate_macro:N \donotcopy
3438   \stex_smsmode_do:
3439 }{
3440   \stex_copymodule_end:n {
3441     \tl_if_exist:cF {
3442       l__stex_features_copymodule_##1?##2_def_tl
3443     }{
3444       \msg_error:nnxx{stex}{error/interpretmodule/nodedefiniens}{
3445         ##1?##2
3446       }{\l_stex_current_copymodule_name_str}
3447     }
3448   }
3449 }
3450
3451 \NewDocumentCommand \donotcopy { 0{} m}{
3452   \stex_import_module_uri:nn { #1 } { #2 }
3453   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3454   \seq_map_inline:Nn \l_stex_collect_imports_seq {
3455     \seq_remove_all:Nn \l_stex_features_copymodule_modules_seq { ##1 }
3456     \seq_map_inline:cn {c_stex_module_##1_constants}{
3457       \seq_remove_all:Nn \l_stex_features_copymodule_fields_seq { ##1 ? #####1 }
3458       \bool_lazy_any_p:nT {
3459         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_name_str}}
3460         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_macroname_str}}
3461         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_def_tl}}
3462       }{
3463         % TODO throw error
3464       }
3465     }
3466   }
3467 }

```

```

3468 \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
3469 \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_ns_str ?\l_stex_import_name_str }
3470 \prop_put:Nnx \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
3471 }
3472
3473 \NewDocumentCommand \assign { m m }{
3474   \stex_get_symbol_in_copymodule:n {#1}
3475   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
3476   \tl_set:cn {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _def_tl}{#2}
3477 }
3478
3479 \keys_define:nn { stex / renamedec1 } {
3480   name .str_set_x:N = \l_stex_renamedec1_name_str
3481 }
3482 \cs_new_protected:Nn \__stex_features_renamedec1_args:n {
3483   \str_clear:N \l_stex_renamedec1_name_str
3484
3485   \keys_set:nn { stex / renamedec1 } { #1 }
3486 }
3487
3488 \NewDocumentCommand \renamedec1 { 0{} m m }{
3489   \__stex_features_renamedec1_args:n { #1 }
3490   \stex_get_symbol_in_copymodule:n {#2}
3491   \stex_debug:nn{renamedec1}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
3492   \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
3493   \str_if_empty:NTF \l_stex_renamedec1_name_str {
3494     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3495       \l_stex_get_symbol_uri_str
3496     } }
3497   } {
3498     \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex_r
3499     \stex_debug:nn{renamedec1}{@~\l_stex_current_module_str ? \l_stex_renamedec1_name_str}
3500     \prop_set_eq:cc {l_stex_symdec1_
3501       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3502     _prop
3503     }{l_stex_symdec1_ \l_stex_get_symbol_uri_str _prop}
3504     \seq_set_eq:cc {l_stex_symdec1_
3505       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3506     _notations
3507     }{l_stex_symdec1_ \l_stex_get_symbol_uri_str _notations}
3508     \prop_put:cnx {l_stex_symdec1_
3509       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3510     _prop
3511     }{ name }{ \l_stex_renamedec1_name_str }
3512     \prop_put:cnx {l_stex_symdec1_
3513       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3514     _prop
3515     }{ module }{ \l_stex_current_module_str }
3516     \exp_args:NNx \seq_put_left:Nn \l__stex_features_copymodule_fields_seq {
3517       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3518     }
3519     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3520       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3521     } }

```

```

3522 }
3523 }
3524 %\NewDocumentCommand \notation_in_copymodules: { 0{} m } {
3525 % \stex_notation_args:n { #1 }
3526 % \tl_clear:N \l_stex_symdecl_definiens_tl
3527 % \stex_get_symbol_in_copymodule:n { #2 }
3528 % \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
3529 % % todo
3530 %}
3531 \stex_deactivate_macro:Nn \assign {copymodules}
3532 \stex_deactivate_macro:Nn \renamedekl {copymodules}
3533 \stex_deactivate_macro:Nn \donotcopy {copymodules}
3534
3535
3536 \seq_new:N \l_stex_implicit_morphisms_seq
3537 \NewDocumentCommand \implicitmorphism { 0{} m m }{
3538 \stex_import_module_uri:nn { #1 } { #2 }
3539 \stex_debug:nn{implicits}{
3540 Implicit~morphism:~
3541 \l_stex_module_ns_str ? \l__stex_features_name_str
3542 }
3543 \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
3544 \l_stex_module_ns_str ? \l__stex_features_name_str
3545 }{
3546 \msg_error:nnn{stex}{error/conflictingmodules}{
3547 \l_stex_module_ns_str ? \l__stex_features_name_str
3548 }
3549 }
3550
3551 % TODO
3552
3553
3554
3555 \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
3556 \l_stex_module_ns_str ? \l__stex_features_name_str
3557 }
3558 }
3559

```

## 32.2 The feature environment

structural@feature

```

3560
3561 \NewDocumentEnvironment{structural@feature}{ m m m }{
3562 \stex_if_in_module:F {
3563 \msg_set:nnn{stex}{error/nomodule}{
3564 Structural~Feature~has~to~occur~in~a~module:\\
3565 Feature~#2~of~type~#1\\
3566 In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
3567 }
3568 \msg_error:nn{stex}{error/nomodule}
3569 }
3570

```



```

3571 \str_set:Nx \l_stex_module_name_str {
3572   \prop_item:Nn \l_stex_current_module_prop
3573     { name } / #2 - feature
3574 }
3575
3576 \str_set:Nx \l_stex_module_ns_str {
3577   \prop_item:Nn \l_stex_current_module_prop
3578     { ns }
3579 }
3580
3581
3582 \str_clear:N \l_tmpa_str
3583 \seq_clear:N \l_tmpa_seq
3584 \tl_clear:N \l_tmpa_tl
3585 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
3586   origname = #2,
3587   name      = \l_stex_module_name_str ,
3588   ns        = \l_stex_module_ns_str ,
3589   imports   = \exp_not:o { \l_tmpa_seq } ,
3590   constants = \exp_not:o { \l_tmpa_seq } ,
3591   content   = \exp_not:o { \l_tmpa_tl } ,
3592   file      = \exp_not:o { \g_stex_currentfile_seq } ,
3593   lang      = \l_stex_module_lang_str ,
3594   sig       = \l_tmpa_str ,
3595   meta      = \l_tmpa_str ,
3596   feature   = #1 ,
3597 }
3598
3599 \stex_if_smsmode:F {
3600   \begin{stex_annotate_env}{ feature:#1 }{}
3601   \stex_annotate_invisible:nnn{header}{}{ #3 }
3602 }
3603 }{
3604   \str_set:Nx \l_tmpa_str {
3605     c_stex_feature_
3606     \prop_item:Nn \l_stex_current_module_prop { ns } ?
3607     \prop_item:Nn \l_stex_current_module_prop { name }
3608     _prop
3609   }
3610   \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
3611   \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
3612   \stex_if_smsmode:TF {
3613     \exp_args:Nx \stex_add_to_sms:n {
3614       \prop_gset_from_keyval:cn {
3615         c_stex_feature_
3616         \prop_item:Nn \l_stex_current_module_prop { ns } ?
3617         \prop_item:Nn \l_stex_current_module_prop { name }
3618         _prop
3619       } {
3620         origname = #2,
3621         name      = \prop_item:cn { \l_tmpa_str } { name } ,
3622         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
3623         imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
3624         constants = \prop_item:cn { \l_tmpa_str } { constants } ,

```

```

3625         content    = \prop_item:cn { \l_tmpa_str } { content } ,
3626         file        = \prop_item:cn { \l_tmpa_str } { file } ,
3627         lang         = \prop_item:cn { \l_tmpa_str } { lang } ,
3628         sig          = \prop_item:cn { \l_tmpa_str } { sig } ,
3629         meta         = \prop_item:cn { \l_tmpa_str } { meta } ,
3630         feature      = \prop_item:cn { \l_tmpa_str } { feature }
3631     }
3632 }
3633 } {
3634     \end{stex_annotate_env}
3635 }
3636 }
3637

```

## 32.3 Features

structure

```

3638
3639 \prop_new:N \l_stex_all_structures_prop
3640
3641 \keys_define:nn { stex / features / structure } {
3642     name          .str_set_x:N = \l__stex_features_structure_name_str ,
3643 }
3644
3645 \cs_new_protected:Nn \__stex_features_structure_args:n {
3646     \str_clear:N \l__stex_features_structure_name_str
3647     \keys_set:nn { stex / features / structure } { #1 }
3648 }
3649
3650 %\stex_new_feature:nnnn { structure } { 0{ } m } {
3651 %   \__stex_features_structure_args:n { ##1 }
3652 %   \str_if_empty:NT \l__stex_features_structure_name_str {
3653 %       \str_set:Nx \l__stex_features_structure_name_str { ##2 }
3654 %   }
3655 %} {
3656 %
3657 %}
3658
3659 \NewDocumentEnvironment{mathstructure}{0{ } m }{
3660     \__stex_features_structure_args:n { #1 }
3661     \str_if_empty:NT \l__stex_features_structure_name_str {
3662         \str_set:Nx \l__stex_features_structure_name_str { #2 }
3663     }
3664     \exp_args:Nnnx
3665     \begin{structural@feature}{ structure }
3666         { \l__stex_features_structure_name_str }{}
3667         \seq_clear:N \l_tmpa_seq
3668         \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
3669         \stex_smsmode_do:
3670     }{
3671         \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
3672         \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
3673         \str_set:Nx \l_tmpa_str {

```

```

3674 \prop_item:Nn \l_stex_current_module_prop { ns } ?
3675 \prop_item:Nn \l_stex_current_module_prop { name }
3676 }
3677 \seq_map_inline:Nn \l_tmpa_seq {
3678 \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
3679 }
3680 \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
3681 \exp_args:Nnx
3682 \AddToHookNext { env / mathstructure / after }{
3683 \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}}{
3684 \_stex_term_math_oms:nxxx { \l_tmpa_str }{}{0}{}{
3685 }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
3686 \STEXexport {
3687 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3688 {\prop_item:Nn \l_stex_current_module_prop { origname }}
3689 {\l_tmpa_str}
3690 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3691 {#2}{\l_tmpa_str}
3692 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3693 % \prop_item:Nn \l_stex_current_module_prop { origname },
3694 % \l_tmpa_str
3695 % }
3696 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3697 % #2,\l_tmpa_str
3698 % }
3699 % \tl_set:cx { #2 } {
3700 % \stex_invoke_structure:n { \l_tmpa_str }
3701 }
3702 }
3703
3704 \end{structural@feature}
3705 % \g_stex_last_feature_prop
3706 }

```

\instantiate

```

3707 \seq_new:N \l__stex_features_structure_field_seq
3708 \str_new:N \l__stex_features_structure_field_str
3709 \str_new:N \l__stex_features_structure_def_tl
3710 \prop_new:N \l__stex_features_structure_prop
3711 \NewDocumentCommand \instantiate { m O{} m }{
3712 \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
3713 \prop_set_eq:Nc \l__stex_features_structure_prop {
3714 c_stex_feature_\l_tmpa_str _prop
3715 }
3716 \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
3717 \seq_map_inline:Nn \l__stex_features_structure_field_seq {
3718 \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
3719 \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3720 \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
3721 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
3722 {} \l_tmpa_tl
3723 \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3724 \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
3725 \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl

```

```

3726     \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3727   }{
3728     \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
3729     \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3730     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
3731       \l_tmpa_tl
3732     \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3733       \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
3734       \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3735     }{
3736       \tl_clear:N \l_tmpb_tl
3737     }
3738   }
3739 }{
3740   \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
3741   \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3742     \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
3743     \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
3744     \tl_clear:N \l_tmpa_tl
3745   }{
3746     % TODO throw error
3747   }
3748 }
3749 % \l_tmpa_str: name
3750 % \l_tmpa_tl: definiens
3751 % \l_tmpb_tl: notation
3752 \tl_if_empty:NT \l__stex_features_structure_field_str {
3753   % TODO throw error
3754 }
3755 \str_clear:N \l_tmpb_str
3756
3757 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3758 \seq_map_inline:Nn \l_tmpa_seq {
3759   \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
3760   \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
3761   \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
3762     \seq_map_break:n {
3763       \str_set:Nn \l_tmpb_str { ####1 }
3764     }
3765   }
3766 }
3767 \prop_get:cnN { l_stex_symdecl_ \l_tmpb_str _prop } {args}
3768   \l_tmpb_str
3769
3770 \tl_if_empty:NTF \l_tmpb_tl {
3771   \tl_if_empty:NF \l_tmpa_tl {
3772     \exp_args:Nx \use:n {
3773       \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fe
3774     }
3775   }
3776 }{
3777   \tl_if_empty:NTF \l_tmpa_tl {
3778     \exp_args:Nx \use:n {
3779       \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\

```

```

3780     }
3781
3782   }{
3783     \exp_args:Nx \use:n {
3784       \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
3785       \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
3786     }
3787   }
3788 }
3789 % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
3790 % \prop_item:Nn \l_stex_current_module_prop {name} ?
3791 % #3/\l__stex_features_structure_field_str
3792 % \par
3793 % \expandafter\present\csname
3794 %   \l_stex_symdecl_
3795 %   \prop_item:Nn \l_stex_current_module_prop {ns} ?
3796 %   \prop_item:Nn \l_stex_current_module_prop {name} ?
3797 %   #3/\l__stex_features_structure_field_str
3798 %   _prop
3799 % \endcsname
3800 }
3801
3802 \tl_clear:N \l__stex_features_structure_def_tl
3803
3804 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3805 \seq_map_inline:Nn \l_tmpa_seq {
3806   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3807   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3808   \exp_args:Nx \use:n {
3809     \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
3810
3811     }
3812   }
3813
3814   \prop_if_exist:cF {
3815     \l_stex_symdecl_
3816     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3817     \prop_item:Nn \l_stex_current_module_prop {name} ?
3818     #3/\l_tmpa_str
3819     _prop
3820   }{
3821     \prop_get:cnN { \l_stex_symdecl_ ##1 _prop } {args}
3822     \l_tmpb_str
3823     \exp_args:Nx \use:n {
3824       \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
3825     }
3826   }
3827 }
3828
3829 \symdecl*[type={\STEXsymbol{module-type}}{
3830   \_stex_term_math_oms:nnnn {
3831     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3832     \prop_item:Nn \l__stex_features_structure_prop {name}
3833   }-{}{0}-{}

```

```

3834   }}]{#3}
3835
3836   % TODO: -> sms file
3837
3838   \tl_set:cx{ #3 }{
3839     \stex_invoke_structure:nnn {
3840       \prop_item:Nn \l_stex_current_module_prop {ns} ?
3841       \prop_item:Nn \l_stex_current_module_prop {name} ? #3
3842     } {
3843       \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3844       \prop_item:Nn \l__stex_features_structure_prop {name}
3845     }
3846   }
3847   \stex_smsmode_do:
3848 }

```

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

`\stex_invoke_structure:nnn`

```

3849 % #1: URI of the instance
3850 % #2: URI of the instantiated module
3851 \cs_new_protected:Nn \stex_invoke_structure:nnn {
3852   \tl_if_empty:nTF{ #3 }{
3853     \prop_set_eq:Nc \l__stex_features_structure_prop {
3854       c_stex_feature_ #2 _prop
3855     }
3856     \tl_clear:N \l_tmpa_tl
3857     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3858     \seq_map_inline:Nn \l_tmpa_seq {
3859       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3860       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3861       \cs_if_exist:cT {
3862         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
3863       }{
3864         \tl_if_empty:NF \l_tmpa_tl {
3865           \tl_put_right:Nn \l_tmpa_tl {,}
3866         }
3867         \tl_put_right:Nx \l_tmpa_tl {
3868           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
3869         }
3870       }
3871     }
3872     \exp_args:No \mathstrut \l_tmpa_tl
3873   }{
3874     \stex_invoke_symbol:n{#1/#3}
3875   }
3876 }

```

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

3877 `\</package>`

## Chapter 33

# STEX -Statements Implementation

```
3878 <*package>
3879
3880 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3881
3882 \protected\def\ignorespacesandpars{
3883   \begingroup\catcode13=10\relax
3884   \@ifnextchar\par{
3885     \endgroup\expandafter\ignorespacesandpars\@gobble
3886   }{
3887     \endgroup
3888   }
3889 }
3890
3891 <@@=stex_statements>
3892
3893   Warnings and error messages
```

\titleemph

```
3893 \def\titleemph#1{\textbf{#1}}
```

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

### 33.1 Definitions

definiendum

```
3894 \keys_define:nn {stex / definiendum }{
3895   post      .tl_set:N      = \l__stex_statements_definiendum_post_tl,
3896   root      .str_set_x:N   = \l__stex_statements_definiendum_root_str,
3897   gfa       .str_set_x:N   = \l__stex_statements_definiendum_gfa_str
3898 }
3899 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
3900   \str_clear:N \l__stex_statements_definiendum_root_str
3901   \tl_clear:N \l__stex_statements_definiendum_post_tl
3902   \str_clear:N \l__stex_statements_definiendum_gfa_str
```

```

3903 \keys_set:nn { stex / definiendum } { #1 }
3904 }
3905 \NewDocumentCommand \definiendum { 0{} m m } {
3906   \__stex_statements_definiendum_args:n { #1 }
3907   \stex_get_symbol:n { #2 }
3908   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3909   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
3910     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
3911       \tl_set:Nn \l_tmpa_tl { #3 }
3912     } {
3913       \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
3914       \tl_set:Nn \l_tmpa_tl {
3915         \l__stex_statements_definiendum_root_str\l__stex_statements_definiendum_post_tl
3916       }
3917     }
3918   } {
3919     \tl_set:Nn \l_tmpa_tl { #3 }
3920   }
3921
3922   % TODO root
3923   \rustex_if:TF {
3924     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
3925   } {
3926     \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
3927   }
3928 }
3929 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

## definame

```

3930
3931 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3932
3933 \NewDocumentCommand \definame { 0{} m } {
3934   \__stex_statements_definiendum_args:n { #1 }
3935   % TODO: root
3936   \stex_get_symbol:n { #2 }
3937   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3938   \str_set:Nx \l_tmpa_str {
3939     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3940   }
3941   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3942   \rustex_if:TF {
3943     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3944       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3945     }
3946   } {
3947     \defemph@uri {
3948       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3949     } { \l_stex_get_symbol_uri_str }
3950   }
3951 }
3952 \stex_deactivate_macro:Nn \definame {definition~environments}

```



```

3953
3954 \NewDocumentCommand \Definame { 0{ } m } {
3955   \_stex_statements_definiendum_args:n { #1 }
3956   \stex_get_symbol:n { #2 }
3957   \str_set:Nx \l_tmpa_str {
3958     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3959   }
3960   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3961   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3962   \rustex_if:TF {
3963     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3964       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3965     }
3966   } {
3967     \defemph@uri {
3968       \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
3969     } { \l_stex_get_symbol_uri_str }
3970   }
3971 }
3972 \stex_deactivate_macro:Nn \Definame {definition-environments}
3973
3974 \NewDocumentCommand \Symname { 0{ } m }{
3975   \stex_symname_args:n { #1 }
3976   \stex_get_symbol:n { #2 }
3977   \str_set:Nx \l_tmpa_str {
3978     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3979   }
3980   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3981   \let\compemph_uri_prev:\compemph@uri
3982   \let\compemph@uri\symrefemph@uri
3983   \exp_args:NNx \use:nn
3984   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }! [
3985     \exp_after:wN \stex_capitalize:n \l_tmpa_str
3986     \l_stex_symname_post_str
3987   ] }
3988   \let\compemph@uri\compemph_uri_prev:
3989 }

```

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

## sdefinition

```

3990
3991 \keys_define:nn {stex / sdefinition }{
3992   type      .str_set_x:N = \sdefinitiontype,
3993   id        .str_set_x:N = \sdefinitionid,
3994   name      .str_set_x:N = \sdefinitionname,
3995   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
3996   title     .tl_set:N     = \sdefinitiontitle
3997 }
3998 \cs_new_protected:Nn \_stex_statements_sdefinition_args:n {
3999   \str_clear:N \sdefinitiontype
4000   \str_clear:N \sdefinitionid
4001   \str_clear:N \sdefinitionname
4002   \clist_clear:N \l__stex_statements_sdefinition_for_clist

```

```

4003 \tl_clear:N \sdefinitiontitle
4004 \keys_set:nn { stex / sdefinition }{ #1 }
4005 }
4006
4007 \NewDocumentEnvironment{sdefinition}{0{}}{
4008   \__stex_statements_sdefinition_args:n{ #1 }
4009   \stex_reactivate_macro:N \definiendum
4010   \stex_reactivate_macro:N \definame
4011   \stex_reactivate_macro:N \Definame
4012   \stex_if_smsmode:F{
4013     \seq_clear:N \l_tmpa_seq
4014     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4015       \str_if_eq:nnF{ ##1 }{ }{
4016         \stex_get_symbol:n { ##1 }
4017         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4018           \l_stex_get_symbol_uri_str
4019         }
4020       }
4021     }
4022     \exp_args:Nnnx
4023     \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {,}}
4024     \str_if_empty:NF \sdefinitiontype {
4025       \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{ }
4026     }
4027     \clist_set:Nn \l_tmpa_clist \sdefinitiontype
4028     \tl_clear:N \l_tmpa_tl
4029     \clist_map_inline:Nn \l_tmpa_clist {
4030       \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
4031         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
4032       }
4033     }
4034     \tl_if_empty:NTF \l_tmpa_tl {
4035       \__stex_statements_sdefinition_start:
4036     }{
4037       \l_tmpa_tl
4038     }
4039   }
4040   \stex_ref_new_doc_target:n \sdefinitionid
4041   \stex_smsmode_do:
4042 }{
4043   \str_if_empty:NF \sdefinitionname { \symdecl*{\sdefinitionname} }
4044   \stex_if_smsmode:F {
4045     \clist_set:Nn \l_tmpa_clist \sdefinitiontype
4046     \tl_clear:N \l_tmpa_tl
4047     \clist_map_inline:Nn \l_tmpa_clist {
4048       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
4049         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
4050       }
4051     }
4052     \tl_if_empty:NTF \l_tmpa_tl {
4053       \__stex_statements_sdefinition_end:
4054     }{
4055       \l_tmpa_tl
4056     }

```

```

4057     \end{stex_annotate_env}
4058   }
4059 }

```

\stexpatchdefinition

```

4060 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
4061   \par\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
4062     ~(\sdefinitiontitle)
4063   }~}
4064 }
4065 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\par\medskip}
4066
4067 \newcommand\stexpatchdefinition[3] [] {
4068   \str_set:Nx \l_tmpa_str{ #1 }
4069   \str_if_empty:NTF \l_tmpa_str {
4070     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
4071     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
4072   }{
4073     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
4074     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
4075   }
4076 }

```

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

\inlinedef inline:

```

4077 \keys_define:nn {stex / inlinedef }{
4078   type      .str_set_x:N = \sdefinitiontype,
4079   id        .str_set_x:N = \sdefinitionid,
4080   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4081   name      .str_set_x:N = \sdefinitionname
4082 }
4083 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
4084   \str_clear:N \sdefinitiontype
4085   \str_clear:N \sdefinitionid
4086   \str_clear:N \sdefinitionname
4087   \clist_clear:N \l__stex_statements_sdefinition_for_clist
4088   \keys_set:nn { stex / inlinedef }{ #1 }
4089 }
4090 \NewDocumentCommand \inlinedef { 0{} m } {
4091   \begingroup
4092   \__stex_statements_inlinedef_args:n{ #1 }
4093   \stex_ref_new_doc_target:n \sdefinitionid
4094   \stex_reactivate_macro:N \definiendum
4095   \stex_reactivate_macro:N \definame
4096   \stex_reactivate_macro:N \Definame
4097   \stex_if_smsmode:TF{
4098     \str_if_empty:NF \sdefinitionname { \symdecl*{\sdefinitionname} }
4099   }{
4100     \seq_clear:N \l_tmpa_seq
4101     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4102       \str_if_eq:nnF{ ##1 }{}{
4103         \stex_get_symbol:n { ##1 }
4104         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {

```

```

4105         \l_stex_get_symbol_uri_str
4106     }
4107 }
4108 }
4109 \exp_args:Nnx
4110 \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
4111     \str_if_empty:NF \sdefinitiontype {
4112         \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{ }
4113     }
4114     #2
4115     \str_if_empty:NF \sdefinitionname { \symdecl*{\sdefinitionname} }
4116 }
4117 }
4118 \endgroup
4119 \stex_smsmode_do:
4120 }

```

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

## 33.2 Assertions

**sassertion**

```

4121
4122 \keys_define:nn {stex / sassertion }{
4123     type      .str_set_x:N = \sassertiontype,
4124     id        .str_set_x:N = \sassertionid,
4125     title     .tl_set:N    = \sassertiontitle ,
4126     for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4127     name      .str_set_x:N = \sassertionname
4128 }
4129 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
4130     \str_clear:N \sassertiontype
4131     \str_clear:N \sassertionid
4132     \str_clear:N \sassertionname
4133     \clist_clear:N \l__stex_statements_sassertion_for_clist
4134     \tl_clear:N \sassertiontitle
4135     \keys_set:nn { stex / sassertion }{ #1 }
4136 }
4137
4138 %\tl_new:N \g__stex_statements_aftergroup_tl
4139
4140 \NewDocumentEnvironment{sassertion}{0{}}{
4141     \__stex_statements_sassertion_args:n{ #1 }
4142     \stex_if_smsmode:F {
4143         \seq_clear:N \l_tmpa_seq
4144         \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4145             \str_if_eq:nnF{ ##1 }{ }{
4146                 \stex_get_symbol:n { ##1 }
4147                 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4148                     \l_stex_get_symbol_uri_str
4149                 }
4150             }
4151         }

```

```

4152 \exp_args:Nnnx
4153 \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}
4154 \str_if_empty:NF \sassertiontype {
4155   \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
4156 }
4157 \clist_set:No \l_tmpa_clist \sassertiontype
4158 \tl_clear:N \l_tmpa_tl
4159 \clist_map_inline:Nn \l_tmpa_clist {
4160   \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
4161     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
4162   }
4163 }
4164 \tl_if_empty:NTF \l_tmpa_tl {
4165   \__stex_statements_sassertion_start:
4166 }{
4167   \l_tmpa_tl
4168 }
4169 }
4170 \stex_ref_new_doc_target:n \sassertionid
4171 \stex_smsmode_do:
4172 ){
4173   \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
4174   \stex_if_smsmode:F {
4175     \clist_set:No \l_tmpa_clist \sassertiontype
4176     \tl_clear:N \l_tmpa_tl
4177     \clist_map_inline:Nn \l_tmpa_clist {
4178       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
4179         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
4180       }
4181     }
4182     \tl_if_empty:NTF \l_tmpa_tl {
4183       \__stex_statements_sassertion_end:
4184     }{
4185       \l_tmpa_tl
4186     }
4187   }
4188 }
4189 }

```

\stexpatchassertion

```

4190
4191 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
4192   \par\noindent\titllemph{Assertion~\tl_if_empty:NF \sassertiontitle {
4193     (\sassertiontitle)
4194   }~}
4195 }
4196 \cs_new_protected:Nn \__stex_statements_sassertion_end: { \par\medskip}
4197
4198 \newcommand\stexpatchassertion[3] [] {
4199   \str_set:Nx \l_tmpa_str{ #1 }
4200   \str_if_empty:NTF \l_tmpa_str {
4201     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
4202     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
4203   }{

```

```

4204     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
4205     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
4206   }
4207 }

```

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

`\inlineass` inline:

```

4208 \keys_define:nn {stex / inlineass }{
4209   type      .str_set_x:N = \sassertiontype,
4210   id        .str_set_x:N = \sassertionid,
4211   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4212   name      .str_set_x:N = \sassertionname
4213 }
4214 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
4215   \str_clear:N \sassertiontype
4216   \str_clear:N \sassertionid
4217   \str_clear:N \sassertionname
4218   \clist_clear:N \l__stex_statements_sassertion_for_clist
4219   \keys_set:nn { stex / inlineass }{ #1 }
4220 }
4221 \NewDocumentCommand \inlineass { 0{} m } {
4222   \begingroup
4223   \__stex_statements_inlineass_args:n{ #1 }
4224   \stex_ref_new_doc_target:n \sassertionid
4225   \stex_if_smsmode:TF{
4226     \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
4227   }{
4228     \seq_clear:N \l_tmpa_seq
4229     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4230       \str_if_eq:nnF{ ##1 }{ }{
4231         \stex_get_symbol:n { ##1 }
4232         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4233           \l_stex_get_symbol_uri_str
4234         }
4235       }
4236     }
4237     \exp_args:Nnx
4238     \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{
4239       \str_if_empty:NF \sassertiontype {
4240         \stex_annotate_invisible:nnn{type}{\sassertiontype}{ }
4241       }
4242       #2
4243       \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
4244     }
4245   }
4246   \endgroup
4247   \stex_smsmode_do:
4248 }

```

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

## 33.3 Examples

sexample

```

4249
4250 \keys_define:nn {stex / sexample }{
4251   type      .str_set_x:N = \exampletype,
4252   id        .str_set_x:N = \sexampleid,
4253   title     .tl_set:N     = \sexampletile,
4254   for       .clist_set:N  = \l__stex_statements_sexample_for_clist,
4255 }
4256 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
4257   \str_clear:N \sexampletype
4258   \str_clear:N \sexampleid
4259   \tl_clear:N \sexampletile
4260   \clist_clear:N \l__stex_statements_sexample_for_clist
4261   \keys_set:nn { stex / sexample }{ #1 }
4262 }
4263
4264 \NewDocumentEnvironment{sexample}{0{}}{
4265   \__stex_statements_sexample_args:n{ #1 }
4266   \stex_if_smsmode:F {
4267     \seq_clear:N \l_tmpa_seq
4268     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4269       \str_if_eq:nnF{ ##1 }{}{
4270         \stex_get_symbol:n { ##1 }
4271         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4272           \l_stex_get_symbol_uri_str
4273         }
4274       }
4275     }
4276     \exp_args:Nnnx
4277     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
4278     \str_if_empty:NF \sexampletype {
4279       \stex_annotate_invisible:nnn{type}{\sexampletype}{}
4280     }
4281     \clist_set:Nn \l_tmpa_clist \sexampletype
4282     \tl_clear:N \l_tmpa_tl
4283     \clist_map_inline:Nn \l_tmpa_clist {
4284       \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
4285         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
4286       }
4287     }
4288     \tl_if_empty:NTF \l_tmpa_tl {
4289       \__stex_statements_sexample_start:
4290     }{
4291       \l_tmpa_tl
4292     }
4293   }
4294   \stex_ref_new_doc_target:n \sexampleid
4295   \stex_smsmode_do:
4296 }{
4297   \str_if_empty:NF \sexamplename { \symdecl*{\sexamplename} }
4298   \stex_if_smsmode:F {
4299     \clist_set:Nn \l_tmpa_clist \sexampletype

```

```

4300 \tl_clear:N \l_tmpa_tl
4301 \clist_map_inline:Nn \l_tmpa_clist {
4302   \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
4303     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
4304   }
4305 }
4306 \tl_if_empty:NTF \l_tmpa_tl {
4307   \__stex_statements_sexample_end:
4308 }{
4309   \l_tmpa_tl
4310 }
4311 \end{stex_annotate_env}
4312 }
4313 }

```

\stexpatchexample

```

4314
4315 \cs_new_protected:Nn \__stex_statements_sexample_start: {
4316   \par\noindent\titllemph{Example~\tl_if_empty:NF \sexampltitle {
4317     (\sexampltitle)
4318   }~}
4319 }
4320 \cs_new_protected:Nn \__stex_statements_sexample_end: {\par\medskip}
4321
4322 \newcommand\stexpatchexample[3] [] {
4323   \str_set:Nx \l_tmpa_str{ #1 }
4324   \str_if_empty:NTF \l_tmpa_str {
4325     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
4326     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
4327   }{
4328     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
4329     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
4330   }
4331 }

```

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

\inlineex inline:

```

4332 \keys_define:nn {stex / inlineex }{
4333   type      .str_set_x:N = \sexampltype,
4334   id        .str_set_x:N = \sexampleid,
4335   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
4336   name      .str_set_x:N = \sexamplname
4337 }
4338 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
4339   \str_clear:N \sexampltype
4340   \str_clear:N \sexampleid
4341   \str_clear:N \sexamplname
4342   \clist_clear:N \l__stex_statements_sexample_for_clist
4343   \keys_set:nn { stex / inlineex }{ #1 }
4344 }
4345 \NewDocumentCommand \inlineex { 0{} m } {
4346   \begingroup
4347   \__stex_statements_inlineex_args:n{ #1 }

```



```

4348 \stex_ref_new_doc_target:n \sexampleid
4349 \stex_if_smsmode:TF{
4350   \str_if_empty:NF \sexamplename { \symdecl*{\examplename} }
4351 }{
4352   \seq_clear:N \l_tmpa_seq
4353   \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4354     \str_if_eq:nnF{ ##1 }{ }{
4355       \stex_get_symbol:n { ##1 }
4356       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4357         \l_stex_get_symbol_uri_str
4358       }
4359     }
4360   }
4361   \exp_args:Nnx
4362   \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpa_seq {},}{ }{
4363     \str_if_empty:NF \sexamplename {
4364       \stex_annotate_invisible:nnn{type}{\sexamplename}{ }
4365     }
4366     #2
4367     \str_if_empty:NF \sexamplename { \symdecl*{\sexamplename} }
4368   }
4369 }
4370 \endgroup
4371 \stex_smsmode_do:
4372 }

```

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

## 33.4 Logical Paragraphs

sparagraph

```

4373 \keys_define:nn { stex / sparagraph } {
4374   id      .str_set:N = \sparagraphid ,
4375   title   .tl_set:N  = \l_stex_sparagraph_title_tl ,
4376   type    .str_set:N = \sparagraphtype ,
4377   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
4378   from    .tl_set:N  = \sparagraphfrom ,
4379   to      .tl_set:N  = \sparagraphto ,
4380   start   .tl_set:N  = \l_stex_sparagraph_start_tl ,
4381   name    .str_set:N  = \sparagraphname
4382 }
4383
4384 \cs_new_protected:Nn \stex_sparagraph_args:n {
4385   \tl_clear:N \l_stex_sparagraph_title_tl
4386   \tl_clear:N \sparagraphfrom
4387   \tl_clear:N \sparagraphto
4388   \tl_clear:N \l_stex_sparagraph_start_tl
4389   \str_clear:N \sparagraphid
4390   \str_clear:N \sparagraphtype
4391   \clist_clear:N \l__stex_statements_sparagraph_for_clist
4392   \str_clear:N \sparagraphname
4393   \keys_set:nn { stex / sparagraph }{ #1 }
4394 }

```

```

4395 \newif\if@in@omtext\@in@omtextfalse
4396
4397 \NewDocumentEnvironment {sparagraph} { 0{} } {
4398   \stex_sparagraph_args:n { #1 }
4399   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4400     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
4401   }{
4402     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
4403   }
4404   \@in@omtexttrue
4405   \stex_if_smsmode:F {
4406     \seq_clear:N \l_tmpa_seq
4407     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
4408       \str_if_eq:nnF{ ##1 }{}{
4409         \stex_get_symbol:n { ##1 }
4410         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4411           \l_stex_get_symbol_uri_str
4412         }
4413       }
4414     }
4415     \exp_args:Nnnx
4416     \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
4417     \str_if_empty:NF \sparagraphtype {
4418       \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}
4419     }
4420     \str_if_empty:NF \sparagraphfrom {
4421       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
4422     }
4423     \str_if_empty:NF \sparagraphto {
4424       \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
4425     }
4426     \clist_set:No \l_tmpa_clist \sparagraphtype
4427     \tl_clear:N \l_tmpa_tl
4428     \clist_map_inline:Nn \sparagraphtype {
4429       \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
4430         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
4431       }
4432     }
4433     \tl_if_empty:NTF \l_tmpa_tl {
4434       \__stex_statements_sparagraph_start:
4435     }{
4436       \l_tmpa_tl
4437     }
4438   }
4439   \stex_ref_new_doc_target:n \sparagraphid
4440   \stex_smsmode_do:
4441   \ignorespacesandpars
4442 }{
4443   \stex_if_smsmode:F {
4444     \clist_set:No \l_tmpa_clist \sparagraphtype
4445     \tl_clear:N \l_tmpa_tl
4446     \clist_map_inline:Nn \l_tmpa_clist {
4447       \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
4448         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}

```

```

4449     }
4450   }
4451   \str_if_empty:NF \sparagraphname { \symdecl*\sparagraphname} }
4452   \tl_if_empty:NTF \l_tmpa_tl {
4453     \__stex_statements_sparagraph_end:
4454   }{
4455     \l_tmpa_tl
4456   }
4457   \end{stex_annotate_env}
4458 }
4459 }

```

# \stexpatchparagraph

```

4460
4461 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
4462   \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4463     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
4464       \titleemph{\l_stex_sparagraph_title_tl}:~
4465     }
4466   }{
4467     \titleemph{\l_stex_sparagraph_start_tl}~
4468   }
4469 }
4470 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
4471
4472 \newcommand\stexpatchparagraph[3] [] {
4473   \str_set:Nx \l_tmpa_str{ #1 }
4474   \str_if_empty:NTF \l_tmpa_str {
4475     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
4476     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
4477   }{
4478     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2 }
4479     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
4480   }
4481 }
4482
4483 \keys_define:nn { stex / inlinepara } {
4484   id      .str_set_x:N = \sparagraphid ,
4485   type    .str_set_x:N = \sparagraphtype ,
4486   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
4487   from    .tl_set:N    = \sparagraphfrom ,
4488   to      .tl_set:N    = \sparagraphto ,
4489   name    .str_set:N   = \sparagraphname
4490 }
4491 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
4492   \tl_clear:N \sparagraphfrom
4493   \tl_clear:N \sparagraphto
4494   \str_clear:N \sparagraphid
4495   \str_clear:N \sparagraphtype
4496   \clist_clear:N \l__stex_statements_sparagraph_for_clist
4497   \str_clear:N \sparagraphname
4498   \keys_set:nn { stex / inlinepara }{ #1 }
4499 }
4500 \NewDocumentCommand \inlinepara { 0{} m } {

```

```

4501 \begingroup
4502 \__stex_statements_inlinepara_args:n{ #1 }
4503 \stex_ref_new_doc_target:n \sparagraphid
4504 \stex_if_smsmode:TF{
4505   \str_if_empty:NF \sparagraphname { \symdecl*\sparagraphname } }
4506 }{
4507   \seq_clear:N \l_tmpa_seq
4508   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
4509     \str_if_eq:nnF{ ##1 }{}{
4510       \stex_get_symbol:n { ##1 }
4511       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4512         \l_stex_get_symbol_uri_str
4513       }
4514     }
4515   }
4516   \exp_args:Nnx
4517   \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
4518     \str_if_empty:NF \sparagraphtype {
4519       \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}}
4520   }
4521   \str_if_empty:NF \sparagraphfrom {
4522     \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}}
4523   }
4524   \str_if_empty:NF \sparagraphto {
4525     \stex_annotate_invisible:nnn{to}{\sparagraphto}{}}
4526   }
4527   #2
4528   \str_if_empty:NF \sparagraphname { \symdecl*\sparagraphname } }
4529 }
4530 }
4531 \endgroup
4532 \stex_smsmode_do:
4533 }
4534

```

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

symboldoc

```

4535 \NewDocumentEnvironment{symboldoc}{ m }{
4536   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
4537   \seq_clear:N \l_tmpb_seq
4538   \seq_map_inline:Nn \l_tmpa_seq {
4539     \str_if_eq:nnF{ ##1 }{}{
4540       \stex_get_symbol:n { ##1 }
4541       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
4542         \l_stex_get_symbol_uri_str
4543       }
4544     }
4545   }
4546   \par
4547   \exp_args:Nnnx
4548   \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
4549 }{
4550   \end{stex_annotate_env}
4551 }

```

4552 </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>

```
4553 <*package>
4554 <@@=stex_sproof>
4555
4556 %%%%%%%%%%% sproof.dtx %%%%%%%%%%%
4557
```

### 34.2 Proofs

We first define some keys for the proof environment.

```
4558 \keys_define:nn { stex / spf } {
4559   id          .str_set:N = \l__stex_sproof_spf_id_str,
4560   display     .tl_set:N  = \l__stex_sproof_spf_display_tl,
4561   for         .tl_set:N  = \l__stex_sproof_spf_for_tl ,
4562   from        .tl_set:N  = \l__stex_sproof_spf_from_tl ,
4563   proofend    .tl_set:N  = \l__stex_sproof_spf_proofend_tl,
4564   type        .tl_set:N  = \l__stex_sproof_spf_type_tl,
4565   title       .tl_set:N  = \l__stex_sproof_spf_title_tl,
4566   continues   .tl_set:N  = \l__stex_sproof_spf_continues_tl,
4567   functions   .tl_set:N  = \l__stex_sproof_spf_functions_tl,
4568   method      .tl_set:N  = \l__stex_sproof_spf_method_tl
4569 }
4570 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
4571   \str_clear:N \l__stex_sproof_spf_id_str
4572   \tl_clear:N \l__stex_sproof_spf_display_tl
4573   \tl_clear:N \l__stex_sproof_spf_for_tl
4574   \tl_clear:N \l__stex_sproof_spf_from_tl
4575   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
4576   \tl_clear:N \l__stex_sproof_spf_type_tl
4577   \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

```

4578 \tl_clear:N \l__stex_sproof_spf_continues_tl
4579 \tl_clear:N \l__stex_sproof_spf_functions_tl
4580 \tl_clear:N \l__stex_sproof_spf_method_tl
4581 \keys_set:nn { stex / spf }{ #1 }
4582 }

```

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

```

4583 \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.

```

4584 \newcount\count_ten
4585 \newenvironment{pst@with@label}[1]{
4586   \edef\pst@label{#1}
4587   \advance\count_ten by 1\relax
4588   \count_ten=1
4589 }{
4590   \advance\count_ten by -1\relax
4591 }

```

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

```

4592 \def\the@pst@label{
4593   \pst@make@label\pst@label{\number\count_ten}\l__stex_sproof_pstlabel_postfix_tl
4594 }

```

(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.

```

4595 \keys_define:nn { stex / pstlabel }{
4596   prefix      .tl_set:N   = \l__stex_sproof_pstlabel_prefix_tl,
4597   delimiter   .tl_set:N   = \l__stex_sproof_pstlabel_delimiter_tl,
4598   postfix     .tl_set:N   = \l__stex_sproof_pstlabel_postfix_tl
4599 }
4600 \cs_new_protected:Nn \__stex_sproof_pstlabel_args:n {

```

---

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

```

4601 \tl_set:Nn \l__stex_sproof_pstlabel_prefix_tl {P}
4602 \tl_set:Nn \l__stex_sproof_pstlabel_delimiter_tl {.}
4603 \tl_clear:N \l__stex_sproof_pstlabel_postfix_tl
4604 }
4605 \__stex_sproof_pstlabel_args:n {}
4606 \newcommand\setpstlabelstyle[1]{
4607   \__stex_sproof_pstlabel_args:n {#1}
4608 }
4609 \newcommand\setpstlabelstyledefault{%
4610   \__stex_sproof_pstlabel_args:n{prefix=P,delimiter=.,postfix={}}
4611 }

```

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

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

```

4612 \ExplSyntaxOff
4613 \def\pst@make@label@long#1#2{\@for\@I:=#1\do{\expandafter\expandafter\expandafter\@I\csname
4614 \def\pst@make@label@angles#1#2{\ensuremath{\@for\@I:=#1\do{\rangle}}#2}
4615 \def\pst@make@label@short#1#2{#2}
4616 \def\pst@make@label@empty#1#2{}
4617 \ExplSyntaxOn
4618 \def\pstlabelstyle#1{%
4619   \def\pst@make@label{\use:c{pst@make@label@#1}}%
4620 }%
4621 \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.

```

4622 \def\next@pst@label{%
4623   \global\advance\count\count10 by 1%
4624 }%

```

(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

```

4625 \def\sproof@box{
4626   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
4627 }
4628 \def\spf@proofend{\sproof@box}
4629 \def\sproofend{
4630   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
4631     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
4632   }
4633 }
4634 \def\sProofEndSymbol#1{\def\sproof@box{#1}}

```

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

**spf@\*@kw**

```

4635 \def\spf@proofsketch@kw{Proof Sketch}
4636 \def\spf@proof@kw{Proof}
4637 \def\spf@step@kw{Step}

```



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

For the other languages, we set up triggers

```

4638 \AddToHook{begindocument}{
4639   \ltx@ifpackageloaded{babel}{
4640     \makeatletter
4641     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4642     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4643       \input{sproof-ngerman.ldf}
4644     }
4645     \clist_if_in:NnT \l_tmpa_clist {finnish}{
4646       \input{sproof-finnish.ldf}
4647     }
4648     \clist_if_in:NnT \l_tmpa_clist {french}{
4649       \input{sproof-french.ldf}
4650     }
4651     \clist_if_in:NnT \l_tmpa_clist {russian}{
4652       \input{sproof-russian.ldf}
4653     }
4654     \makeatother
4655   }{}
4656 }

```

`spfsketch`

```

4657 \newcommand\spfsketch[2][]{
4658   \__stex_sproof_spf_args:n{#1}
4659   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4660     \titleemph{
4661       \tl_if_empty:NnT \l__stex_sproof_spf_type_tl {
4662         \spf@proofsketch@kw
4663       }{
4664         \l__stex_sproof_spf_type_tl
4665       }
4666     }:
4667   }
4668   {~#2}
4669   %\sref@label@id{this \ifx\spf@type\@empty\spf@proofsketch@kw\else\spf@type\fi}
4670   \sproofend
4671 }

```

(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>

```

4672 \newenvironment{spfeq}[2][]{
4673   \__stex_sproof_spf_args:n{#1}
4674   %\sref@target
4675   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4676     \titleemph{
4677       \tl_if_empty:NnT \l__stex_sproof_spf_type_tl {
4678         \spf@proof@kw
4679       }{

```

<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

```

4680     \l__stex_sproof_spf_type_tl
4681   }
4682   }:
4683 }
4684 {~#2}
4685 \begin{displaymath}\begin{array}{rcll}
4686 }{
4687   \end{array}\end{displaymath}
4688 }

```

(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.

```

4689 \newenvironment{spf@proof}[2] []{
4690   \l__stex_sproof_spf_args:n{#1}
4691   %\sref@target
4692   \count_ten=10
4693   \par\noindent
4694   \tl_if_eq:NNTF \l__stex_sproof_spf_display_tl\spf@flow{
4695     \titleemph{
4696       \tl_if_empty:NNTF \l__stex_sproof_spf_type_tl {
4697         \spf@proof@kw
4698       }{
4699         \l__stex_sproof_spf_type_tl
4700       }
4701     }:
4702   }
4703   {~#2}
4704   %\sref@label{id{this \ifx\spf@type\@empty\spf@proof@kw\else\spf@type\fi}}
4705   \def\pst@label{}
4706   \newcount\pst@count% initialize the labeling mechanism
4707   \begin{description}\begin{pst@with@label}{\l__stex_sproof_pstlabel_prefix_tl}
4708   }{
4709     \end{pst@with@label}\end{description}
4710   }
4711   \newenvironment{sproof}[2] []{\begin{spf@proof}[#1]{#2}}{\sproofend\end{spf@proof}}
4712   \newenvironment{sProof}[2] []{\begin{spf@proof}[#1]{#2}}{\end{spf@proof}}

```

**\spfidea**

```

4713 \newcommand\spfidea[2] []{
4714   \l__stex_sproof_spf_args:n{#1}
4715   \titleemph{
4716     \tl_if_empty:NNTF \l__stex_sproof_spf_type_tl {Proof~Idea}{
4717       \l__stex_sproof_spf_type_tl
4718     }:
4719   }~#2
4720   \sproofend
4721 }

```

(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
4722 \newenvironment{spfstep}[1][]{
4723   \_stex_sproof_spf_args:n{#1}
4724   \@in@omtexttrue
4725   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4726     \item[\the@pst@label]
4727   }
4728   \tl_if_empty:NF \l__stex_sproof_spf_title_tl {
4729     {(\titleemph{\l__stex_sproof_spf_title_tl})\enspace}
4730   }
4731   %\sref@label@id{\pst@label}
4732   \ignorespacesandpars
4733 }{
4734   \next@pst@label\ignorespacesandpars
4735 }

```

**sproofcomment**

```

4736 \newenvironment{sproofcomment}[1][]{
4737   \_stex_sproof_spf_args:n{#1}
4738   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4739     \item[\the@pst@label]
4740   }
4741 }{
4742   \next@pst@label
4743 }

```

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.

```

4744 \newenvironment{subproof}[2][]{
4745   \_stex_sproof_spf_args:n{#1}
4746   \def\@test{#2}
4747   \ifx\@test\empty\else
4748     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4749       \item[\the@pst@label]
4750     }{#2}
4751   \fi
4752   \begin{pst@with@label}{\pst@label,\number\count_ten}
4753 }{
4754   \end{pst@with@label}\next@pst@label
4755 }

```

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

```

4756 \newenvironment{spfcases}[2][]{
4757   \def\@test{#1}
4758   \ifx\@test\empty
4759     \begin{subproof}[method=by-cases]{#2}

```

---

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

```

4760 \else
4761   \begin{subproof}[#1,method=by-cases]{#2}
4762 \fi
4763 }{
4764   \end{subproof}
4765 }

```

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

```

4766 \newenvironment{spfcase}[2] [] {
4767   \__stex_sproof_spf_args:n{#1}
4768   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4769     \item[\the@pst@label]
4770   }
4771   \def\@test{#2}
4772   \ifx\@test\@empty
4773   \else
4774     {\titleemph{#2}:~}
4775   \fi
4776   \begin{pst@with@label}{\pst@label,\number\count_ten}
4777 }{
4778   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4779     \sproofend
4780   }
4781   \end{pst@with@label}
4782   \next@pst@label
4783 }

```

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

```

4784 \newcommand\spfcasesketch[3] [] {
4785   \__stex_sproof_spf_args:n{#1}
4786   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4787     \item[\the@pst@label]
4788   }
4789   \def\@test{#2}
4790   \ifx\@test\@empty
4791   \else
4792     {\titleemph{#2}:~}
4793   \fi#3
4794   \next@pst@label
4795 }%

```

### 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.

```

4796 \keys_define:nn { stex / just }{
4797   id      .str_set_x:N = \l__stex_sproof_just_id_str,
4798   method  .tl_set:N    = \l__stex_sproof_just_method_tl,
4799   premises .tl_set:N    = \l__stex_sproof_just_premises_tl,
4800   args    .tl_set:N    = \l__stex_sproof_just_args_tl
4801 }

```

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`

4802 `\newenvironment{justification}[1] [] {}{}`

`\premise`

4803 `\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.

4804 `\newcommand\justarg[2] [] {#2}`

4805 `\</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

```
4806 <*package>
4807
4808 %%%%%%%%%%% others.dtx %%%%%%%%%%%
4809
4810 <@@=stex_others>
    Warnings and error messages
4811 % None

\MSC Math subject classifier

4812 \NewDocumentCommand \MSC {m} {
4813 % TODO
4814 }

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

4815 \@ifpackageloaded{tikzinput}{
4816 \RequirePackage{stex-tikzinput}
4817 }{}
4818 </package>
```

## Chapter 36

# STEX -Metatheory Implementation

```
4819 \*package>
4820 \@@=stex_modules>
4821
4822 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
4823
4824 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
4825 \begingroup
4826 \stex_module_setup:nn{
4827   ns=\c_stex_metatheory_ns_str,
4828   meta=NONE
4829 }{Metatheory}
4830 \stex_reactivate_macro:N \symdecl
4831 \stex_reactivate_macro:N \notation
4832 \stex_reactivate_macro:N \symdef
4833 \ExplSyntaxOff
4834 \csname stex_suppress_html:n\endcsname{
4835   % is-a (a:A, a \in A, a is an A, etc.)
4836   \symdecl[args=ai]{isa}
4837   \notation[typed]{isa}{#1 \comp{:} #2}{##1 \comp, ##2}
4838   \notation[in]{isa}{#1 \comp\in #2}{##1 \comp, ##2}
4839   \notation[pred]{isa}{#2\comp(#1 \comp)}{##1 \comp, ##2}
4840
4841   % bind (\forall, \Pi, \lambda etc.)
4842   \symdecl[args=Bi]{bind}
4843   \notation[forall]{bind}{\comp\forall #1.\;#2}{##1 \comp, ##2}
4844   \notation[\Pi]{bind}{\comp\prod_{#1}#2}{##1 \comp, ##2}
4845   \notation[deffun]{bind}{\comp( #1 \comp{ }\;\to\; ) #2}{##1 \comp, ##2}
4846
4847   % dummy variable
4848   \symdecl{dummyvar}
4849   \notation[underscore]{dummyvar}{\comp\_}
4850   \notation[dot]{dummyvar}{\comp\cdot}
4851   \notation[dash]{dummyvar}{\comp{\rm --}}
4852
4853   %fromto (function space, Hom-set, implication etc.)
```

```

4854 \symdecl[args=ai]{fromto}
4855 \notation[xarrow]{fromto}{#1 \comp\to #2}{##1 \comp\times ##2}
4856 \notation[arrow]{fromto}{#1 \comp\to #2}{##1 \comp\to ##2}
4857
4858 % mapto (lambda etc.)
4859 %\symdecl[args=Bi]{mapto}
4860 %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
4861 %\notation[lambda]{mapto}{\comp\lambda #1 \comp.; #2}{#1 \comp, #2}
4862 %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp.; #2}{#1 \comp, #2}
4863
4864 % function/operator application
4865 \symdecl[args=ia]{apply}
4866 \notation[prec=0;0x\infpres,parens]{apply}{#1 \comp( #2 \comp)}{##1 \comp, ##2}
4867 \notation[prec=0;0x\infpres,lambda]{apply}{#1 \; #2 }{##1 \; ; ##2}
4868
4869 % ‘type’ of all collections (sets, classes, types, kinds)
4870 \symdecl{collection}
4871 \notation[U]{collection}{\comp{\mathcal{U}}}
4872 \notation[set]{collection}{\comp{\textsf{Set}}}
4873
4874 % sequences
4875 \symdecl[args=1]{seqtype}
4876 \notation[kleene]{seqtype}{#1^{\comp\ast}}
4877
4878 \symdef[args=2,li,prec=nobrackets]{sequence-index}{#1}_{#2}
4879 \notation[ui,prec=nobrackets]{sequence-index}{#1}^{#2}
4880
4881 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{,\ellipses}}{##1\comp,##2}
4882 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{,\ellipses},#2}{##1\comp,##2}
4883 \symdef[args=aui,prec=nobrackets]{aseqfromtovia}{#1\comp{,\ellipses},#2\comp{,\ellipses},#3}
4884
4885 % letin (‘let’, local definitions, variable substitution)
4886 \symdecl[args=bii]{letin}
4887 \notation[let]{letin}{\comp{\rm let}}{#1\comp{=}#2;\comp{\rm in}}{#3}
4888 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
4889 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
4890
4891 % structures
4892 \symdecl*[args=1]{module-type}
4893 \notation{module-type}{\mathtt{MOD} #1}
4894 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
4895 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{##1 \comp, ##2}
4896
4897 }
4898 \ExplSyntaxOn
4899 \stex_add_to_current_module:n{
4900   \let\nappa\apply
4901   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
4902   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
4903   \def\livar{\csname sequence-index\endcsname[li]}
4904   \def\uivar{\csname sequence-index\endcsname[ui]}
4905   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
4906   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
4907   \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}

```



```
4908 }  
4909 \__stex_modules_end_module:  
4910 \endgroup  
4911 </package>
```

## Chapter 37

# Tikzinput Implementation

```
4912 <*package>
4913
4914 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
4915
4916 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
4917 \RequirePackage{l3keys2e}
4918
4919 \keys_define:nn { tikzinput } {
4920   image .bool_set:N = \c_tikzinput_image_bool,
4921   image .default:n = false ,
4922   unknown .code:n = {}
4923 }
4924
4925 \ProcessKeysOptions { tikzinput }
4926
4927 \bool_if:NTF \c_tikzinput_image_bool {
4928   \RequirePackage{graphicx}
4929
4930   \providecommand\usetikzlibrary[]{}
4931   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
4932 }{
4933   \RequirePackage{tikz}
4934   \RequirePackage{standalone}
4935
4936   \newcommand \tikzinput [2] [] {
4937     \setkeys{Gin}{#1}
4938     \ifx \Gin@ewidth \Gin@exclamation
4939       \ifx \Gin@eheight \Gin@exclamation
4940         \input { #2 }
4941       \else
4942         \resizebox{!}{ \Gin@eheight }{
4943           \input { #2 }
4944         }
4945       \fi
4946     \else
4947       \ifx \Gin@eheight \Gin@exclamation
4948         \resizebox{ \Gin@ewidth }{!}{
4949           \input { #2 }
```

```

4950     }
4951     \else
4952         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
4953             \input { #2 }
4954         }
4955     \fi
4956 \fi
4957 }
4958 }
4959
4960 \newcommand \ctikzinput [2] [] {
4961     \begin{center}
4962         \tikzinput [1] {#2}
4963     \end{center}
4964 }
4965
4966 \@ifpackageloaded{stex}{
4967     \RequirePackage{stex-tikzinput}
4968 }{}
4969
4970 </package>
4971 <*stex>
4972 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
4973 \RequirePackage{stex}
4974 \RequirePackage{tikzinput}
4975
4976 \newcommand\mhtikzinput [2] [] {%
4977     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
4978     \stex_in_repository:nn\Gin@mhrepos{
4979         \tikzinput [1]{\mhpath{##1}{#2}}
4980     }
4981 }
4982 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [1] {#2}\end{center}}
4983 </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.

```
4984 \*cls)
4985 \@@=document_structure)
4986 \ProvidesExplClass{document-structure}{2022/02/10}{3.0}{Modular Document Structure Class}
4987 \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`

```
4988 \keys_define:nn{ document-structure / pkg }{
4989   class      .str_set_x:N = \c_document_structure_class_str,
4990   minimal    .bool_set:N = \c_document_structure_minimal_bool,
4991   report     .code:n      = {
4992     \ClassWarning{document-structure}{the option 'report' is deprecated, use 'class=report',
4993     \str_set:Nn \c_document_structure_class_str {report}
4994   },
4995   book       .code:n      = {
4996     \ClassWarning{document-structure}{the option 'book' is deprecated, use 'class=book', ins
4997     \str_set:Nn \c_document_structure_class_str {book}
4998   },
4999   bookpart   .code:n      = {
5000     \ClassWarning{document-structure}{the option 'bookpart' is deprecated, use 'class=book,t
5001     \str_set:Nn \c_document_structure_class_str {book}
5002     \str_set:Nn \c_document_structure_topsect_str {chapter}
5003   },
```

```

5004 docopt      .str_set_x:N = \c_document_structure_docopt_str,
5005 unknown     .code:n      = {
5006   \PassOptionsToPackage{ \CurrentOption }{ document-structure }
5007 }
5008 }
5009 \ProcessKeysOptions{ document-structure / pkg }
5010 \str_if_empty:NT \c_document_structure_class_str {
5011   \str_set:Nn \c_document_structure_class_str {article}
5012 }
5013 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
5014   {\c_document_structure_class_str}
5015

```

### 38.3 Beefing up the document environment

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

```

5016 \RequirePackage{document-structure}
5017 \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>

```

5018 \keys_define:nn { document-structure / document }{
5019   id .str_set_x:N = \c_document_structure_document_id_str
5020 }
5021 \let\__document_structure_orig_document=\document
5022 \renewcommand{\document}[1][]{
5023   \keys_set:nn{ document-structure / document }{ #1 }
5024   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
5025   \__document_structure_orig_document
5026 }

```

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

```

5027 }
5028 \</cls>

```

### 38.4 Implementation: document-structure Package

```

5029 \<*package>
5030 \ProvidesExplPackage{document-structure}{2022/02/10}{3.0}{Modular Document Structure}
5031 \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

```

5032
5033 \keys_define:nn{ document-structure / pkg }{
5034   class      .str_set_x:N = \c_document_structure_class_str,
5035   topsect     .str_set_x:N = \c_document_structure_topsect_str,
5036   % showignores .bool_set:N = \c_document_structure_showignores_bool,
5037 }
5038 \ProcessKeysOptions{ document-structure / pkg }
5039 \str_if_empty:NT \c_document_structure_class_str {
5040   \str_set:Nn \c_document_structure_class_str {article}
5041 }
5042 \str_if_empty:NT \c_document_structure_topsect_str {
5043   \str_set:Nn \c_document_structure_topsect_str {section}
5044 }

```

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

```

5045 \RequirePackage{xspace}
5046 \RequirePackage{comment}
5047 \AddToHook{begindocument}{
5048   \ltx@ifpackageloaded{babel}{
5049     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5050     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5051       \makeatletter\input{omdoc-ngerman.ldf}\makeatother
5052     }
5053   }{}
5054 }

```

`\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.

```

5055 \int_new:N \l_document_structure_section_level_int
5056 \str_case:VnF \c_document_structure_topsect_str {
5057   {part}}{
5058     \int_set:Nn \l_document_structure_section_level_int {0}
5059   }
5060   {chapter}}{
5061     \int_set:Nn \l_document_structure_section_level_int {1}
5062   }
5063 }{
5064   \str_case:VnF \c_document_structure_class_str {
5065     {book}}{
5066       \int_set:Nn \l_document_structure_section_level_int {0}
5067     }
5068     {report}}{
5069       \int_set:Nn \l_document_structure_section_level_int {0}
5070     }
5071   }{
5072     \int_set:Nn \l_document_structure_section_level_int {2}
5073   }
5074 }

```

## 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

```
5075 \def\current@section@level{document}%
5076 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
5077 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
5078 \cs_new_protected:Npn \skipomgroup {
5079   \ifcase\l_document_structure_section_level_int
5080   \or\stepcounter{part}
5081   \or\stepcounter{chapter}
5082   \or\stepcounter{section}
5083   \or\stepcounter{subsection}
5084   \or\stepcounter{subsubsection}
5085   \or\stepcounter{paragraph}
5086   \or\stepcounter{subparagraph}
5087   \fi
5088 }
```

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

`blindomgroup`

```
5089 \newcommand\at@begin@blindomgroup[1]{%
5090 \newenvironment{blindomgroup}
5091 {
5092   \int_incr:N\l_document_structure_section_level_int
5093   \at@begin@blindomgroup\l_document_structure_section_level_int
5094 }{}}
```

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

```
5095 \newcommand\omgroup@nonum[2]{
5096   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
5097   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
5098 }
```

*(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.

```
5099 \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.

```

5100 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
5101   \@nameuse{#1}{#2}
5102 }{
5103   \cs_if_exist:NTF\rdfmata@sectioning{
5104     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
5105   }{
5106     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
5107   }
5108 }
5109 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
5110 }

```

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

omgroup

```

5111 \keys_define:nn { document-structure / omgroup }{
5112   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
5113   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
5114   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
5115   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
5116   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
5117   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
5118   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
5119   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
5120   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
5121   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
5122 }
5123 \cs_new_protected:Nn \__document_structure_omgroup_args:n {
5124   \str_clear:N \l__document_structure_omgroup_id_str
5125   \str_clear:N \l__document_structure_omgroup_date_str
5126   \clist_clear:N \l__document_structure_omgroup_creators_clist
5127   \clist_clear:N \l__document_structure_omgroup_contributors_clist
5128   \tl_clear:N \l__document_structure_omgroup_srccite_tl
5129   \tl_clear:N \l__document_structure_omgroup_type_tl
5130   \tl_clear:N \l__document_structure_omgroup_short_tl
5131   \tl_clear:N \l__document_structure_omgroup_display_tl
5132   \tl_clear:N \l__document_structure_omgroup_intro_tl
5133   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
5134   \keys_set:nn { document-structure / omgroup } { #1 }
5135 }

```

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.

```

5136 \newif\if@mainmatter\@mainmattertrue
5137 \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.

```

5138 \keys_define:nn { document-structure / sectioning }{
5139   name .str_set_x:N = \l__document_structure_sect_name_str ,
5140   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
5141   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
5142   num .bool_set:N = \l__document_structure_sect_num_bool ,
5143 }

```



```

5144 \cs_new_protected:Nn \__document_structure_sect_args:n {
5145   \str_clear:N \l__document_structure_sect_name_str
5146   \str_clear:N \l__document_structure_sect_ref_str
5147   \bool_set_false:N \l__document_structure_sect_clear_bool
5148   \bool_set_false:N \l__document_structure_sect_num_bool
5149   \keys_set:nn { document-structure / sectioning } { #1 }
5150 }
5151 \newcommand\omdoc@sectioning[3][]{
5152   \__document_structure_sect_args:n {#1 }
5153   \let\omdoc@sect@name\l__document_structure_sect_name_str
5154   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
5155   \if@mainmatter% numbering not overridden by frontmatter, etc.
5156     \bool_if:NTF \l__document_structure_sect_num_bool {
5157       \omgroup@num{#2}{#3}
5158     }{
5159       \omgroup@nonum{#2}{#3}
5160     }
5161     \def\current@section@level{\omdoc@sect@name}
5162   \else
5163     \omgroup@nonum{#2}{#3}
5164   \fi
5165 }% 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.

```

5166 \newcommand\omgroup@redefine@addtocontents[1]{%
5167   %\edef\__document_structureimport{#1}%
5168   %\@for\@I:=\__document_structureimport\do{%
5169     %\edef\@path{\csname module@\@I @path\endcsname}%
5170     %\@ifundefined{tf@toc}\relax%
5171     %    {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
5172   %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
5173   %\def\addcontentsline##1##2##3{%
5174     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
5175   %\else% hyperref.sty not loaded
5176   %\def\addcontentsline##1##2##3{%
5177     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
5178   %\fi
5179 }% 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.

```

5180 \int_new:N \l_document_structure_omgroup_level_int
5181 \newenvironment{omgroup}[2][]{% keys, title
5182 {
5183   \__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.

```

5184 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
5185   \omgroup@redefine@addtocontents{
5186     %\@ifundefined{module@id}\used@modules%
5187     %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}

```

```

5188     }
5189 }

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

5190 \int_incr:N \l_document_structure_omgroup_level_int
5191 \int_incr:N \l_document_structure_section_level_int
5192 \ifcase\l_document_structure_section_level_int
5193   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
5194   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
5195   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
5196   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
5197   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
5198   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
5199   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
5200 \fi
5201 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
5202 \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
5203 }% for customization
5204 {}

```

and finally, we localize the sections

```

5205 \newcommand\omdoc@part@kw{Part}
5206 \newcommand\omdoc@chapter@kw{Chapter}
5207 \newcommand\omdoc@section@kw{Section}
5208 \newcommand\omdoc@subsection@kw{Subsection}
5209 \newcommand\omdoc@subsubsection@kw{Subsubsection}
5210 \newcommand\omdoc@paragraph@kw{paragraph}
5211 \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`

```

5212 \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).

```

5213 \cs_if_exist:NTF\frontmatter{
5214   \let\__document_structure_orig_frontmatter\frontmatter
5215   \let\frontmatter\relax
5216 }{
5217   \tl_set:Nn\__document_structure_orig_frontmatter{
5218     \clearpage
5219     \@mainmatterfalse
5220     \pagenumbering{roman}
5221   }
5222 }

```

```

5223 \cs_if_exist:NTF\backmatter{
5224   \let\__document_structure_orig_backmatter\backmatter
5225   \let\backmatter\relax
5226 }{
5227   \tl_set:Nn\__document_structure_orig_backmatter{
5228     \clearpage
5229     \@mainmatterfalse
5230     \pagenumbering{roman}
5231   }
5232 }

```

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.

```

5233 \newenvironment{frontmatter}{
5234   \__document_structure_orig_frontmatter
5235 }{
5236   \cs_if_exist:NTF\mainmatter{
5237     \mainmatter
5238   }{
5239     \clearpage
5240     \@mainmattertrue
5241     \pagenumbering{arabic}
5242   }
5243 }

```

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

```

5244 \newenvironment{backmatter}{
5245   \__document_structure_orig_backmatter
5246 }{
5247   \cs_if_exist:NTF\mainmatter{
5248     \mainmatter
5249   }{
5250     \clearpage
5251     \@mainmattertrue
5252     \pagenumbering{arabic}
5253   }
5254 }

```

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

```

5255 \@mainmattertrue\pagenumbering{arabic}

```

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

```

5256 \def \c__document_structure_document_str{document}
5257 \newcommand\afterprematurestop{}
5258 \def\prematurestop@endomgroup{
5259   \unless\ifx\@currenvir\c__document_structure_document_str
5260     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter\expandafter\expandafter
5261     \expandafter\prematurestop@endomgroup
5262   \fi
5263 }

```

```

5264 \providecommand\prematurestop{
5265   \message{Stopping~sTeX~processing~prematurely}
5266   \prematurestop@endomgroup
5267   \afterprematurestop
5268   \end{document}
5269 }

```

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

## 38.8 Global Variables

**\setSGvar** set a global variable

```

5270 \RequirePackage{etoolbox}
5271 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

**\useSGvar** use a global variable

```

5272 \newrobustcmd\useSGvar[1]{%
5273   \@ifundefined{sTeX@Gvar@#1}
5274   {\PackageError{document-structure}
5275    {The sTeX Global variable #1 is undefined}
5276    {set it with \protect\setSGvar}}
5277   \@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.

```

5278 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
5279   \@ifundefined{sTeX@Gvar@#1}
5280   {\PackageError{document-structure}
5281    {The sTeX Global variable #1 is undefined}
5282    {set it with \protect\setSGvar}}
5283   {\expandafter\ifx\cename 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.

```
5284 \*cls)
5285 \@@=notesslides)
5286 \ProvidesExplClass{notesslides}{2022/02/10}{3.0}{notesslides Class}
5287 \RequirePackage{l3keys2e,expl-keystr-compatible}
5288
5289 \keys_define:nn{notesslides / cls}{
5290   class .code:n = {
5291     \PassOptionsToClass{\CurrentOption}{omdoc}
5292     \str_if_eq:nnT{#1}{book}{
5293       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5294     }
5295     \str_if_eq:nnT{#1}{report}{
5296       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5297     }
5298   },
5299   notes .bool_set:N = \c__notesslides_notes_bool ,
5300   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
5301   unknown .code:n = {
5302     \PassOptionsToClass{\CurrentOption}{omdoc}
5303     \PassOptionsToClass{\CurrentOption}{beamer}
5304     \PassOptionsToPackage{\CurrentOption}{notesslides}
5305   }
5306 }
5307 \ProcessKeysOptions{ notesslides / cls }
5308 \bool_if:NTF \c__notesslides_notes_bool {
5309   \PassOptionsToPackage{notes=true}{notesslides}
5310 }{
5311   \PassOptionsToPackage{notes=false}{notesslides}
5312 }
5313 \</cls)
```

now we do the same for the notesslides package.

```

5314 <*package>
5315 \ProvidesExplPackage{notesslides}{2022/02/10}{3.0}{notesslides Package}
5316 \RequirePackage{13keys2e,expl-keystr-compat}
5317
5318 \keys_define:nn{notesslides / pkg}{
5319   topsect      .str_set_x:N = \c__notesslides_topsect_str,
5320   defaulttopsect .str_set_x:N = \c__notesslides_defaulttopsec_str,
5321   notes        .bool_set:N = \c__notesslides_notes_bool ,
5322   slides       .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
5323   sectocframes .bool_set:N = \c__notesslides_sectocframes_bool ,
5324   frameimages  .bool_set:N = \c__notesslides_frameimages_bool ,
5325   fiboxed      .bool_set:N = \c__notesslides_fiboxed_bool ,
5326   nopproblems  .bool_set:N = \c__notesslides_nopproblems_bool,
5327   unknown      .code:n      = {
5328     \PassOptionsToClass{\CurrentOption}{stex}
5329     \PassOptionsToClass{\CurrentOption}{tikzinput}
5330   }
5331 }
5332 \ProcessKeysOptions{ notesslides / pkg }
5333 \newif\ifnotes
5334 \bool_if:NTF \c__notesslides_notes_bool {
5335   \notesttrue
5336 }{
5337   \notesfalse
5338 }
5339

```

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

```

5340 \str_if_empty:NTF \c__notesslides_topsect_str {
5341   \str_set_eq:NN \__notesslides_topsect \c__notesslides_defaulttopsec_str
5342 }{
5343   \str_set_eq:NN \__notesslides_topsect \c__notesslides_topsect_str
5344 }
5345 </package>

```

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

```

5346 <*cls>
5347 \bool_if:NTF \c__notesslides_notes_bool {
5348   \LoadClass{document-structure}
5349 }{
5350   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
5351   \newcounter{Item}
5352   \newcounter{paragraph}
5353   \newcounter{subparagraph}
5354   \newcounter{Hfootnote}
5355   \RequirePackage{document-structure}
5356 }

```

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

```

5357 \RequirePackage{notesslides}
5358 </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.

```

5359 \*package>
5360 \bool_if:NT \c__notesslides_notes_bool {
5361   \RequirePackage{a4wide}
5362   \RequirePackage{marginnote}
5363   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
5364   \RequirePackage{mdframed}
5365   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
5366   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
5367 }
5368 \RequirePackage{stex-tikzinput}
5369 \RequirePackage{etoolbox}
5370 \RequirePackage{amssymb}
5371 \RequirePackage{amsmath}
5372 \RequirePackage{comment}
5373 \RequirePackage{textcomp}
5374 \RequirePackage{url}
5375 \RequirePackage{graphicx}
5376 \RequirePackage{pgf}

```

## 39.2 Notes and Slides

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

```

5377 \bool_if:NT \c__notesslides_notes_bool {
5378   \renewcommand\usetheme[2][\usepackage[#1]{beamernotestheme#2}]
5379 }

```

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

```

5380 \newcounter{slide}
5381 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
5382 \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.

```

5383 \bool_if:NTF \c__notesslides_notes_bool {
5384   \renewenvironment{note}{\ignorespaces}{}
5385 }{
5386   \excludecomment{note}
5387 }

```

---

<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.

```
5388 \bool_if:NT \c__notesslides_notes_bool {
5389   \newlength{\slideframewidth}
5390   \setlength{\slideframewidth}{1.5pt}
```

**frame** We first define the keys.

```
5391 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
5392   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
5393     \bool_set_true:N #1
5394   }{
5395     \bool_set_false:N #1
5396   }
5397 }
5398 \keys_define:nn{notesslides / frame}{
5399   label .str_set_x:N = \l__notesslides_frame_label_str,
5400   allowframebreaks .code:n = {
5401     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
5402   },
5403   allowdisplaybreaks .code:n = {
5404     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
5405   },
5406   fragile .code:n = {
5407     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
5408   },
5409   shrink .code:n = {
5410     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
5411   },
5412   squeeze .code:n = {
5413     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
5414   },
5415   t .code:n = {
5416     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
5417   },
5418 }
5419 \cs_new_protected:Nn \__notesslides_frame_args:n {
5420   \str_clear:N \l__notesslides_frame_label_str
5421   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
5422   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
5423   \bool_set_true:N \l__notesslides_frame_fragile_bool
5424   \bool_set_true:N \l__notesslides_frame_shrink_bool
5425   \bool_set_true:N \l__notesslides_frame_squeeze_bool
5426   \bool_set_true:N \l__notesslides_frame_t_bool
5427   \keys_set:nn { notesslides / frame }{ #1 }
5428 }
```

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

```
5429 \renewenvironment{frame}[1][]{
5430   \__notesslides_frame_args:n{#1}
5431   \sffamily
5432   \stepcounter{slide}
5433   \def\@currentlabel{\theslide}
5434   \str_if_empty:NF \l__notesslides_frame_label_str {
5435     \label{\l__notesslides_frame_label_str}
```



5436 }  
5437

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

5437 \def\itemize@level{outer}  
5438 \def\itemize@outer{outer}  
5439 \def\itemize@inner{inner}  
5440 \renewcommand\newpage{\addtocounter{framenum}{1}}  
5441 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}  
5442 \renewenvironment{itemize}{  
5443 \ifx\itemize@level\itemize@outer  
5444 \def\itemize@label{\rhd\$}  
5445 \fi  
5446 \ifx\itemize@level\itemize@inner  
5447 \def\itemize@label{\$\scriptstyle\rhd\$}  
5448 \fi  
5449 \begin{list}  
5450 {\itemize@label}  
5451 {\setlength{\labelsep}{.3em}  
5452 \setlength{\labelwidth}{.5em}  
5453 \setlength{\leftmargin}{1.5em}  
5454 }  
5455 \edef\itemize@level{\itemize@inner}  
5456 }{  
5457 \end{list}  
5458 }

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

5459 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth=100pt]  
5460 }{  
5461 \medskip\miko@slidelabel\end{mdframed}  
5462 }

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

\frametitle

5463 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip  
5464 }

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

EdN:21

\pause 21

5465 \bool\_if:NT \c\_\_notesslides\_notes\_bool {  
5466 \newcommand\pause{  
5467 }  
5468 }

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

nparagraph

5468 \bool\_if:NTF \c\_\_notesslides\_notes\_bool {  
5469 \newenvironment{nparagraph}[1][\begin{sparagraph}[#1]}{\end{sparagraph}}  
5470 }{  
5471 \excludecomment{nparagraph}  
5472 }  
5473 }

---

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

```

nomgroup
5473 \bool_if:NTF \c__notesslides_notes_bool {
5474 \newenvironment{nomgroup}[2] [] {\begin{omgroup}[#1]{#2}}{\end{omgroup}}
5475 }{
5476 \excludecomment{nomgroup}
5477 }

ntheorem
5478 \bool_if:NTF \c__notesslides_notes_bool {
5479 \newenvironment{ntheorem}[1] [] {\begin{sdefinition}[#1]}{\end{sdefinition}}
5480 }{
5481 \excludecomment{ntheorem}
5482 }

nassertion
5483 \bool_if:NTF \c__notesslides_notes_bool {
5484 \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}{\end{sassertion}}
5485 }{
5486 \excludecomment{nassertion}
5487 }

nsproof
5488 \bool_if:NTF \c__notesslides_notes_bool {
5489 \newenvironment{nsproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
5490 }{
5491 \excludecomment{nsproof}
5492 }

nexample
5493 \bool_if:NTF \c__notesslides_notes_bool {
5494 \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}{\end{sexample}}
5495 }{
5496 \excludecomment{nexample}
5497 }

\inputref@*skip We customize the hooks for in \inputref.
5498 \def\inputref@preskip{\smallskip}
5499 \def\inputref@postskip{\medskip}

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

\inputref*
5500 \let\orig@inputref\inputref
5501 \def\inputref{\@ifstar\ninputref\orig@inputref}
5502 \newcommand\ninputref[2] [] {
5503 \bool_if:NT \c__notesslides_notes_bool {
5504 \orig@inputref[#1]{#2}
5505 }
5506 }

(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>}`.

```
5507 \newlength{\slidelogoheight}
5508
5509 \bool_if:NTF \c__notesslides_notes_bool {
5510   \setlength{\slidelogoheight}{.4cm}
5511 }{
5512   \setlength{\slidelogoheight}{1cm}
5513 }
5514 \newsavebox{\slidelogo}
5515 \sbox{\slidelogo}{\TeX}
5516 \newrobustcmd{\setslidelogo}[1]{
5517   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
5518 }
```

(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.

```
5519 \def\source{Michael Kohlhase}% customize locally
5520 \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.

```
5521 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
5522 \newsavebox{\cclogo}
5523 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
5524 \newif\ifcchref\cchreffalse
5525 \AtBeginDocument{
5526   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
5527 }
5528 \def\licensing{
5529   \ifcchref
5530     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
5531   \else
5532     {\usebox{\cclogo}}
5533   \fi
5534 }
5535 \newrobustcmd{\setlicensing}[2][]{
5536   \def@url{#1}
5537   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
5538   \ifx@url@empty
5539     \def\licensing{{\usebox{\cclogo}}}
5540   \else
5541     \def\licensing{
```

```

5542     \ifcchref
5543     \href{#1}{\usebox{\cclogo}}
5544     \else
5545     {\usebox{\cclogo}}
5546     \fi
5547   }
5548 \fi
5549 }

```

(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>

```

5550 \newrobustcmd\miko@slidelabel{
5551   \vbox to \slidelogoheight{
5552     \vss\hbox to \slidewidth
5553     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
5554   }
5555 }

```

(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.

```

5556 \def\Gin@mhrepos{}
5557 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
5558 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
5559 \newrobustcmd\frameimage[2][]{
5560   \stepcounter{slide}
5561   \bool_if:NT \c__notesslides_frameimages_bool {
5562     \def\Gin@ewidth{}\setkeys{Gin}{#1}
5563     \bool_if:NF \c__notesslides_notes_bool { \vfill }
5564     \begin{center}
5565       \bool_if:NTF \c__notesslides_fiboxed_bool {
5566         \fbox{
5567           \ifx\Gin@ewidth\@empty
5568             \ifx\Gin@mhrepos\@empty
5569               \mhgraphics[width=\slidewidth,#1]{#2}
5570             \else
5571               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5572             \fi
5573           \else% Gin@ewidth empty
5574             \ifx\Gin@mhrepos\@empty
5575               \mhgraphics[#1]{#2}
5576             \else
5577               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5578             \fi
5579           \fi% Gin@ewidth empty
5580         }
5581       }{
5582         \ifx\Gin@ewidth\@empty

```

---

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

```

5583         \ifx\Gin@mhrepos\@empty
5584             \mhgraphics[width=\slidewidth,#1]{#2}
5585         \else
5586             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5587         \fi
5588         \ifx\Gin@mhrepos\@empty
5589             \mhgraphics[#1]{#2}
5590         \else
5591             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5592         \fi
5593     \fi% Gin@ewidth empty
5594 }
5595 \end{center}
5596 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
5597 \bool_if:NF \c__notesslides_notes_bool { \vfill }
5598 }
5599 } % 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.

```

5600 \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

```

5601 \AddToHook{begindocument}{
5602     \definecolor{green}{rgb}{0,.5,0}
5603     \definecolor{purple}{cmyk}{.3,1,0,.17}
5604 }

```

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`.

```

5605 % \def\STpresent#1{\textcolor{blue}{#1}}
5606 \def\defemph#1{\textcolor{magenta}{#1}}
5607 \def\symrefemph#1{\textcolor{cyan}{#1}}
5608 \def\compemph#1{\textcolor{blue}{#1}}
5609 \def\titleemph#1{\textcolor{blue}{#1}}
5610 \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.

```

5611 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
5612 \def\smalltextwarning{
5613     \pgfuseimage{miko@small@dbend}
5614     \xspace
5615 }
5616 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}

```

```

5617 \newrobustcmd\textwarning{
5618   \raisebox{-0.05cm}{\pgfuseimage{miko@dbend}}
5619   \xspace
5620 }
5621 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}
5622 \newrobustcmd\bigtextwarning{
5623   \raisebox{-0.05cm}{\pgfuseimage{miko@big@dbend}}
5624   \xspace
5625 }

```

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

```

5626 \newrobustcmd\putgraphicsat[3]{
5627   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
5628 }
5629 \newrobustcmd\putat[2]{
5630   \begin{picture}(0,0)\put(#1){#2}\end{picture}
5631 }

```

## 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`.

```

5632 \bool_if:NT \c__notesslides_sectocframes_bool {
5633   \str_if_eq:VnTF \__notesslidesstopsect{part}{
5634     \newcounter{chapter}\counterwithin*{section}{chapter}
5635   }{
5636     \str_if_eq:VnT\__notesslidesstopsect{chapter}{
5637       \newcounter{chapter}\counterwithin*{section}{chapter}
5638     }
5639   }
5640 }

```

`\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
5641 \def\part@prefix{}
5642 \@ifpackageloaded{document-structure}{\{
5643   \str_case:VnF \__notesslidesstopsect {
5644     {part}{
5645       \int_set:Nn \l_document_structure_section_level_int {0}
5646       \def\thesection{\arabic{chapter}.\arabic{section}}
5647       \def\part@prefix{\arabic{chapter}.}
5648     }
5649     {chapter}{
5650       \int_set:Nn \l_document_structure_section_level_int {1}
5651       \def\thesection{\arabic{chapter}.\arabic{section}}
5652       \def\part@prefix{\arabic{chapter}.}
5653     }
5654   }{
5655     \int_set:Nn \l_document_structure_section_level_int {2}
5656     \def\part@prefix{}

```

```

5657 }
5658 }
5659
5660 \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`

```

5661 \renewenvironment{omgroup}[2][]{
5662   \__document_structure_omgroup_args:n { #1 }
5663   \int_incr:N \l_document_structure_omgroup_level_int
5664   \int_incr:N \l_document_structure_section_level_int
5665   \bool_if:NT \c__notesslides_sectocframes_bool {
5666     \stepcounter{slide}
5667     \begin{frame}[noframenumbering]
5668     \vfill\Large\centering
5669     \red{
5670       \ifcase\l_document_structure_section_level_int\or
5671         \stepcounter{part}
5672         \def\__notesslideslabel{\omdoc@part@kw~\Roman{part}}
5673         \def\currentsectionlevel{\omdoc@part@kw}
5674       \or
5675         \stepcounter{chapter}
5676         \def\__notesslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
5677         \def\currentsectionlevel{\omdoc@chapter@kw}
5678       \or
5679         \stepcounter{section}
5680         \def\__notesslideslabel{\part@prefix\arabic{section}}
5681         \def\currentsectionlevel{\omdoc@section@kw}
5682       \or
5683         \stepcounter{subsection}
5684         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
5685         \def\currentsectionlevel{\omdoc@subsection@kw}
5686       \or
5687         \stepcounter{subsubsection}
5688         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
5689         \def\currentsectionlevel{\omdoc@subsubsection@kw}
5690       \or
5691         \stepcounter{paragraph}
5692         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
5693         \def\currentsectionlevel{\omdoc@paragraph@kw}
5694       \else
5695         \def\__notesslideslabel{}
5696         \def\currentsectionlevel{\omdoc@paragraph@kw}
5697       \fi% end ifcase
5698       \__notesslideslabel%\sref@label@id\__notesslideslabel
5699       \quad #2%
5700     }%
5701     \vfill%
5702     \end{frame}%
5703   }
5704   \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str%

```

```

5705 }{}
5706 }

```

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

```

5707 \def\inserttheorembodyfont{\normalfont}
5708 %\bool_if:NF \c__notesslides_notes_bool {
5709 % \defbeamertemplate{theorem begin}{miko}
5710 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
5711 % \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
5712 % \inserttheorempunctuation\inserttheorembodyfont\hspace}
5713 % \defbeamertemplate{theorem end}{miko}{}}

```

and we set it as the default one.

```

5714 % \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.

```

5715 % \expandafter\def\csname Parent2\endcsname{}
5716 %}
5717
5718 \AddToHook{begindocument}{ % this does not work for some reasons
5719 \setbeamertemplate{theorems}[ams style]
5720 }
5721 \bool_if:NT \c__notesslides_notes_bool {
5722 \renewenvironment{columns}[1][{}]{%
5723 \par\noindent%
5724 \begin{minipage}%
5725 \slidewidth\centering\leavevmode%
5726 }{}%
5727 \end{minipage}\par\noindent%
5728 }%
5729 \newsavebox\columnbox%
5730 \renewenvironment<>{column}[2][{}]{%
5731 \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
5732 }{}%
5733 \end{minipage}\end{lrbox}\usebox\columnbox%
5734 }%
5735 }
5736 \bool_if:NTF \c__notesslides_noproblems_bool {
5737 \newenvironment{problems}{}{}
5738 }{
5739 \excludecomment{problems}
5740 }

```

## 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.

```

5741 \gdef\printexcursions{}
5742 \newcommand\excursionref[2]{% label, text
5743 \bool_if:NT \c__notesslides_notes_bool {

```



```

5744 \begin{sparagraph}[title=Excursion]
5745 #2 \sref[fallback=the appendix]{#1}.
5746 \end{sparagraph}
5747 }
5748 }
5749 \newcommand\activate@excursion[2][]{
5750 \gappto\printexcursions{\inputref[#1]{#2}}
5751 }
5752 \newcommand\excursion[4][]{% repos, label, path, text
5753 \bool_if:NT \c__notesslides_notes_bool {
5754 \activate@excursion[#1]{#3}\excursionref{#2}{#4}
5755 }
5756 }

```

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

\excursiongroup

```

5757 \keys_define:nn{notesslides / excursiongroup }{
5758 id .str_set_x:N = \l__notesslides_excursion_id_str,
5759 intro .tl_set:N = \l__notesslides_excursion_intro_tl,
5760 mhrepos .str_set_x:N = \l__notesslides_excursion_mhrepos_str
5761 }
5762 \cs_new_protected:Nn \__notesslides_excursion_args:n {
5763 \tl_clear:N \l__notesslides_excursion_intro_tl
5764 \str_clear:N \l__notesslides_excursion_id_str
5765 \str_clear:N \l__notesslides_excursion_mhrepos_str
5766 \keys_set:nn {notesslides / excursiongroup }{ #1 }
5767 }
5768 \newcommand\excursiongroup[1][]{
5769 \__notesslides_excursion_args:n{ #1 }
5770 \ifdefempty\printexcursions{}% only if there are excursions
5771 {\begin{note}
5772 \begin{omgroup}[#1]{Excursions}%
5773 \ifdefempty\l__notesslides_excursion_intro_tl}{
5774 \inputref[\l__notesslides_excursion_mhrepos_str]{
5775 \l__notesslides_excursion_intro_tl
5776 }
5777 }
5778 \printexcursions%
5779 \end{omgroup}
5780 \end{note}}
5781 }
5782 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
5783 \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.

```
5784 <*package>
5785 <@@=problems>
5786 \ProvidesExplPackage{problem}{2019/03/20}{1.3}{Semantic Markup for Problems}
5787 \RequirePackage{l3keys2e,expl-keystr-compat}
5788
5789 \keys_define:nn { problem / pkg }{
5790   notes      .default:n    = { true },
5791   notes      .bool_set:N   = \c__problems_notes_bool,
5792   gnotes     .default:n    = { true },
5793   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
5794   hints      .default:n    = { true },
5795   hints      .bool_set:N   = \c__problems_hints_bool,
5796   solutions  .default:n    = { true },
5797   solutions  .bool_set:N   = \c__problems_solutions_bool,
5798   pts        .default:n    = { true },
5799   pts        .bool_set:N   = \c__problems_pts_bool,
5800   min        .default:n    = { true },
5801   min        .bool_set:N   = \c__problems_min_bool,
5802   boxed      .default:n    = { true },
5803   boxed      .bool_set:N   = \c__problems_boxed_bool,
5804   unknown    .code:n       = {}
5805 }
5806 \newif\ifsolutions
5807
5808 \ProcessKeysOptions{ problem / pkg }
5809 \bool_if:NTF \c__problems_solutions_bool {
5810   \solutionstrue
5811 }{
5812   \solutionsfalse
5813 }
```

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

```
5814 \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.

```
5815 \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.

```
5816 \def\prob@problem@kw{Problem}
5817 \def\prob@solution@kw{Solution}
5818 \def\prob@hint@kw{Hint}
5819 \def\prob@note@kw{Note}
5820 \def\prob@gnote@kw{Grading}
5821 \def\prob@pt@kw{pt}
5822 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
5823 \AddToHook{begindocument}{
5824   \ltx@ifpackageloaded{babel}{
5825     \makeatletter
5826     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5827     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5828       \input{problem-ngerman.ldf}
5829     }
5830     \clist_if_in:NnT \l_tmpa_clist {finnish}{
5831       \input{problem-finnish.ldf}
5832     }
5833     \clist_if_in:NnT \l_tmpa_clist {french}{
5834       \input{problem-french.ldf}
5835     }
5836     \clist_if_in:NnT \l_tmpa_clist {russian}{
5837       \input{problem-russian.ldf}
5838     }
5839     \makeatother
5840   }{ }
5841 }
```

## 40.2 Problems and Solutions

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

```
5842 \keys_define:nn{ problem / problem }{
5843   id      .str_set_x:N = \l__problems_prob_id_str,
5844   pts     .tl_set:N    = \l__problems_prob_pts_tl,
5845   min     .tl_set:N    = \l__problems_prob_min_tl,
5846   title   .tl_set:N    = \l__problems_prob_title_tl,
5847   type    .tl_set:N    = \l__problems_prob_type_tl,
5848   refnum  .int_set:N   = \l__problems_prob_refnum_int
5849 }
5850 \cs_new_protected:Nn \__problems_prob_args:n {
```

```

5851 \str_clear:N \l__problems_prob_id_str
5852 \tl_clear:N \l__problems_prob_pts_tl
5853 \tl_clear:N \l__problems_prob_min_tl
5854 \tl_clear:N \l__problems_prob_title_tl
5855 \tl_clear:N \l__problems_prob_type_tl
5856 \int_zero_new:N \l__problems_prob_refnum_int
5857 \keys_set:nn { problem / problem }{ #1 }
5858 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
5859   \let\l__problems_prob_refnum_int\undefined
5860 }
5861 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

5862 \newcounter{problem}
5863 \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.

```

5864 \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

```

5865 \newcommand\prob@number{
5866   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
5867     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
5868   }{
5869     \int_if_exist:NTF \l__problems_prob_refnum_int {
5870       \prob@label{\int_use:N \l__problems_prob_refnum_int }
5871     }{
5872       \prob@label\theproblem
5873     }
5874   }
5875 }

```

(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.

```

5876 \newcommand\prob@title[3]{%
5877   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
5878     #2 \l__problems_inclprob_title_tl #3
5879   }{
5880     \tl_if_exist:NTF \l__problems_prob_title_tl {
5881       #2 \l__problems_prob_title_tl #3
5882     }{
5883       #1
5884     }
5885   }
5886 }

```

(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.

```
5887 \def\prob@heading{
5888   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
5889   %\sref@label{id}\prob@problem@kw~\prob@number}{~}
5890 }
```

(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`

```
5891 \newenvironment{sproblem}[1][]{
5892   \__problems_prob_args:n{#1}%\sref@target%
5893   \@in@omtexttrue% we are in a statement (for inline definitions)
5894   \stepcounter{problem}\record@problem
5895   \def\current@section@level{\prob@problem@kw}
5896   \tl_if_exist:NTF \l__problems_inclprob_type_tl {
5897     \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
5898   }{
5899     \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
5900   }
5901   \str_if_exist:NTF \l__problems_inclprob_id_str {
5902     \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
5903   }{
5904     \str_set_eq:NN \sproblemid \l__problems_prob_id_str
5905   }
5906
5907
5908   \clist_set:No \l_tmpa_clist \sproblemtype
5909   \tl_clear:N \l_tmpa_tl
5910   \clist_map_inline:Nn \l_tmpa_clist {
5911     \tl_if_exist:cT {\__problems_sproblem_##1_start:}{
5912       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_start:}}
5913     }
5914   }
5915   \tl_if_empty:NTF \l_tmpa_tl {
5916     \__problems_sproblem_start:
5917   }{
5918     \l_tmpa_tl
5919   }
5920   \stex_ref_new_doc_target:n \sproblemid
5921 }{
5922   \clist_set:No \l_tmpa_clist \sproblemtype
5923   \tl_clear:N \l_tmpa_tl
5924   \clist_map_inline:Nn \l_tmpa_clist {
5925     \tl_if_exist:cT {\__problems_sproblem_##1_end:}{
5926       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_end:}}
5927     }
5928   }
```

```

5928 }
5929 \tl_if_empty:NTF \l_tmpa_tl {
5930   \__problems_sproblem_end:
5931 }{
5932   \l_tmpa_tl
5933 }
5934
5935
5936 \smallskip
5937 }
5938
5939
5940 \cs_new_protected:Nn \__problems_sproblem_start: {
5941   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
5942 }
5943 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
5944
5945 \newcommand\stexpatchproblem[3][] {
5946   \str_set:Nx \l_tmpa_str{ #1 }
5947   \str_if_empty:NTF \l_tmpa_str {
5948     \tl_set:Nn \__problems_sproblem_start: { #2 }
5949     \tl_set:Nn \__problems_sproblem_end: { #3 }
5950   }{
5951     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
5952     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
5953   }
5954 }
5955
5956
5957 \bool_if:NT \c__problems_boxed_bool {
5958   \surroundwithmdframed{problem}
5959 }

```

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

```

5960 \def\record@problem{
5961   \protected@write\@auxout{}
5962   {
5963     \string\@problem{\prob@number}
5964     {
5965       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
5966         \l__problems_inclprob_pts_tl
5967       }{
5968         \l__problems_prob_pts_tl
5969       }
5970     }%
5971     {
5972       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
5973         \l__problems_inclprob_min_tl
5974       }{
5975         \l__problems_prob_min_tl
5976       }
5977     }
5978   }
5979 }

```

(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).

```
5980 \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.

```
5981 \keys_define:nn { problem / solution }{
5982   id                .str_set_x:N = \l__problems_solution_id_str ,
5983   for               .tl_set:N    = \l__problems_solution_for_tl ,
5984   height            .dim_set:N   = \l__problems_solution_height_dim ,
5985   creators          .clist_set:N = \l__problems_solution_creators_clist ,
5986   contributors      .clist_set:N = \l__problems_solution_contributors_clist ,
5987   srccite           .tl_set:N    = \l__problems_solution_srccite_tl
5988 }
5989 \cs_new_protected:Nn \__problems_solution_args:n {
5990   \str_clear:N \l__problems_solution_id_str
5991   \tl_clear:N \l__problems_solution_for_tl
5992   \tl_clear:N \l__problems_solution_srccite_tl
5993   \clist_clear:N \l__problems_solution_creators_clist
5994   \clist_clear:N \l__problems_solution_contributors_clist
5995   \dim_zero:N \l__problems_solution_height_dim
5996   \keys_set:nn { problem / solution }{ #1 }
5997 }
```

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

```
5998 \newcommand\@startsolution[1][{}]{
5999   \__problems_solution_args:n { #1 }
6000   \@in@omtexttrue% we are in a statement.
6001   \bool_if:NF \c__problems_boxed_bool { \hrule }
6002   \smallskip\noindent
6003   {\textbf\prob@solution@kw : \enspace}
6004   \begin{small}
6005   \def\current@section@level{\prob@solution@kw}
6006   \ignorespacesandpars
6007 }
```

**\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.

```
6008 \newcommand\startsolutions{
6009   \specialcomment{solution}{\@startsolution}{
6010     \bool_if:NF \c__problems_boxed_bool {
6011       \hrule\medskip
6012     }
6013     \end{small}%
6014   }
6015   \bool_if:NT \c__problems_boxed_bool {
6016     \surroundwithmdframed{solution}
6017   }
6018 }
```

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

\stopsolutions

```
6019 \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.

```
6020 \ifsolutions
6021 \startsolutions
6022 \else
6023 \stopsolutions
6024 \fi
```

exnote

```
6025 \bool_if:NTF \c__problems_notes_bool {
6026 \newenvironment{exnote}[1][]{
6027 \par\smallskip\hrule\smallskip
6028 \noindent\textbf{\prob@note@kw : }\small
6029 }{
6030 \smallskip\hrule
6031 }
6032 }{
6033 \excludecomment{exnote}
6034 }
```

hint

```
6035 \bool_if:NTF \c__problems_notes_bool {
6036 \newenvironment{hint}[1][]{
6037 \par\smallskip\hrule\smallskip
6038 \noindent\textbf{\prob@hint@kw :~ }\small
6039 }{
6040 \smallskip\hrule
6041 }
6042 \newenvironment{exhint}[1][]{
6043 \par\smallskip\hrule\smallskip
6044 \noindent\textbf{\prob@hint@kw :~ }\small
6045 }{
6046 \smallskip\hrule
6047 }
6048 }{
6049 \excludecomment{hint}
6050 \excludecomment{exhint}
6051 }
```

gnote

```
6052 \bool_if:NTF \c__problems_notes_bool {
6053 \newenvironment{gnote}[1][]{
6054 \par\smallskip\hrule\smallskip
6055 \noindent\textbf{\prob@gnote@kw : }\small
6056 }{
6057 \smallskip\hrule
6058 }
6059 }{
6060 \excludecomment{gnote}
6061 }
```



## 40.3 Multiple Choice Blocks

EdN:23

mcb 23

```
6062 \newenvironment{mcb}{
6063   \begin{enumerate}
6064 }{
6065   \end{enumerate}
6066 }
```

we define the keys for the mcc macro

```
6067 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
6068   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
6069     \bool_set_true:N #1
6070   }{
6071     \bool_set_false:N #1
6072   }
6073 }
6074 \keys_define:nn { problem / mcc }{
6075   id          .str_set:N = \l__problems_mcc_id_str ,
6076   feedback    .tl_set:N   = \l__problems_mcc_feedback_tl ,
6077   T           .default:n   = { true } ,
6078   T           .bool_set:N  = \l__problems_mcc_t_bool ,
6079   F           .default:n   = { true } ,
6080   F           .bool_set:N  = \l__problems_mcc_f_bool ,
6081   Ttext       .code:n      = {
6082     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
6083   } ,
6084   Ftext       .code:n      = {
6085     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
6086   }
6087 }
6088 \cs_new_protected:Nn \l__problems_mcc_args:n {
6089   \str_clear:N \l__problems_mcc_id_str
6090   \tl_clear:N \l__problems_mcc_feedback_tl
6091   \bool_set_true:N \l__problems_mcc_t_bool
6092   \bool_set_true:N \l__problems_mcc_f_bool
6093   \bool_set_true:N \l__problems_mcc_Ttext_bool
6094   \bool_set_false:N \l__problems_mcc_Ftext_bool
6095   \keys_set:nn { problem / mcc }{ #1 }
6096 }
```

\mcc

```
6097 \newcommand\mcc[2][]{
6098   \l__problems_mcc_args:n{ #1 }
6099   \item #2
6100   \ifsolutions
6101     \\\
6102     \bool_if:NT \l__problems_mcc_t_bool {
6103       % TODO!
6104       % \ifcsstring{mcc@T}{T}{\mcc@Ttext}%
6105     }
6106     \bool_if:NT \l__problems_mcc_f_bool {
```

---

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

```

6107         % TODO!
6108         % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
6109     }
6110     \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
6111         !
6112     }{
6113         \l__problems_mcc_feedback_tl
6114     }
6115     \fi
6116 } %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.

```

6117
6118 \keys_define:nn{ problem / inclproblem }{
6119     id      .str_set_x:N = \l__problems_inclprob_id_str,
6120     pts     .tl_set:N    = \l__problems_inclprob_pts_tl,
6121     min     .tl_set:N    = \l__problems_inclprob_min_tl,
6122     title   .tl_set:N    = \l__problems_inclprob_title_tl,
6123     refnum  .int_set:N   = \l__problems_inclprob_refnum_int,
6124     type    .tl_set:N    = \l__problems_inclprob_type_tl,
6125     mhrepos .str_set_x:N = \l__problems_inclprob_mhrepos_str
6126 }
6127 \cs_new_protected:Nn \l__problems_inclprob_args:n {
6128     \str_clear:N \l__problems_prob_id_str
6129     \tl_clear:N \l__problems_inclprob_pts_tl
6130     \tl_clear:N \l__problems_inclprob_min_tl
6131     \tl_clear:N \l__problems_inclprob_title_tl
6132     \tl_clear:N \l__problems_inclprob_type_tl
6133     \int_zero_new:N \l__problems_inclprob_refnum_int
6134     \str_clear:N \l__problems_inclprob_mhrepos_str
6135     \keys_set:nn { problem / inclproblem }{ #1 }
6136     \tl_if_empty:NT \l__problems_inclprob_pts_tl {
6137         \let\l__problems_inclprob_pts_tl\undefined
6138     }
6139     \tl_if_empty:NT \l__problems_inclprob_min_tl {
6140         \let\l__problems_inclprob_min_tl\undefined
6141     }
6142     \tl_if_empty:NT \l__problems_inclprob_title_tl {
6143         \let\l__problems_inclprob_title_tl\undefined
6144     }
6145     \tl_if_empty:NT \l__problems_inclprob_type_tl {
6146         \let\l__problems_inclprob_type_tl\undefined
6147     }
6148     \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
6149         \let\l__problems_inclprob_refnum_int\undefined
6150     }
6151 }

```

```

6152
6153 \cs_new_protected:Nn \__problems_inclprob_clear: {
6154   \let\l__problems_inclprob_id_str\undefined
6155   \let\l__problems_inclprob_pts_tl\undefined
6156   \let\l__problems_inclprob_min_tl\undefined
6157   \let\l__problems_inclprob_title_tl\undefined
6158   \let\l__problems_inclprob_type_tl\undefined
6159   \let\l__problems_inclprob_refnum_int\undefined
6160   \let\l__problems_inclprob_mhrepos_str\undefined
6161 }
6162 \__problems_inclprob_clear:
6163
6164 \newcommand\includeproblem[2][ ]{
6165   \__problems_inclprob_args:n{ #1 }
6166   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
6167     \input{#2}
6168   }{
6169     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
6170       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
6171     }
6172   }
6173   \__problems_inclprob_clear:
6174 }

```

(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.

```

6175 \AddToHook{enddocument}{
6176   \bool_if:NT \c__problems_pts_bool {
6177     \message{Total:~\arabic{pts}~points}
6178   }
6179   \bool_if:NT \c__problems_min_bool {
6180     \message{Total:~\arabic{min}~minutes}
6181   }
6182 }

```

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

```

6183 \def\pts#1{
6184   \bool_if:NT \c__problems_pts_bool {
6185     \marginpar{#1~\prob@pt@kw}
6186   }
6187 }
6188 \def\min#1{
6189   \bool_if:NT \c__problems_min_bool {
6190     \marginpar{#1~\prob@min@kw}
6191   }
6192 }

```

`\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.

```

6193 \newcounter{pts}
6194 \def\show@pts{
6195   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
6196     \bool_if:NT \c__problems_pts_bool {
6197       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
6198       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
6199     }
6200   }{
6201     \tl_if_exist:NT \l__problems_prob_pts_tl {
6202       \bool_if:NT \c__problems_pts_bool {
6203         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
6204         \addtocounter{pts}{\l__problems_prob_pts_tl}
6205       }
6206     }
6207   }
6208 }

```

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

`\show@min`

```

6209 \newcounter{min}
6210 \def\show@min{
6211   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
6212     \bool_if:NT \c__problems_min_bool {
6213       \marginpar{\l__problems_inclprob_min_tl\ min}
6214       \addtocounter{min}{\l__problems_inclprob_min_tl}
6215     }
6216   }{
6217     \tl_if_exist:NT \l__problems_prob_min_tl {
6218       \bool_if:NT \c__problems_min_bool {
6219         \marginpar{\l__problems_prob_min_tl\ min}
6220         \addtocounter{min}{\l__problems_prob_min_tl}
6221       }
6222     }
6223   }
6224 }
6225 \</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.

```
6226 \@@=hwexam>
6227 \*cls>
6228 \ProvidesExplClass{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
6229 \RequirePackage{l3keys2e,expl-keystr-compatible}
6230 \DeclareOption*{
6231   \PassOptionsToClass{\CurrentOption}{document-structure}
6232   \PassOptionsToPackage{\CurrentOption}{stex}
6233   \PassOptionsToPackage{\CurrentOption}{hwexam}
6234   \PassOptionsToPackage{\CurrentOption}{tikzinput}
6235 }
6236 \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.

```
6237 \LoadClass{document-structure}
6238 \RequirePackage{stex}
6239 \RequirePackage{hwexam}
6240 \RequirePackage{tikzinput}
6241 \RequirePackage{graphicx}
6242 \RequirePackage{a4wide}
6243 \RequirePackage{amssymb}
6244 \RequirePackage{amstext}
6245 \RequirePackage{amsmath}
```

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

```

6246 \newcommand\assig@default@type{\hwexam@assignment@kw}
6247 \def\document@hwexamtype{\assig@default@type}
6248 <@@=document_structure>
6249 \keys_define:nn { document-structure / document }{
6250 id .str_set_x:N = \c_document_structure_document_id_str,
6251 hwexamtype .tl_set:N = \document@hwexamtype
6252 }
6253 <@@=hwexam>
6254 </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.

```
6255 \*package>
6256 \ProvidesExplPackage{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
6257 \RequirePackage{l3keys2e,expl-keystr-compat}
6258
6259 \newif\iftest\testfalse
6260 \DeclareOption{test}{\testtrue}
6261 \newif\ifmultiple\multiplefalse
6262 \DeclareOption{multiple}{\multipletrue}
6263 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
6264 \ProcessOptions
```

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

```
6265 \RequirePackage{keyval}[1997/11/10]
6266 \RequirePackage{problem}
```

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

```
6267 \newcommand\hwexam@assignment@kw{Assignment}
6268 \newcommand\hwexam@given@kw{Given}
6269 \newcommand\hwexam@due@kw{Due}
6270 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
6271 blank~for~extra~space}
6272 \def\hwexam@minutes@kw{minutes}
6273 \newcommand\correction@probs@kw{prob.}
6274 \newcommand\correction@pts@kw{total}
6275 \newcommand\correction@reached@kw{reached}
6276 \newcommand\correction@sum@kw{Sum}
6277 \newcommand\correction@grade@kw{grade}
6278 \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

```

6279 \AddToHook{begindocument}{
6280 \ltx@ifpackageloaded{babel}{
6281 \makeatletter
6282 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6283 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6284 \input{hwexam-ngerman.ldf}
6285 }
6286 \clist_if_in:NnT \l_tmpa_clist {finnish}{
6287 \input{hwexam-finnish.ldf}
6288 }
6289 \clist_if_in:NnT \l_tmpa_clist {french}{
6290 \input{hwexam-french.ldf}
6291 }
6292 \clist_if_in:NnT \l_tmpa_clist {russian}{
6293 \input{hwexam-russian.ldf}
6294 }
6295 \makeatother
6296 }{}
6297 }
6298

```

## 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.

```

6299 \newcounter{assignment}
6300 \numberproblemsin{assignment}
6301 \renewcommand\prob@label[1]{\assignment@number.#1}

```

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

```

6302 \keys_define:nn { hwexam / assignment } {
6303 id .str_set:N = \l__hwexam_assign_id_str,
6304 number .int_set:N = \l__hwexam_assign_number_int,
6305 title .tl_set:N = \l__hwexam_assign_title_tl,
6306 type .tl_set:N = \l__hwexam_assign_type_tl,
6307 given .tl_set:N = \l__hwexam_assign_given_tl,
6308 due .tl_set:N = \l__hwexam_assign_due_tl,
6309 loadmodules .code:n = {
6310 \bool_set_true:N \l__hwexam_assign_loadmodules_bool
6311 }
6312 }
6313 \cs_new_protected:Nn \__hwexam_assignment_args:n {
6314 \str_clear:N \l__hwexam_assign_id_str
6315 \int_set:Nn \l__hwexam_assign_number_int {-1}
6316 \tl_clear:N \l__hwexam_assign_title_tl
6317 \tl_clear:N \l__hwexam_assign_type_tl
6318 \tl_clear:N \l__hwexam_assign_given_tl
6319 \tl_clear:N \l__hwexam_assign_due_tl
6320 \bool_set_false:N \l__hwexam_assign_loadmodules_bool

```



```

6321 \keys_set:nn { hwexam / assignment }{ #1 }
6322 }

```

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.

```

6323 \newcommand\given@due[2]{
6324 \bool_lazy_all:nF {
6325 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
6326 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
6327 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
6328 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
6329 }{ #1 }
6330
6331 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
6332 \tl_if_empty:NF \l__hwexam_assign_given_tl {
6333 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
6334 }
6335 }{
6336 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
6337 }
6338
6339 \bool_lazy_or:nnF {
6340 \bool_lazy_and_p:nn {
6341 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
6342 }{
6343 \tl_if_empty_p:V \l__hwexam_assign_due_tl
6344 }
6345 }{
6346 \bool_lazy_and_p:nn {
6347 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
6348 }{
6349 \tl_if_empty_p:V \l__hwexam_assign_due_tl
6350 }
6351 }{ ,~ }
6352
6353 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
6354 \tl_if_empty:NF \l__hwexam_assign_due_tl {
6355 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
6356 }
6357 }{
6358 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
6359 }
6360
6361 \bool_lazy_all:nF {
6362 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
6363 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
6364 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
6365 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
6366 }{ #2 }
6367 }

```

`\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.

```

6368 \newcommand\assignment@title[3]{
6369 \tl_if_empty:NTF \l__hwexam_inclasssign_title_tl {
6370 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
6371 #1
6372 }{
6373 #2\l__hwexam_assign_title_tl#3
6374 }
6375 }{
6376 #2\l__hwexam_inclasssign_title_tl#3
6377 }
6378 }

```

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

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

```

6379 \newcommand\assignment@number{
6380 \int_compare:nNnTF \l__hwexam_inclasssign_number_int = {-1} {
6381 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
6382 \arabic{assignment}
6383 } {
6384 \int_use:N \l__hwexam_assign_number_int
6385 }
6386 }{
6387 \int_use:N \l__hwexam_inclasssign_number_int
6388 }
6389 }

```

(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.

```

6390 \newenvironment{assignment}[1][ ]{
6391 \__hwexam_assignment_args:n { #1 }
6392 %\sref@target
6393 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
6394 \global\stepcounter{assignment}
6395 }{
6396 \global\setcounter{assignment}{\int_use:N\l__hwexam_assign_number_int}
6397 }
6398 \setcounter{problem}{0}
6399 \def\current@section@level{\document@hwexamtype}
6400 %\sref@label@id{\document@hwexamtype \thesection}
6401 \begin{@assignment}
6402 }{
6403 \end{@assignment}
6404 }

```

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

```

6405 \def\ass@title{
6406 \protect\document@hwexamtype~\arabic{assignment}
6407 \assignment@title{}\{;\}\{;\} -- \given@due{}\{;\}
6408 }
6409 \ifmultiple
6410 \newenvironment{@assignment}{
6411 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
6412 \begin{omgroup}[loadmodules]{\ass@title}
6413 }{
6414 \begin{omgroup}{\ass@title}
6415 }
6416 }{
6417 \end{omgroup}
6418 }

```

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

```

6419 \else
6420 \newenvironment{@assignment}{
6421 \begin{center}\bf
6422 \Large@title\strut\\
6423 \document@hwexamtype~\arabic{assignment}\assignment@title{}\{;\}\{;\}\{;\}
6424 \large\given@due{--;\}\{;\}--}
6425 \end{center}
6426 }{}
6427 \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.

```

6428 \keys_define:nn { hwexam / inclassignment } {
6429 %id .str_set_x:N = \l__hwexam_assign_id_str,
6430 number .int_set:N = \l__hwexam_inclassign_number_int,
6431 title .tl_set:N = \l__hwexam_inclassign_title_tl,
6432 type .tl_set:N = \l__hwexam_inclassign_type_tl,
6433 given .tl_set:N = \l__hwexam_inclassign_given_tl,
6434 due .tl_set:N = \l__hwexam_inclassign_due_tl,
6435 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
6436 }
6437 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
6438 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
6439 \tl_clear:N \l__hwexam_inclassign_title_tl
6440 \tl_clear:N \l__hwexam_inclassign_type_tl
6441 \tl_clear:N \l__hwexam_inclassign_given_tl
6442 \tl_clear:N \l__hwexam_inclassign_due_tl
6443 \str_clear:N \l__hwexam_inclassign_mhrepos_str
6444 \keys_set:nn { hwexam / inclassignment }{ #1 }
6445 }
6446 \__hwexam_inclassignment_args:n {}
6447
6448 \newcommand\inputassignment[2][{}]{

```

```

6449 \_hwexam_inclassnment_args:n { #1 }
6450 \str_if_empty:NTF \l__hwexam_inclassn_mhrepos_str {
6451 \input{#2}
6452 }{
6453 \stex_in_repository:nn{\l__hwexam_inclassn_mhrepos_str}{
6454 \input{\mhp{path}\l__hwexam_inclassn_mhrepos_str}{#2}}
6455 }
6456 }
6457 \_hwexam_inclassnment_args:n {}
6458 }
6459 \newcommand\includeassignment[2][]{
6460 \newpage
6461 \inputassignment[#1]{#2}
6462 }

```

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

## 42.4 Typesetting Exams

\quizheading

```

6463 \ExplSyntaxOff
6464 \newcommand\quizheading[1]{%
6465 \def\@tas{#1}%
6466 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
6467 \ifx\@tas\@empty\else%
6468 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
6469 \fi%
6470 }
6471 \ExplSyntaxOn

```

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

\testheading

```

6472
6473 \def\hwexamheader{\input{hwexam-default.header}}
6474
6475 \def\hwexamminutes{
6476 \tl_if_empty:NTF \testheading@duration {
6477 {\testheading@min}~\hwexam@minutes@kw
6478 }{
6479 \testheading@duration
6480 }
6481 }
6482
6483 \keys_define:nn { hwexam / testheading } {
6484 min .tl_set:N = \testheading@min,
6485 duration .tl_set:N = \testheading@duration,
6486 reqpts .tl_set:N = \testheading@reqpts,
6487 tools .tl_set:N = \testheading@tools
6488 }
6489 \cs_new_protected:Nn \_hwexam_testheading_args:n {
6490 \tl_clear:N \testheading@min
6491 \tl_clear:N \testheading@duration

```

```

6492 \tl_clear:N \testheading@reqpts
6493 \tl_clear:N \testheading@tools
6494 \keys_set:nn { hwexam / testheading }{ #1 }
6495 }
6496 \newenvironment{testheading}[1][]{
6497   \__hwexam_testheading_args:n{ #1 }
6498   \newcount\check@time\check@time=\testheading@min
6499   \advance\check@time by -\theassignment@totalmin
6500   \newif\if@bonuspoints
6501   \tl_if_empty:NTF \testheading@reqpts {
6502     \@bonuspointsfalse
6503   }{
6504     \newcount\bonus@pts
6505     \bonus@pts=\theassignment@totalpts
6506     \advance\bonus@pts by -\testheading@reqpts
6507     \edef\bonus@pts{\the\bonus@pts}
6508     \@bonuspointstrue
6509   }
6510   \edef\check@time{\the\check@time}
6511
6512   \makeatletter\hwexamheader\makeatother
6513 }{
6514   \newpage
6515 }

```

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

\testspace

```

6516 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

6517 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

6518 \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.

```

6519 \<@=problems>
6520 \renewcommand\@problem[3]{
6521   \stepcounter{assignment@probs}
6522   \def\__problemspts{#2}
6523   \ifx\__problemspts\@empty\else
6524     \addtocounter{assignment@totalpts}{#2}
6525   \fi
6526   \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
6527   \xdef\correction@probs{\correction@probs & #1}%
6528   \xdef\correction@pts{\correction@pts & #2}
6529   \xdef\correction@reached{\correction@reached &}

```

```

6530 }
6531 <@@=hwexam>

```

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

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

```

6532 \newcounter{assignment@probs}
6533 \newcounter{assignment@totalpts}
6534 \newcounter{assignment@totalmin}
6535 \def\correction@probs{\correction@probs@kw}
6536 \def\correction@pts{\correction@pts@kw}
6537 \def\correction@reached{\correction@reached@kw}
6538 \stepcounter{assignment@probs}
6539 \newcommand\correction@table{
6540 \resizebox{\textwidth}{!}{%
6541 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
6542 &\multicolumn{\theassignment@probs}{c|}|%|
6543 {\footnotesize\correction@forgrading@kw} &\\ \hline
6544 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
6545 \correction@pts & \theassignment@totalpts & \\ \hline
6546 \correction@reached & & \[.7cm]\hline
6547 \end{tabular}}
6548 </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}

```