

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

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

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

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

---

\*Version 3.0 (last revised 2022-02-16)

# 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>99</b>
28.1	The module environment . . . . .	102
28.2	Invoking modules . . . . .	108
<b>29</b>	<b>STeX-Module Inheritance Implementation</b>	<b>110</b>
29.1	SMS Mode . . . . .	110
29.2	Inheritance . . . . .	113
<b>30</b>	<b>STeX-Symbols Implementation</b>	<b>118</b>
30.1	Symbol Declarations . . . . .	118
30.2	Notations . . . . .	125
<b>31</b>	<b>STeX-Terms Implementation</b>	<b>136</b>
31.1	Symbol Invocations . . . . .	136
31.2	Terms . . . . .	139
31.3	Notation Components . . . . .	146
<b>32</b>	<b>STeX-Structural Features Implementation</b>	<b>149</b>
32.1	Imports with modification . . . . .	149
32.2	The feature environment . . . . .	156
32.3	Features . . . . .	158
<b>33</b>	<b>STeX-Statements Implementation</b>	<b>163</b>
33.1	Definitions . . . . .	163
33.2	Assertions . . . . .	168
33.3	Examples . . . . .	170
33.4	Logical Paragraphs . . . . .	173

<b>34 The Implementation</b>	<b>177</b>
34.1 Package Options . . . . .	177
34.2 Proofs . . . . .	177
34.3 Justifications . . . . .	183
<b>35 <math>\text{\LaTeX}</math>-Others Implementation</b>	<b>185</b>
<b>36 <math>\text{\LaTeX}</math>-Metatheory Implementation</b>	<b>186</b>
<b>37 Tikzinput Implementation</b>	<b>189</b>
<b>38 document-structure.sty Implementation</b>	<b>191</b>
38.1 The document-structure Class . . . . .	191
38.2 Class Options . . . . .	191
38.3 Beefing up the <code>document</code> environment . . . . .	192
38.4 Implementation: document-structure Package . . . . .	192
38.5 Package Options . . . . .	192
38.6 Document Structure . . . . .	194
38.7 Front and Backmatter . . . . .	197
38.8 Global Variables . . . . .	199
<b>39 NotesSlides – Implementation</b>	<b>200</b>
39.1 Class and Package Options . . . . .	200
39.2 Notes and Slides . . . . .	202
39.3 Header and Footer Lines . . . . .	206
39.4 Frame Images . . . . .	207
39.5 Colors and Highlighting . . . . .	208
39.6 Sectioning . . . . .	209
39.7 Excursions . . . . .	211
<b>40 The Implementation</b>	<b>213</b>
40.1 Package Options . . . . .	213
40.2 Problems and Solutions . . . . .	214
40.3 Multiple Choice Blocks . . . . .	220
40.4 Including Problems . . . . .	221
40.5 Reporting Metadata . . . . .	222
<b>41 Implementation: The hwexam Class</b>	<b>224</b>
41.1 Class Options . . . . .	224
<b>42 Implementation: The hwexam Package</b>	<b>226</b>
42.1 Package Options . . . . .	226
42.2 Assignments . . . . .	227
42.3 Including Assignments . . . . .	230
42.4 Typesetting Exams . . . . .	231
42.5 Leftovers . . . . .	233

**Part I**  
**Manual**

# Chapter 1

## What is sTeX?

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

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

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

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

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

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



# Chapter 2

## Quickstart

### 2.1 Setup

#### 2.1.1 The sTeX IDE

TODO: VSCode Plugin

#### 2.1.2 Manual Setup

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

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

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

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

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

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

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

---

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

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

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

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

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

The document we will consider is the following:

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

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

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

Compiling this document with `pdflatex` should yield the output

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

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

### \usemodule

The command `\usemodule[some/archive]{modulename}` finds some module in the appropriate archive – in the first case (`\usemodule[smglom/calculus]{series}`), s<sub>T</sub>E<sub>X</sub> looks for the archive `smglom/calculus` in our local MathHub-directory (see [chapter 4](#)), and in its source-folder for a file `series.tex`. Since no such file exists, and by default the document is assumed to be in *english*, it picks the file `series.en.tex`, and indeed, in here we find a statement `\begin{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/arithmetics,smglom/sets,smglom/topology,
              smglom/mv,smglom/linear-algebra,smglom/algebra
responsible: Michael.Kohlhase@FAU.de
title: Elementary Calculus
teaser: Terminology for the mathematical study of change.
description: desc.html
```

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

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

`source-base` or

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

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

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

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

## Chapter 5

# Creating New Modules and Symbols

TODO

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

---

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

### 8.1.1 Semantic Macros and Notations

Semantic macros invoke a formally declared symbol.

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

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

Multiplying again by  $b$  yields...

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

### Example 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.  $\int \int \int f(x,y,z) dx dy dz$ ?

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

## Precedences

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

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

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

$\text{\TeX}$  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{\textbackslash plus}\{a\}\{\text{\textbackslash times}\{b\}\{c\}\}\$ and  $\text{\textbackslash times}\{a\}\{\text{\textbackslash plus}\{b\}\{c\}\}\$$$ 
```

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

## 8.1.2 Archives and Imports

### Namespaces

Ideally,  $\text{\TeX}$  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,  $\text{\TeX}$  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  $\text{\TeX}$  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 {&lt;property&gt;} {&lt;resource&gt;} {&lt;content&gt;}</code>
<code>\stex_annotate_invisible:nnn</code>	
<code>\stex_annotate_invisible:n</code>	

---

Annotates the HTML generated by `<content>` with

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

`\stex_annotate_invisible:n` adds the attributes

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

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

<code>stex_annotate_env</code>	<code>\begin{stex_annotate_env}{&lt;property&gt;}{&lt;resource&gt;}</code> <code>&lt;content&gt;</code> <code>\end{stex_annotate_env}</code> behaves like <code>\stex_annotate:nnn {&lt;property&gt;} {&lt;resource&gt;} {&lt;content&gt;}</code> .
--------------------------------	--

---

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

---

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

---

<code>\stex_deactivate_macro:Nn</code>	<code>\stex_deactivate_macro:Nn&lt;cs&gt;{&lt;environments&gt;}</code>
<code>\stex_reactivate_macro:N</code>	

---

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

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

---

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

---

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

# Chapter 10

## sTEX-MathHub

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

### 10.1 Macros and Environments

---

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

---

#### 10.1.1 Files, Paths, URIs

---

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

---

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

---

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

---

---

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

---

---

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

---

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

---

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

---

`\g_stex_currentfile_seq`

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

**Test 1**

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

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

**10.1.2 MathHub Archives**

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

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

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

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

`\l_stex_current_repository_prop`

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

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

## Test 2

```

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

```

```

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

```

## Chapter 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?collection>,  
<http://mathhub.info/sTeX?Metatheory?aseqfromto, http://mathhub.info/sTeX?Metatheory?aseqfromtovia, http://mathhub.info/sTeX?Metatheory?collection>,  
<http://mathhub.info/sTeX?Metatheory?module-type, http://mathhub.info/sTeX?Metatheory?mathematical-structure, http://mathhub.info/sTeX?Metatheory?collection>,  
<http://mathhub.info/sTeX?Metatheory?isa, http://mathhub.info/sTeX?Metatheory?bind, http://mathhub.info/sTeX?Metatheory?dummyvar, http://mathhub.info/sTeX?Metatheory?collection>,  
<http://mathhub.info/sTeX?Metatheory?seqtype, http://mathhub.info/sTeX?Metatheory?sequence-index, http://mathhub.info/sTeX?Metatheory?collection>,  
<http://mathhub.info/sTeX?Metatheory?aseqfromto, http://mathhub.info/sTeX?Metatheory?aseqfromtovia, http://mathhub.info/sTeX?Metatheory?collection>,  
<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}<

---

---

`\stex_import_module_uri:nn`

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

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

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

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

(a) If `<path>` is empty, then `<name>` must have been declared earlier in the same file and retrievable from `\g_stex_modules_in_file_seq`, or a file with name `<name>.<lang>.tex` must exist in the same folder, containing a module `<name>`.

That module should have the same namespace as the current one.

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

2. Otherwise:

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

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

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

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

---

---

`\stex_import_require_module:nnnn`

`{<ns>} {<archive-ID>} {<path>} {<name>}`

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

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



# Chapter 14

## TeX-Symbols

Code related to symbol declarations and notations

### 14.1 Macros and Environments

---

<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\mathbb{N}}{x\geq 0}`.

---

---

`\stex_symdecl_do:n`

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

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

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

### Test 11

```
\begin{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}_ {\com
```

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},\frac{ac}{b}}}$
$ [\plus{a,\mult{b,c}}]\text{ and } \mult{a,\plus{\frac{ab}{c},\frac{ac}{b}}}
$ \displaystyle \plus{a,\mult{b,c}}$ and
\withbrackets[ ]{ $ \displaystyle
\mult{a,\plus{\frac{ab}{c},\frac{ac}{b}}}$ }
\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`

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

`\compemph@uri`

`\defemph`

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@uri`

`\symrefemph`

`\symrefemph@uri`

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

---

`\STEXinvisible`

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

---

`\ellipses`

TODO

## Chapter 16

# TeX-Structural Features

Code related to structural features

### 16.1 Macros and Environments

#### 16.1.1 Structures

`mathstructure` TODO

## Chapter 17

# sTeX-Statements

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

### 17.1 Macros and Environments

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



## Chapter 18

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

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

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

## Contents

## 18.1 Introduction

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

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

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

Example 1: A very explicit proof, marked up semantically

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

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

## 18.2 The User Interface

### 18.2.1 Package Options

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

### 18.2.2 Proofs and Proof steps

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

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

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

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

### 18.2.3 Justifications

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

#### 18.2.4 Proof Structure

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

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 L<sup>A</sup>T<sub>E</sub>X `\@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 S<sub>T</sub>E<sub>X</sub> 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  $\text{\S}$ TeX collection, a version of T<sub>E</sub>X/L<sup>A</sup>T<sub>E</sub>X that allows to markup T<sub>E</sub>X/L<sup>A</sup>T<sub>E</sub>X documents semantically without leaving the document format, essentially turning T<sub>E</sub>X/L<sup>A</sup>T<sub>E</sub>X into a document format for mathematical knowledge management (MKM).

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

### 21.1 Introduction

$\text{\S}$ TeX is a version of T<sub>E</sub>X/L<sup>A</sup>T<sub>E</sub>X that allows to markup T<sub>E</sub>X/L<sup>A</sup>T<sub>E</sub>X documents semantically without leaving the document format, essentially turning T<sub>E</sub>X/L<sup>A</sup>T<sub>E</sub>X into a document format for mathematical knowledge management (MKM). The package supports direct translation to the OMDOC format [Koh06]

The `document-structure` package supplies macros and environments that allow to label document fragments and to reference them later in the same document or in other documents. In essence, this enhances the document-as-trees model to documents-as-directed-acyclic-graphs (DAG) model. This structure can be used by MKM systems for added-value services, either directly from the  $\text{\S}$ TeX sources, or after translation. Currently, trans-document referencing provided by this package can only be used in the  $\text{\S}$ TeX collection.

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

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

## 21.2 The User Interface

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

### 21.2.1 Package and Class Options

The `document-structure` class accept the following options:

<code>class=&lt;name&gt;</code>	load <code>&lt;name&gt;.cls</code> instead of <code>article.cls</code>
<code>topsect=&lt;sect&gt;</code>	The top-level sectioning level; the default for <code>&lt;sect&gt;</code> is <code>section</code>
<code>showignores</code>	show the the contents of the <code>ignore</code> environment after all
<code>showmeta</code>	show the metadata; see <code>metakeys.sty</code>
<code>showmods</code>	show modules; see <code>modules.sty</code>
<code>extrefs</code>	allow external references; see <code>sref.sty</code>
<code>defindex</code>	index definienda; see <code>statements.sty</code>
<code>minimal</code>	for testing; do not load any $\text{\LaTeX}$ 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 $\text{\LaTeX}$ ML transformation.
	The structure of the document is given by the <code>omgroup</code> environment just like in OM-
	DOC. In the $\text{\LaTeX}$ 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
<code>id</code>	metadata followed by a regular argument for the (section) title of the <code>omgroup</code> . The op-
<code>creators</code>	tional metadata argument has the keys <code>id</code> for an identifier, <code>creators</code> and <code>contributors</code>
<code>contributors</code>	for the Dublin Core metadata [DCM03]; see [Koh20a] for details of the format. The
<code>short</code>	<code>short</code> allows to give a short title for the generated section. If the title contains semantic
<code>loadmodules</code>	macros, they need to be protected by <code>\protect</code> , and we need to give the <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.bardderiv,loadmodules]{Introducing $\protect\bar$ Derivations}

```

<sup>9</sup>EdNOTE: integrate with `latexml`'s `XMRef` in the Math mode.

<sup>2</sup>We cannot patch the document environment to accept an optional argument, since other packages we load already do; pity.

`blindomgroup`

$\text{\TeX}$  automatically computes the sectioning level, from the nesting of `omgroup` environments. But sometimes, we want to skip levels (e.g. to use a `subsection*` as an introduction for a chapter). Therefore the `document-structure` package provides a variant `blindomgroup` that does not produce markup, but increments the sectioning level and logically groups document parts that belong together, but where traditional document markup relies on convention rather than explicit markup. The `blindomgroup` environment is useful e.g. for creating frontmatter at the correct level. Example 3 shows a typical setup for the outer document structure of a book with parts and chapters. We use two levels of `blindomgroup`:

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

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

Example 3: A typical Document Structure of a Book

`\skipomgroup`

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

`\currentsectionlevel`  
`\CurrentSectionLevel`

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

---

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

### 21.2.3 Ignoring Inputs

`ignore` The `ignore` environment can be used for hiding text parts from the document structure.  
`showignores` The body of the environment is not PDF or DVI output unless the `showignores` option is given to the `document-structure` class or `package`. But in the generated OMDoc result, the body is marked up with a `ignore` element. This is useful in two situations. For

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

**narrative/content markup** In  $\text{\TeX}$  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

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

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

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

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

### 25.2 Preliminaries

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

Package options:

```

26 \keys_define:nn { stex } {
27   debug      .clist_set:N = \c_stex_debug_clist ,
28   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 sTeXlogo:
\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
301 \protected\def\ignorespacesandpars{
302   \begingroup\catcode13=10\relax
303   \@ifnextchar\par{
304     \endgroup\expandafter\ignorespacesandpars\@gobble
305   }{
306     \endgroup
307   }
308 }
309
310
311 </package>
```

## Chapter 26

# STEX -MathHub Implementation

```
312 <*package>
313
314 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
315
316 <@@=stex_path>
317
318 Warnings and error messages
319 \msg_new:nnn{stex}{error/norepository}{
320   No~archive~#1~found~in~#2
321 }
322 \msg_new:nnn{stex}{error/notinarchive}{
323   Not~currently~in~an~archive,~but~\detokenize{#1}~
324   needs~one!
325 }
326 \msg_new:nnn{stex}{error/nofile}{
327   \detokenize{#1}~could~not~find~file~#2
328 }
329 \msg_new:nnn{stex}{error/twofiles}{
330   \detokenize{#1}~found~two~candidates~for~#2
331 }
332 }
```

### 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
330 \cs_new_protected:Nn \stex_path_from_string:Nn {
331   \str_set:Nx \l_tmpa_str { #2 }
332   \str_if_empty:NTF \l_tmpa_str {
333     \seq_clear:N #1
334   }{
335     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
336     \sys_if_platform_windows:T{
337       \seq_clear:N \l_tmpa_tl
```

```

338     \seq_map_inline:Nn #1 {
339       \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
340       \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
341     }
342     \seq_set_eq:NN #1 \l_tmpa_tl
343   }
344   \stex_path_canonicalize:N #1
345 }
346 }
347 \cs_generate_variant:Nn \stex_path_from_string:Nn
348 { 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`

```

349 \cs_new_protected:Nn \stex_path_to_string:NN {
350   \exp_args:Nne \str_set:Nn #2 { \seq_use:Nn #1 / }
351 }
352
353 \cs_new:Nn \stex_path_to_string:N {
354   \seq_use:Nn #1 /
355 }

```

(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`

```

356 \str_const:Nn \c__stex_path_dot_str {.}
357 \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.

```

358 \cs_new_protected:Nn \stex_path_canonicalize:N {
359   \seq_if_empty:NF #1 {
360     \seq_clear:N \l_tmpa_seq
361     \seq_get_left:NN #1 \l_tmpa_tl
362     \str_if_empty:NT \l_tmpa_tl {
363       \seq_put_right:Nn \l_tmpa_seq {}
364     }
365     \seq_map_inline:Nn #1 {
366       \str_set:Nn \l_tmpa_tl { ##1 }
367       \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_dot_str {} {
368         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
369           \seq_if_empty:NNTF \l_tmpa_seq {
370             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
371               \c__stex_path_up_str
372             }
373           }{
374             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
375             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
376               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
377                 \c__stex_path_up_str
378               }

```

```

379         }{
380         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
381         }
382     }
383     }{
384     \str_if_empty:NF \l_tmpa_tl {
385     \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
386     }
387     }
388 }
389 }
390 \seq_gset_eq:NN #1 \l_tmpa_seq
391 }
392 }

```

(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 $\underline{TF}$`

```

393 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
394   \seq_if_empty:NTF #1 {
395     \prg_return_false:
396   }{
397     \seq_get_left:NN #1 \l_tmpa_tl
398     \str_if_empty:NTF \l_tmpa_tl {
399       \prg_return_true:
400     }{
401       \prg_return_false:
402     }
403   }
404 }

```

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

## 26.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

405 \str_new:N\l_stex_kpsewhich_return_str
406 \cs_new_protected:Nn \stex_kpsewhich:n {
407   \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
408   \exp_args:NNo \str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
409   \tl_trim_spaces:N \l_stex_kpsewhich_return_str
410 }

```

(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`

```

411 \sys_if_platform_windows:TF{
412   \stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
413 }{
414   \stex_kpsewhich:n{-var-value~PWD}
415 }
416

```



```

417 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
418 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
419 \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

```

420 <@@=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

```

421 \seq_gclear_new:N\g__stex_files_stack

```

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`  
`\c_stex_mainfile_str`

```

422 \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
423 \stex_path_from_string:Nn \c_stex_mainfile_seq
424 \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`.

```

425 \seq_gclear_new:N\g_stex_currentfile_seq
426 \cs_new_protected:Nn \stex_filestack_push:n {
427   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
428   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
429     \stex_path_from_string:Nn\g_stex_currentfile_seq{
430       \c_stex_pwd_str/#1
431     }
432   }
433   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
434   \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
435 }
436 \cs_new_protected:Nn \stex_filestack_pop: {
437   \seq_if_empty:NF\g__stex_files_stack{
438     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
439   }
440   \seq_if_empty:NTF\g__stex_files_stack{
441     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
442   }{
443     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
444     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
445   }
446 }
447

```

```

448 \AddToHook{file/before}{
449   \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
450 }
451 \AddToHook{file/after}{
452   \stex_filestack_pop:
453 }

```

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

## 26.4 MathHub Repositories

```

454 <@=stex_mathhub>

\mathhub
\c_stex_mathhub_seq
\c_stex_mathhub_str
455 \str_if_empty:NTF\mathhub{
456   \stex_kpsewhich:n{-var-value~MATHHUB}
457   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
458
459   \str_if_empty:NTF\c_stex_mathhub_str{
460     \msg_warning:nn{stex}{warning/nomathhub}
461   }{
462     \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
463     \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
464   }
465 }{
466   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
467   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
468     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
469       \c_stex_pwd_str/\mathhub
470     }
471   }
472   \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
473   \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
474 }

```

(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
475 \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
476   \str_set:Nx \l_tmpa_str { #1 }
477   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
478     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
479     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
480     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
481     \__stex_mathhub_find_manifest:N \l_tmpa_seq
482     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
483       \msg_error:nnxx{stex}{error/norepository}{#1}{
484         \stex_path_to_string:N \c_stex_mathhub_str
485       }
486     } {
487       \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
488     }
489   }
490 }

```

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

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

491 \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:

```

492 \cs_new_protected:Nn \_stex_mathhub_find_manifest:N {
493   \seq_set_eq:NN\l_tmpa_seq #1
494   \bool_set_true:N\l_tmpa_bool
495   \bool_while_do:Nn \l_tmpa_bool {
496     \seq_if_empty:NTF \l_tmpa_seq {
497       \bool_set_false:N\l_tmpa_bool
498     }{
499       \file_if_exist:nTF{
500         \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
501       }{
502         \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
503         \bool_set_false:N\l_tmpa_bool
504       }{
505         \file_if_exist:nTF{
506           \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
507         }{
508           \seq_put_right:Nn\l_tmpa_seq{META-INF}
509           \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
510           \bool_set_false:N\l_tmpa_bool
511         }{
512           \file_if_exist:nTF{
513             \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
514           }{
515             \seq_put_right:Nn\l_tmpa_seq{meta-inf}
516             \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
517             \bool_set_false:N\l_tmpa_bool
518           }{
519             \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
520           }
521         }
522       }
523     }
524   }
525   \seq_set_eq:NN\l_stex_mathhub_manifest_file_seq\l_tmpa_seq
526 }
```

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

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

527 \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:

```

528 \cs_new_protected:Nn \_stex_mathhub_parse_manifest:n {
529   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
530   \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
531   \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
532     \str_set:Nn \l_tmpa_str {##1}
533     \exp_args:NNo \seq_set_split:Nnn
534       \l_tmpb_seq \c_colon_str \l_tmpa_str
535     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
536       \exp_args:NNe \str_set:Nn \l_tmpb_tl {
537         \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
538       }
539       \exp_args:No \str_case:nnTF \l_tmpa_tl {
540         {id} {
541           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
542             { id } \l_tmpb_tl
543         }
544         {narration-base} {
545           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
546             { narr } \l_tmpb_tl
547         }
548         {url-base} {
549           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
550             { docurl } \l_tmpb_tl
551         }
552         {source-base} {
553           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
554             { ns } \l_tmpb_tl
555         }
556         {ns} {
557           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
558             { ns } \l_tmpb_tl
559         }
560         {dependencies} {
561           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
562             { deps } \l_tmpb_tl
563         }
564       }{}{}
565     }{}
566   }
567   \ior_close:N \c__stex_mathhub_manifest_ior
568 }

```

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

`\stex_set_current_repository:n`

```

569 \cs_new_protected:Nn \stex_set_current_repository:n {
570   \stex_require_repository:n { #1 }
571   \prop_set_eq:Nc \l_stex_current_repository_prop {
572     c_stex_mathhub_#1_manifest_prop
573   }
574 }

```

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

`\stex_require_repository:n`

```

575 \cs_new_protected:Nn \stex_require_repository:n {
576   \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
577     \stex_debug:nn{mathhub}{Opening~archive:~#1}
578     \__stex_mathhub_do_manifest:n { #1 }
579     \exp_args:Nx \stex_add_to_sms:n {
580       \prop_const_from_keyval:cn { c_stex_mathhub_#1_manifest_prop } {
581         id   = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { id } ,
582         ns   = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { ns } ,
583         narr = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { narr } ,
584         deps = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { deps }
585       }
586     }
587   }
588 }

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

589 %\prop_new:N \l_stex_current_repository_prop
590
591 \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
592 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
593   \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
594 } {
595   \__stex_mathhub_parse_manifest:n { main }
596   \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
597   \l_tmpa_str
598   \prop_set_eq:cN { c_stex_mathhub_#1_manifest_prop }
599   \c_stex_mathhub_main_manifest_prop
600   \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
601   \stex_debug:nn{mathhub}{Current~repository:~
602     \prop_item:Nn \l_stex_current_repository_prop {id}
603   }
604 }

```

(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.

```

605 \cs_new_protected:Nn \stex_in_repository:nn {
606   \str_set:Nx \l_tmpa_str { #1 }
607   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
608   \str_if_empty:NTF \l_tmpa_str {
609     \prop_if_exist:NTF \l_stex_current_repository_prop {
610       \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
611       \exp_args:Ne \l_tmpa_cs{
612         \prop_item:Nn \l_stex_current_repository_prop { id }
613       }
614     }{
615       \l_tmpa_cs{}
616     }
617   }{
618     \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}

```

```

619 \stex_require_repository:n \l_tmpa_str
620 \str_set:Nx \l_tmpa_str { #1 }
621 \exp_args:Nne \use:nn {
622   \stex_set_current_repository:n \l_tmpa_str
623   \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
624 }{
625   \stex_debug:nn{mathhub}{switching~back~to:~
626     \prop_if_exist:NTF \l_stex_current_repository_prop {
627       \prop_item:Nn \l_stex_current_repository_prop { id }::~
628       \meaning\l_stex_current_repository_prop
629     }{
630       no~repository
631     }
632   }
633   \prop_if_exist:NTF \l_stex_current_repository_prop {
634     \stex_set_current_repository:n {
635       \prop_item:Nn \l_stex_current_repository_prop { id }
636     }
637   }{
638     \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
639   }
640 }
641 }
642 }

```

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

```

\inputref
\stex_inputref:nn 643 \newif \ifinputref \inputreffalse
\mhinput\stex_mhinput:nn 644
645 \cs_new_protected:Nn \stex_mhinput:nn {
646   \stex_in_repository:nn {#1} {
647     \ifinputref
648       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
649     \else
650       \inputreftrue
651       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
652     \inputreffalse
653   \fi
654 }
655 }
656 \NewDocumentCommand \mhinput { 0{} m}{
657   \stex_mhinput:nn{ #1 }{ #2 }
658 }
659
660 \cs_new_protected:Nn \stex_inputref:nn {
661   \stex_in_repository:nn {#1} {
662     % \bool_lazy_any:nTF {
663     %   {\rustex_if_p:} {\latexml_if_p:}
664     % } {
665     %   \str_clear:N \l_tmpa_str
666     %   \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
667     %     \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
668     %   }

```

```

669 % \stex_annotate_invisible:nnn{inputref}{
670 % \l_tmpa_str / #2
671 % }{}
672 % }{
673 \begingroup
674 \inputreftrue
675 \input{ \c_stex_mathhub_str / ##1 / source / #2 }
676 \endgroup
677 % }
678 }
679 }
680
681 \NewDocumentCommand \inputref { 0{} m}{
682 \stex_inputref:nn{ #1 }{ #2 }
683 }
684
685 \cs_new_protected:Nn \stex_mhbibresource:nn {
686 \stex_in_repository:nn {#1} {
687 \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
688 }
689 }
690 \newcommand\addmhbibresource[2][]{
691 \stex_mhbibresource:nn{ #1 }{ #2 }
692 }

```

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

### `\mhpath`

```

693 \def \mhpath #1 #2 {
694 \exp_args:Ne \str_if_eq:nnTF{#1}{}{
695 \c_stex_mathhub_str /
696 \prop_item:Nn \l_stex_current_repository_prop { id }
697 / source / #2
698 }{
699 \c_stex_mathhub_str / #1 / source / #2
700 }
701 }

```

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

### `\libinput`

```

702 \cs_new_protected:Npn \libinput #1 {
703 \prop_if_exist:NF \l_stex_current_repository_prop {
704 \msg_error:nnn{stex}{error/notinarchive}\libinput
705 }
706 \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
707 \msg_error:nnn{stex}{error/notinarchive}\libinput
708 }
709 \bool_set_false:N \l_tmpa_bool
710 \tl_clear:N \l_tmpa_tl
711 \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
712 \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
713 \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str
714 \seq_pop_left:NNT \l_tmpb_seq \l_tmpb_str {

```

```

715 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
716 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
717 / meta-inf / lib / #1.tex}{
718 \bool_set_true:N \l_tmpa_bool
719 \tl_put_right:Nx \l_tmpa_tl {
720 \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
721 / meta-inf / lib / #1.tex}
722 }
723 }{}
724 }
725 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
726 / \l_tmpa_str / lib / #1.tex
727 }{
728 \bool_set_true:N \l_tmpa_bool
729 \tl_put_right:Nx \l_tmpa_tl {
730 \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
731 / \l_tmpa_str / lib / #1.tex}
732 }
733 }{}
734 \bool_if:NF \l_tmpa_bool {
735 \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
736 }
737 \l_tmpa_tl
738 }

```

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

`\libusepackage`

```

739 \NewDocumentCommand \libusepackage {0{} m} {
740 \prop_if_exist:NF \l_stex_current_repository_prop {
741 \msg_error:nnn{stex}{error/notinarchive}\libusepackage
742 }
743 \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
744 \msg_error:nnn{stex}{error/notinarchive}\libusepackage
745 }
746 \bool_set_false:N \l_libusepackage_bool
747 \tl_clear:N \l_tmpa_tl
748 \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
749 \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
750 \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str
751 \seq_pop_left:NNT \l_tmpb_seq \l_tmpb_str {
752 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
753 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
754 / meta-inf / lib / #2.sty}{
755 \bool_set_true:N \l_libusepackage_bool
756 \tl_put_right:Nx \l_tmpa_tl {
757 \exp_not:N \usepackage[#1] { \stex_path_to_string:N \l_tmpa_seq
758 / meta-inf / lib / #2}
759 }
760 }{}
761 }
762 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
763 / \l_tmpa_str / lib / #2.sty
764 }{

```



```

765 \bool_if:NT \l_libusepackage_bool {
766   \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
767 }
768 \bool_set_true:N \l_libusepackage_bool
769 \tl_put_right:Nx \l_tmpa_tl {
770   \exp_not:N \usepackage[#1] { \stex_path_to_string:N \l_tmpa_seq
771     / \l_tmpa_str / lib / #2}
772 }
773 }{}
774 \bool_if:NF \l_libusepackage_bool {
775   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
776 }
777 \l_tmpa_tl
778 }

```

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

```

779
780 \AddToHook{begindocument}{
781 \ltx@ifpackageloaded{graphicx}{
782   \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
783   \newcommand\mhgraphics[2][]{\%
784     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
785     \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}}
786   \newcommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
787 }{}
788 \ltx@ifpackageloaded{listings}{
789   \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
790   \newcommand\lstinputmhlstlisting[2][]{\%
791     \def\lst@mhrepos{}\setkeys{lst}{#1}%
792     \lstinputlisting[#1]{\mhp\lst@mhrepos{#2}}}
793   \newcommand\clstinputmhlstlisting[2][]{\begin{center}\lstinputmhlstlisting[#1]{#2}\end{center}}
794 }{}
795 }
796
797
798 \end{package}

```

## Chapter 27

# STEX -References Implementation

```
799 <*package>
800
801 %%%%%%%%%% references.dtx %%%%%%%%%%
802
803 %\RequirePackage{hyperref}
804 %\RequirePackage{cleveref}
805 <@@=stex_refs>
806
807 Warnings and error messages
808
809 \iow_new:N \c__stex_refs_refs_iow
810 \AddToHook{begindocument}{
811   \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
812 }
813 \AddToHook{enddocument}{
814   \iow_close:N \c__stex_refs_refs_iow
815 }
816
817 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
818
819 \NewDocumentCommand \STEXreftitle { m } {
820   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
821 }
822
```

### 27.1 Document URIs and URLs

```
820
821 \str_new:N \l_stex_current_docns_str
822
823 \cs_new_protected:Nn \stex_get_document_uri: {
824   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
825   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
826   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
827   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
828   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
829 }
```

```

829
830 \str_clear:N \l_tmpa_str
831 \prop_if_exist:NT \l_stex_current_repository_prop {
832   \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
833     \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
834   }
835 }
836
837 \str_if_empty:NTF \l_tmpa_str {
838   \str_set:Nx \l_stex_current_docns_str {
839     file:/\stex_path_to_string:N \l_tmpa_seq
840   }
841 }{
842   \bool_set_true:N \l_tmpa_bool
843   \bool_while_do:Nn \l_tmpa_bool {
844     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
845     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
846       {source} { \bool_set_false:N \l_tmpa_bool }
847     }{}{
848       \seq_if_empty:NT \l_tmpa_seq {
849         \bool_set_false:N \l_tmpa_bool
850       }
851     }
852   }
853
854   \seq_if_empty:NTF \l_tmpa_seq {
855     \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
856   }{
857     \str_set:Nx \l_stex_current_docns_str {
858       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
859     }
860   }
861 }
862 }
863
864 \str_new:N \l_stex_current_docurl_str
865 \cs_new_protected:Nn \stex_get_document_url: {
866   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
867   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
868   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
869   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
870   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
871
872   \str_clear:N \l_tmpa_str
873   \prop_if_exist:NT \l_stex_current_repository_prop {
874     \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
875       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
876         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
877       }
878     }
879
880   \str_if_empty:NTF \l_tmpa_str {
881     \str_set:Nx \l_stex_current_docurl_str {
882       file:/\stex_path_to_string:N \l_tmpa_seq

```

```

883     }
884   }{
885     \bool_set_true:N \l_tmpa_bool
886     \bool_while_do:Nn \l_tmpa_bool {
887       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
888       \exp_args:No \str_case:nnTF { \l_tmpb_str } {
889         {source} { \bool_set_false:N \l_tmpa_bool }
890       }{}{
891         \seq_if_empty:NT \l_tmpa_seq {
892           \bool_set_false:N \l_tmpa_bool
893         }
894       }
895     }
896
897     \seq_if_empty:NTF \l_tmpa_seq {
898       \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
899     }{
900       \str_set:Nx \l_stex_current_docurl_str {
901         \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
902       }
903     }
904   }
905 }

```

## 27.2 Setting Reference Targets

```

906 \str_const:Nn \c__stex_refs_url_str{URL}
907 \str_const:Nn \c__stex_refs_ref_str{REF}
908 \str_new:N \l__stex_refs_curr_label_str
909 % @currentlabel -> number
910 % @currentlabelname -> title
911 % @currentHref -> name.number <- id of some kind
912 % \theH# -> \arabic{section}
913 % \the# -> number
914 % \hyper@makecurrent{#}
915 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
916   \str_clear:N \l__stex_refs_curr_label_str
917   \str_set:Nx \l_tmpa_str { #1 }
918   \str_if_empty:NF \l_tmpa_str {
919     \stex_get_document_uri:
920     \str_set:Nx \l__stex_refs_curr_label_str {
921       \l_stex_current_docns_str?#1
922     }
923     \seq_if_exist:cF{g__stex_refs_labels_#1_seq}{
924       \seq_new:c {g__stex_refs_labels_#1_seq}
925     }
926     \seq_if_in:coF{g__stex_refs_labels_#1_seq}\l__stex_refs_curr_label_str {
927       \seq_gput_right:co{g__stex_refs_labels_#1_seq}\l__stex_refs_curr_label_str
928     }
929     \stex_if_smsmode:TF {
930       \stex_get_document_url:
931       \str_gset_eq:cN {sref_url_}\l__stex_refs_curr_label_str _str}\l_stex_current_docurl_str
932       \str_gset_eq:cN {sref_}\l__stex_refs_curr_label_str _type}\c__stex_refs_url_str
933     }{

```

```

934     \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~~~\expandafter\unexpanded\expandafter
935     \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
936     \immediate\write\@auxout{\stexauxadddocref{\l_stex_current_docns_str}{#1}}
937     \str_gset:cx {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_ref_str
938   }
939 }
940 }
941
942 \cs_new_protected:Npn \stexauxadddocref #1 #2 {
943   \str_set:Nn \l_tmpa_str {#1?#2}
944   \str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
945   \seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
946     \seq_new:c {g__stex_refs_labels_#2_seq}
947   }
948   \seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
949     \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
950   }
951 }
952
953 \AtEndDocument{
954   \def\stexauxadddocref#1 #2 {}{}}
955 }
956
957 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
958   \stex_if_smsmode:TF {
959     \str_if_exist:cF{sref_sym_#1_type}{
960       \stex_get_document_url:
961       \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str
962       \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
963     }
964   }{
965     \str_if_empty:NF \l__stex_refs_curr_label_str {
966       \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
967       \immediate\write\@auxout{
968         \exp_not:N\expandafter\def\exp_not:N\csname sref_sym_#1_label_str\exp_not:N\endcsname
969         \l__stex_refs_curr_label_str
970       }
971     }
972   }
973 }
974 }

```

## 27.3 Using References

```

975 \str_new:N \l__stex_refs_indocument_str
976 \keys_define:nn { stex / sref } {
977   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
978   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
979   pre           .tl_set:N = \l__stex_refs_pre_tl ,
980   post          .tl_set:N = \l__stex_refs_post_tl ,
981   %indoc        .str_set_x:N = \l__stex_refs_repo_str ,
982 }
983
984

```

```

985
986 \cs_new_protected:Nn \__stex_refs_args:n {
987   \tl_clear:N \l__stex_refs_linktext_tl
988   \tl_clear:N \l__stex_refs_fallback_tl
989   \tl_clear:N \l__stex_refs_pre_tl
990   \tl_clear:N \l__stex_refs_post_tl
991   \str_clear:N \l__stex_refs_repo_str
992   \keys_set:nn { stex / sref } { #1 }
993 }
994
995 \NewDocumentCommand \sref { 0{} m}{
996   \__stex_refs_args:n { #1 }
997   \str_if_empty:NTF \l__stex_refs_indocument_str {
998     \str_set:Nx \l_tmpa_str { #2 }
999     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
1000     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
1001       \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
1002         \seq_get_left:cnF {g__stex_refs_labels_\l_tmpa_str _seq} \l_tmpa_str {
1003           \str_clear:N \l_tmpa_str
1004         }
1005       }{
1006         \str_clear:N \l_tmpa_str
1007       }
1008     }{
1009       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1010       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1011       \int_set:Nn \l_tmpa_int { \exp_args:Nx \str_count:n {\l_tmpb_str?\l_tmpa_str} }
1012       \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
1013         \str_set_eq:NN \l_tmpc_str \l_tmpa_str
1014         \str_clear:N \l_tmpa_str
1015         \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str _seq} {
1016           \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
1017             \str_range:nnn { ##1 }{ -\l_tmpa_int}{ -1 }
1018           }{
1019             \seq_map_break:n {
1020               \str_set:Nn \l_tmpa_str { ##1 }
1021             }
1022           }
1023         }
1024       }{
1025         \str_clear:N \l_tmpa_str
1026       }
1027     }
1028     \str_if_empty:NTF \l_tmpa_str {
1029       \l__stex_refs_fallback_tl
1030     }{
1031       \str_if_eq:cNTF {sref_\l_tmpa_str _type} \c__stex_refs_ref_str {
1032         \cs_if_exist:cTF{autoref}{
1033           \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1034         }{
1035           \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1036         }
1037       }{
1038         \ltx@ifpackageloaded{hyperref}{

```

```

1039         \exp_args:Nx \href{\use:c{sref_url_\l_tmpa_str _str}}{\l__stex_refs_fallback_tl}
1040     }{
1041         \l__stex_refs_fallback_tl
1042     }
1043 }
1044 }
1045 }{
1046     % TODO
1047 }
1048 }
1049
1050 \NewDocumentCommand \srefsym { 0{} m}{
1051     \stex_get_symbol:n { #2 }
1052     \str_if_exist:cTF {sref_sym_\l_stex_get_symbol_uri_str _label_str }{
1053         \sref[#1]{\use:c{sref_sym_\l_stex_get_symbol_uri_str _label_str}}
1054     }{
1055         \__stex_refs_args:n { #1 }
1056         \str_if_empty:NTF \l__stex_refs_indocument_str {
1057             \tl_set:Nn \l_tmpa_tl {
1058                 \l__stex_refs_fallback_tl
1059             }
1060             \tl_if_exist:cT{sref_sym_\l_stex_get_symbol_uri_str _type}{
1061                 \tl_set:Nn \l_tmpa_tl {
1062                     % doc uri in \l_tmpb_str
1063                     \str_set:Nx \l_tmpa_str {\use:c{sref_sym_\l_stex_get_symbol_uri_str _type}}
1064                     \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1065                         % reference
1066                         \cs_if_exist:cTF{autoref}{
1067                             \l__stex_refs_pre_tl\autoref{sref_sym_\l_stex_get_symbol_uri_str}\l__stex_refs_pos
1068                         }{
1069                             \l__stex_refs_pre_tl\ref{sref_sym_\l_stex_get_symbol_uri_str}\l__stex_refs_pos
1070                         }
1071                     }{
1072                         % URL
1073                         \ltx@ifpackageloaded{hyperref}{
1074                             \exp_args:Nx \href{\use:c{sref_sym_url_\l_stex_get_symbol_uri_str _str}}{\l__s
1075                         }{
1076                             \l__stex_refs_fallback_tl
1077                         }
1078                     }
1079                 }
1080             }
1081             \l_tmpa_tl
1082         }{
1083             % TODO
1084         }
1085     }
1086 }
1087
1088 \cs_new_protected:Npn \srefsymuri #1 #2 {
1089     \str_if_exist:cTF {sref_sym_#1 _label_str }{
1090         \exp_args:Nx \hyperref{\use:c{sref_sym_\l_stex_get_symbol_uri_str _label_str}}{#2}
1091     }{
1092         \hyperref[sref_sym_#1]{#2}

```

```
1093     }  
1094   }  
1095  
1096 </package>
```



## Chapter 28

# STEX -Modules Implementation

```
1097 <*package>
1098
1099 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1100
1101 <@@=stex_modules>
1102
1103     Warnings and error messages
1104 \msg_new:nnn{stex}{error/unknownmodule}{
1105     No~module~#1~found
1106 }
1107 \msg_new:nnn{stex}{error/syntax}{
1108     Syntax~error:~#1
1109 }
1110 \msg_new:nnn{stex}{error/siglanguage}{
1111     Module~#1~declares~signature~#2,~but~does~not~
1112     declare~its~language
1113 }
1114 \msg_new:nnn{stex}{warning/deprecated}{
1115     #1~is~deprecated;~please~use~#2~instead!
1116 }
1117 \msg_new:nnn{stex}{error/conflictingmodules}{
1118     Conflicting~imports~for~module~#1
1119 }
1120
1121 \l_stex_current_module_str The current module:
1122 \str_new:N \l_stex_current_module_str
1123
1124 (End definition for \l_stex_current_module_str. This variable is documented on page 26.)
1125
1126 \l_stex_all_modules_seq Stores all available modules
1127 \seq_new:N \l_stex_all_modules_seq
1128
1129 (End definition for \l_stex_all_modules_seq. This variable is documented on page 26.)
```

```

\stex_if_in_module_p:
\stex_if_in_module:TF
1121 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1122   \str_if_empty:NTF \l_stex_current_module_str
1123   \prg_return_false: \prg_return_true:
1124 }

(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
1125 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1126   \prop_if_exist:cTF { c_stex_module_#1_prop }
1127   \prg_return_true: \prg_return_false:
1128 }

(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:
1129 \cs_new_protected:Nn \stex_add_to_current_module:n {
1130   \tl_gput_right:cn {c_stex_module_#1_stex_current_module_str _code} { #1 }
1131 }
1132 \cs_new_protected:Npn \STEXexport {
1133   \begingroup
1134   \newlinechar=-1\relax
1135   \endlinechar=-1\relax
1136   %\catcode'\ = 9\relax
1137   \expandafter\endgroup\STEXexport:n
1138 }
1139 \cs_new_protected:Nn \STEXexport:n {
1140   \ignorespaces #1
1141   \stex_add_to_current_module:n { \ignorespaces #1 }
1142   \stex_smsmode_do:
1143 }
1144 \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
1145 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1146   \str_set:Nx \l_tmpa_str { #1 }
1147   \seq_gput_right:co {c_stex_module_#1_stex_current_module_str _constants} { \l_tmpa_str }
1148 }
1149
1150 %\cs_new_protected:Nn \stex_add_field_to_current_module:n {
1151 % \str_set:Nx \l_tmpa_str { #1 }
1152 % \seq_gput_right:co {c_stex_module_#1_stex_current_module_str _fields} { \l_tmpa_str }
1153 %}

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

```

```

\stex_collect_imports:n
1154 \cs_new_protected:Nn \stex_collect_imports:n {
1155   \seq_clear:N \l_stex_collect_imports_seq
1156   \__stex_modules_collect_imports:n {#1}

```

```

1157 }
1158 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1159   \seq_map_inline:cn {c_stex_module_#1_imports} {
1160     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1161       \__stex_modules_collect_imports:n { ##1 }
1162     }
1163   }
1164   \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1165     \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1166   }
1167 }

```

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

`\stex_add_import_to_current_module:n`

```

1168 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1169   \str_set:Nx \l_tmpa_str { #1 }
1170   \exp_args:Nno
1171   \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str_imports}\l_tmpa_str{
1172     \seq_gput_right:co{c_stex_module_\l_stex_current_module_str_imports}\l_tmpa_str
1173   }
1174 }

```

(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).

```

1175 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
1176   \str_set:Nx \l_tmpa_str { #1 }
1177   \seq_set_eq:NN \l_tmpa_seq #2
1178   % split off file extension
1179   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1180   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1181   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1182   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1183
1184   \bool_set_true:N \l_tmpa_bool
1185   \bool_while_do:Nn \l_tmpa_bool {
1186     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1187     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1188       {source} { \bool_set_false:N \l_tmpa_bool }
1189     }{}{
1190       \seq_if_empty:NT \l_tmpa_seq {
1191         \bool_set_false:N \l_tmpa_bool
1192       }
1193     }
1194   }
1195
1196   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
1197   \str_if_empty:NTF \l_stex_modules_subpath_str {
1198     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
1199   }{
1200     \str_set:Nx \l_stex_modules_ns_str {
1201       \l_tmpa_str/\l_stex_modules_subpath_str

```

```

1202     }
1203   }
1204 }

```

(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
1205 \str_new:N \l_stex_modules_ns_str
1206 \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.

```

1207 \cs_new_protected:Nn \stex_modules_current_namespace: {
1208   \str_clear:N \l_stex_modules_subpath_str
1209   \prop_if_exist:NTF \l_stex_current_repository_prop {
1210     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1211     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1212   }{
1213     % split off file extension
1214     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1215     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1216     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1217     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1218     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1219     \str_set:Nx \l_stex_modules_ns_str {
1220       file:/\stex_path_to_string:N \l_tmpa_seq
1221     }
1222   }
1223 }

```

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

## 28.1 The module environment

module arguments:

```

1224 \keys_define:nn { stex / module } {
1225   title      .tl_set:N      = \smodulename ,
1226   type       .str_set_x:N   = \smodulename ,
1227   id         .str_set_x:N   = \smodulename ,
1228   deprecate  .str_set_x:N   = \l_stex_module_deprecate_str ,
1229   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1230   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1231   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1232   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1233   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1234   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1235   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1236 }
1237
1238 \cs_new_protected:Nn \__stex_modules_args:n {

```

```

1239 \str_clear:N \smoduletitle
1240 \str_clear:N \smoduletype
1241 \str_clear:N \smoduleid
1242 \str_clear:N \l_stex_module_ns_str
1243 \str_clear:N \l_stex_module_deprecate_str
1244 \str_clear:N \l_stex_module_lang_str
1245 \str_clear:N \l_stex_module_sig_str
1246 \str_clear:N \l_stex_module_creators_str
1247 \str_clear:N \l_stex_module_contributors_str
1248 \str_clear:N \l_stex_module_meta_str
1249 \str_clear:N \l_stex_module_srccite_str
1250 \keys_set:nn { stex / module } { #1 }
1251 }
1252
1253 % module parameters here? In the body?
1254

```

**\stex\_module\_setup:nn** Sets up a new module property list:

```

1255 \cs_new_protected:Nn \stex_module_setup:nn {
1256   \str_set:Nx \l_stex_module_name_str { #2 }
1257   \__stex_modules_args:n { #1 }

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

1258   \stex_if_in_module:TF {
1259     % Nested module
1260     \prop_get:cnN {c_stex_module\l_stex_current_module_str _prop}
1261       { ns } \l_stex_module_ns_str
1262     \str_set:Nx \l_stex_module_name_str {
1263       \prop_item:cn {c_stex_module\l_stex_current_module_str _prop}
1264         { name } / \l_stex_module_name_str
1265     }
1266   }{
1267     % not nested:
1268     \str_if_empty:NT \l_stex_module_ns_str {
1269       \stex_modules_current_namespace:
1270       \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1271       \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1272         / {\l_stex_module_ns_str}
1273       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1274       \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1275         \str_set:Nx \l_stex_module_ns_str {
1276           \stex_path_to_string:N \l_tmpa_seq
1277         }
1278       }
1279     }
1280   }

```

Next, we determine the language of the module:

```

1281 \str_if_empty:NT \l_stex_module_lang_str {
1282   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1283   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1284   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1285   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>

```

```

1286 \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1287 \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1288   inferred~from~file~name}
1289 \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1290 }
1291 }
1292
1293 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
1294 \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1295 \l_tmpa_str {
1296 \ltx@ifpackageloaded{babel}{
1297 \exp_args:Nx \selectlanguage { \l_tmpa_str }
1298 }{}
1299 } {
1300 \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
1301 }
1302 }}

```

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

```

1303 \str_if_empty:NNTF \l_stex_module_sig_str {
1304 \exp_args:Nnx \prop_gset_from_keyval:cn {
1305   c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1306 } {
1307   name      = \l_stex_module_name_str ,
1308   ns        = \l_stex_module_ns_str ,
1309   file      = \exp_not:o { \g_stex_currentfile_seq } ,
1310   lang      = \l_stex_module_lang_str ,
1311   sig       = \l_stex_module_sig_str ,
1312   deprecate = \l_stex_module_deprecate_str ,
1313   meta      = \l_stex_module_meta_str
1314 }
1315 \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1316 \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _fields}
1317 \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1318 \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1319 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1320 \str_if_empty:NT \l_stex_module_meta_str {
1321 \str_set:Nx \l_stex_module_meta_str {
1322 \c_stex_metatheory_ns_str ? Metatheory
1323 }
1324 }
1325 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1326 \bool_set_true:N \l_stex_in_meta_bool
1327 \exp_args:Nx \stex_add_to_current_module:n {
1328 \bool_set_true:N \l_stex_in_meta_bool
1329 \stex_activate_module:n {\l_stex_module_meta_str}
1330 \bool_set_false:N \l_stex_in_meta_bool
1331 }
1332 \stex_activate_module:n {\l_stex_module_meta_str}
1333 \bool_set_false:N \l_stex_in_meta_bool
1334 }

```

```

1335 }{
1336   \str_if_empty:NT \l_stex_module_lang_str {
1337     \msg_error:nnxx{stex}{error/siglanguage}{
1338       \l_stex_module_ns_str?\l_stex_module_name_str
1339     }\l_stex_module_sig_str}
1340   }
1341
1342   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1343   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1344   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1345   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1346   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1347   \str_set:Nx \l_tmpa_str {
1348     \stex_path_to_string:N \l_tmpa_seq /
1349     \l_tmpa_str . \l_stex_module_sig_str .tex
1350   }
1351   \IfFileExists \l_tmpa_str {
1352     \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1353       \str_clear:N \l_stex_current_module_str
1354       \seq_clear:N \l_stex_all_modules_seq
1355       \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1356     }
1357   }{
1358     \msg_error:nnxx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1359   }
1360   \stex_if_smsmode:F {
1361     \stex_activate_module:n {
1362       \l_stex_module_ns_str ? \l_stex_module_name_str
1363     }
1364   }
1365   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1366 }
1367 \str_if_empty:NF \l_stex_module_deprecate_str {
1368   \msg_warning:nnxx{stex}{warning/deprecated}{
1369     Module~\l_stex_current_module_str
1370   }{
1371     \l_stex_module_deprecate_str
1372   }
1373 }
1374 }

```

(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}
1375 \int_new:N \l_stex_module_group_depth_int
1376 \cs_new_protected:Nn \__stex_modules_begin_module: {
1377   \stex_reactivate_macro:N \STEXexport
1378   \stex_reactivate_macro:N \importmodule
1379   \stex_reactivate_macro:N \symdecl
1380   \stex_reactivate_macro:N \notation
1381   \stex_reactivate_macro:N \symdef
1382

```

```

1383 \stex_debug:nn{modules}{
1384   New~module:\\
1385   Namespace::~\l_stex_module_ns_str\\
1386   Name::~\l_stex_module_name_str\\
1387   Language::~\l_stex_module_lang_str\\
1388   Signature::~\l_stex_module_sig_str\\
1389   Metatheory::~\l_stex_module_meta_str\\
1390   File::~\stex_path_to_string:N \g_stex_currentfile_seq
1391 }
1392
1393 \seq_put_right:Nx \l_stex_all_modules_seq {
1394   \l_stex_module_ns_str ? \l_stex_module_name_str
1395 }
1396
1397 % \seq_gput_right:Nx \g_stex_modules_in_file_seq
1398 %   { \l_stex_module_ns_str ? \l_stex_module_name_str }
1399
1400
1401 \stex_if_smsmode:F{
1402   \begin{stex_annotate_env} {theory} {
1403     \l_stex_module_ns_str ? \l_stex_module_name_str
1404   }
1405
1406   \stex_annotate_invisible:nnn{header}{} {
1407     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1408     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1409     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1410       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1411     }
1412     \str_if_empty:NF \smodulotype {
1413       \stex_annotate:nnn{type}{\smodulotype}{}
1414     }
1415   }
1416 }
1417 \int_set:Nn \l_stex_module_group_depth_int {\currentgrouplevel}
1418 % TODO: Inherit metatheory for nested modules?
1419 }
1420 \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}

```

1421 \cs_new_protected:Nn \_stex_modules_end_module: {
1422 % \str_set:Nx \l_tmpa_str {
1423 %   c_stex_module_
1424 %   \prop_item:Nn \l_stex_current_module_prop { ns } ?
1425 %   \prop_item:Nn \l_stex_current_module_prop { name }
1426 %   _prop
1427 % }
1428 %%^A \prop_new:c { \l_tmpa_str }
1429 % \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
1430 \stex_debug:nn{modules}{Closing module~\prop_item:cn {c_stex_module_\l_stex_current_module_
1431 }

```

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



smodule The core environment, with no header

```

1432 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1433 \NewDocumentEnvironment { smodule } { 0{} m } {
1434   \stex_module_setup:nn{#1}{#2}
1435   \par
1436   \stex_if_smsmode:F{
1437     \tl_clear:N \l_tmpa_tl
1438     \clist_map_inline:Nn \smoduletype {
1439       \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1440         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1441       }
1442     }
1443     \tl_if_empty:NTF \l_tmpa_tl {
1444       \__stex_modules_smodule_start:
1445     }{
1446       \l_tmpa_tl
1447     }
1448   }
1449   \__stex_modules_begin_module:
1450   \stex_ref_new_doc_target:n \smoduleid
1451   \stex_smsmode_do:
1452 } {
1453   \__stex_modules_end_module:
1454   \stex_if_smsmode:TF {
1455     % \exp_args:Nx \stex_add_to_sms:n {
1456     %   \prop_gset_from_keyval:cn {
1457     %     c_stex_module_
1458     %     \prop_item:Nn \l_stex_current_module_prop { ns } ?
1459     %     \prop_item:Nn \l_stex_current_module_prop { name }
1460     %     _prop
1461     %   } {
1462     %     name      = \prop_item:cn { \l_tmpa_str } { name } ,
1463     %     ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
1464     %     file      = \prop_item:cn { \l_tmpa_str } { file } ,
1465     %     lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
1466     %     sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
1467     %     meta      = \prop_item:cn { \l_tmpa_str } { meta }
1468     %   }
1469     % }
1470   }{
1471     \end{stex_annotate_env}
1472     \clist_set:No \l_tmpa_clist \smoduletype
1473     \tl_clear:N \l_tmpa_tl
1474     \clist_map_inline:Nn \l_tmpa_clist {
1475       \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1476         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1477       }
1478     }
1479     \tl_if_empty:NTF \l_tmpa_tl {
1480       \__stex_modules_smodule_end:
1481     }{
1482       \l_tmpa_tl
1483     }
1484   }

```

```

1485 }
1486
1487 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1488 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1489
1490 \newcommand\stexpatchmodule[3] [] {
1491   \str_set:Nx \l_tmpa_str{ #1 }
1492   \str_if_empty:NTF \l_tmpa_str {
1493     \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1494     \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1495   }{
1496     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1497     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1498   }
1499 }
1500

```

## 28.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n
1501 \NewDocumentCommand \STEXModule { m } {
1502   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1503   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1504   \tl_set:Nn \l_tmpa_tl {
1505     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1506   }
1507   \seq_map_inline:Nn \l_stex_all_modules_seq {
1508     \str_set:Nn \l_tmpb_str { ##1 }
1509     \str_if_eq:eeT { \l_tmpa_str } {
1510       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1511     } {
1512       \seq_map_break:n {
1513         \tl_set:Nn \l_tmpa_tl {
1514           \stex_invoke_module:n { ##1 }
1515         }
1516       }
1517     }
1518   }
1519   \l_tmpa_tl
1520 }
1521
1522 \cs_new_protected:Nn \stex_invoke_module:n {
1523   \stex_debug:nn{modules}{Invoking~module~#1}
1524   \peek_charcode_remove:NTF ! {
1525     \__stex_modules_invoke_uri:nN { #1 }
1526   } {
1527     \peek_charcode_remove:NTF ? {
1528       \__stex_modules_invoke_symbol:nn { #1 }
1529     } {
1530       \msg_error:nnx{stex}{error/syntax}{
1531         ?~or~!~expected~after~
1532         \c_backslash_str STEXModule{#1}
1533       }

```

```

1534     }
1535   }
1536 }
1537
1538 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1539   \str_set:Nn #2 { #1 }
1540 }
1541
1542 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1543   \stex_invoke_symbol:n{#1?#2}
1544 }

```

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

`\stex_activate_module:n`

```

1545 \bool_new:N \l_stex_in_meta_bool
1546 \bool_set_false:N \l_stex_in_meta_bool
1547 \cs_new_protected:Nn \stex_activate_module:n {
1548   \stex_debug:nn{modules}{Activating~module~#1}
1549   \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1550     \msg_error:nnn{stex}{error/conflictingmodules}{ #1 }
1551   }
1552   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1553     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1554     \use:c{ c_stex_module_#1_code }
1555   }
1556 }

```

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

```

1557 </package>

```

## Chapter 29

# STEX -Module Inheritance Implementation

```
1558 <*package>
1559
1560 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1561
```

### 29.1 SMS Mode

```
1562 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1563 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1564 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1565 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1566
1567 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1568   \makeatletter
1569   \makeatother
1570   \ExplSyntaxOn
1571   \ExplSyntaxOff
1572   \rustexBREAK
1573 }
1574
1575 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1576   \symdef
1577   \importmodule
1578   \notation
1579   \symdecl
1580   \STEXexport
1581   \inlineass
1582   \inlinedef
1583   \inlineex
1584   \endinput
1585   \setnotation
```

```

1586 \copynotation
1587 }
1588
1589 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1590   \tl_to_str:n {
1591     smodule,
1592     copymodule,
1593     interpretmodule
1594     sdefinition,
1595     sexample,
1596     sassertion,
1597     sparagraph
1598   }
1599 }

```

(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`

```

1600 \bool_new:N \g__stex_smsmode_bool
1601 \bool_set_false:N \g__stex_smsmode_bool
1602 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1603   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1604 }

```

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

`\stex_in_smsmode:nn`

```

1605 \cs_new_protected:Nn \stex_in_smsmode:nn {
1606   \vbox_set:Nn \l_tmpa_box {
1607     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1608     \bool_gset_true:N \g__stex_smsmode_bool
1609     #2
1610     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1611   }
1612   \box_clear:N \l_tmpa_box
1613 }
1614
1615 \quark_new:N \q__stex_smsmode_break
1616
1617 %\ior_new:N \c__stex_smsmode_ior
1618 %\tl_new:N \l__stex_smsmode_filecontent_tl
1619 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1620   % \tl_clear:N \l__stex_smsmode_filecontent_tl
1621   % \ior_open:Nn \c__stex_smsmode_ior {#1}
1622   % \ior_map_inline:Nn \c__stex_smsmode_ior {
1623     %   \tl_put_right:Nn \l__stex_smsmode_filecontent_tl { ##1 }
1624   % }
1625   % \ior_close:N \c__stex_smsmode_ior
1626   \stex_filestack_push:n{#1}
1627   \stex_in_smsmode:nn{#1} {
1628     #2
1629     \everyeof{\q__stex_smsmode_break\noexpand}
1630     \expandafter\expandafter\expandafter
1631     \stex_smsmode_do:

```

```

1632     \csname @ @ input\endcsname "#1"\relax
1633     %\expandafter \stex_smsmode_do: \l__stex_smsmode_filecontent_tl
1634   }
1635   \stex_filestack_pop:
1636 }

```

(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:

```

1637 \cs_new_protected:Npn \stex_smsmode_do: {
1638   \stex_if_smsmode:T {
1639     \__stex_smsmode_do:w
1640   }
1641 }
1642 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1643   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
1644     \expandafter\if\expandafter\relax\noexpand#1
1645     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1646   \else\expandafter\__stex_smsmode_do:w\fi
1647 }{
1648   \__stex_smsmode_do:w % #1
1649 }
1650 }
1651 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1652   \cs_if_eq:NNTF #1 \q__stex_smsmode_break {
1653     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1654       #1\__stex_smsmode_do:w
1655     }{
1656       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1657         #1
1658       }{
1659         \cs_if_eq:NNTF \begin #1 {
1660           \__stex_smsmode_check_begin:n
1661         }{
1662           \cs_if_eq:NNTF \end #1 {
1663             \__stex_smsmode_check_end:n
1664           }{
1665             \__stex_smsmode_do:w
1666           }
1667         }
1668       }
1669     }
1670   }
1671 }
1672
1673 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1674   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1675     \begin{#1}
1676   }{
1677     \__stex_smsmode_do:w
1678   }
1679 }
1680 \cs_new_protected:Nn \__stex_smsmode_check_end:n {

```

```

1681 \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1682   \end{#1}\__stex_smsmode_do:w
1683 }{
1684   \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1685 }
1686 }

```

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

## 29.2 Inheritance

```

1687 <@@=stex_importmodule>

```

`\stex_import_module_uri:nn`

```

1688 \cs_new_protected:Nn \stex_import_module_uri:nn {
1689   \str_set:Nx \l_stex_import_archive_str { #1 }
1690   \str_set:Nn \l_stex_import_path_str { #2 }
1691
1692   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1693   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1694   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1695
1696   \stex_modules_current_namespace:
1697   \bool_lazy_all:nTF {
1698     {\str_if_empty_p:N \l_stex_import_archive_str}
1699     {\str_if_empty_p:N \l_stex_import_path_str}
1700     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1701   }{
1702     \str_set_eq:NN \l_stex_import_path_str \l_stex_modules_subpath_str
1703     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1704   }{
1705     \str_if_empty:NT \l_stex_import_archive_str {
1706       \prop_if_exist:NT \l_stex_current_repository_prop {
1707         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1708       }
1709     }
1710     \str_if_empty:NTF \l_stex_import_archive_str {
1711       \str_if_empty:NF \l_stex_import_path_str {
1712         \str_set:Nx \l_stex_import_ns_str {
1713           \l_stex_module_ns_str / \l_stex_import_path_str
1714         }
1715       }
1716     }{
1717       \stex_require_repository:n \l_stex_import_archive_str
1718       \prop_get:cnN { c_stex_mathhub_\l_stex_import_archive_str_manifest_prop } { ns }
1719       \l_stex_import_ns_str
1720       \str_if_empty:NF \l_stex_import_path_str {
1721         \str_set:Nx \l_stex_import_ns_str {
1722           \l_stex_import_ns_str / \l_stex_import_path_str
1723         }
1724       }
1725     }
1726   }
1727 }

```

(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 1728 \str_new:N \l_stex_import_name_str
\l_stex_import_path_str 1729 \str_new:N \l_stex_import_archive_str
\l_stex_import_ns_str 1730 \str_new:N \l_stex_import_path_str
1731 \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:nmmn {<ns>} {<archive-ID>} {<path>} {<name>}
1732 \cs_new_protected:Nn \stex_import_require_module:nmmn {
1733   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1734
1735     % archive
1736     \str_set:Nx \l_tmpa_str { #2 }
1737     \str_if_empty:NTF \l_tmpa_str {
1738       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1739     } {
1740       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1741       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1742       \seq_put_right:Nn \l_tmpa_seq { source }
1743     }
1744
1745     % path
1746     \str_set:Nx \l_tmpb_str { #3 }
1747     \str_if_empty:NTF \l_tmpb_str {
1748       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1749
1750       \ltx@ifpackageloaded{babel} {
1751         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1752           { \languagename } \l_tmpb_str {
1753           \msg_error:nnx{stex}{error/unknownlanguage}{\languagename}
1754         }
1755       } {
1756         \str_clear:N \l_tmpb_str
1757       }
1758
1759       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1760       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1761         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1762       }{
1763         \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1764         \IfFileExists{ \l_tmpa_str.tex }{
1765           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1766         }{
1767           % try english as default
1768           \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1769           \IfFileExists{ \l_tmpa_str.en.tex }{
1770             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1771           }{
1772             \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1773           }
1774         }

```



```

1775     }
1776
1777   } {
1778     \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1779     \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1780
1781     \ltx@ifpackageloaded{babel} {
1782       \exp_args:Nnx \prop_get:NnNF \c_stex_language_abbrevs_prop
1783         { \language } \l_tmpb_str {
1784         \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1785       }
1786     } {
1787       \str_clear:N \l_tmpb_str
1788     }
1789
1790     \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1791
1792     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1793     \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1794       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1795     }{
1796       \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1797       \IfFileExists{ \l_tmpa_str/#4.tex }{
1798         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1799       }{
1800         % try english as default
1801         \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1802         \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1803           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1804         }{
1805           \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1806           \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1807             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1808           }{
1809             \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1810             \IfFileExists{ \l_tmpa_str.tex }{
1811               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1812             }{
1813               % try english as default
1814               \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1815               \IfFileExists{ \l_tmpa_str.en.tex }{
1816                 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1817               }{
1818                 \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1819               }
1820             }
1821           }
1822         }
1823       }
1824     }
1825   }
1826
1827   \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
1828     \seq_clear:N \l_stex_all_modules_seq

```

```

1829     \str_clear:N \l_stex_current_module_str
1830     \str_set:Nx \l_tmpb_str { #2 }
1831     \str_if_empty:NF \l_tmpb_str {
1832         \stex_set_current_repository:n { #2 }
1833     }
1834     \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1835 }
1836
1837 \stex_if_module_exists:nF { #1 ? #4 } {
1838     \msg_error:nnx{stex}{error/unknownmodule}{
1839         #1?#4~(in~file~\g__stex_importmodule_file_str)
1840     }
1841 }
1842 }
1843 \stex_activate_module:n { #1 ? #4 }
1844 }

```

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

### `\importmodule`

```

1845 \NewDocumentCommand \importmodule { 0{} m } {
1846     \stex_import_module_uri:nn { #1 } { #2 }
1847     \stex_debug:nn{modules}{Importing~module:~
1848         \l_stex_import_ns_str ? \l_stex_import_name_str
1849     }
1850     \stex_if_smsmode:F {
1851         \stex_import_require_module:nnnn
1852         { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1853         { \l_stex_import_path_str } { \l_stex_import_name_str }
1854         \stex_annotate_invisible:nnn
1855         {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1856     }
1857     \exp_args:Nx \stex_add_to_current_module:n {
1858         \stex_import_require_module:nnnn
1859         { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1860         { \l_stex_import_path_str } { \l_stex_import_name_str }
1861     }
1862     \exp_args:Nx \stex_add_import_to_current_module:n {
1863         \l_stex_import_ns_str ? \l_stex_import_name_str
1864     }
1865     \stex_smsmode_do:
1866     \ignorespacesandpars
1867 }
1868 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

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

### `\usemodule`

```

1869 \NewDocumentCommand \usemodule { 0{} m } {
1870     \stex_if_smsmode:F {
1871         \stex_import_module_uri:nn { #1 } { #2 }
1872         \stex_import_require_module:nnnn
1873         { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1874         { \l_stex_import_path_str } { \l_stex_import_name_str }
1875         \stex_annotate_invisible:nnn

```

```

1876     {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1877   }
1878   \stex_smsmode_do:
1879   \ignorespacesandpars
1880 }

```

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

```

1881 </package>

```

# Chapter 30

## STEX -Symbols Implementation

```
1882 <*package>
1883
1884 %%%%%%%%%% symbols.dtx %%%%%%%%%%
1885
```

Warnings and error messages

```
1886
```

### 30.1 Symbol Declarations

```
1887 <@@=stex_symdecl>
```

`\l_stex_all_symbols_seq` Stores all available symbols

```
1888 \seq_new:N \l_stex_all_symbols_seq
```

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

`\STEXsymbol`

```
1889 \NewDocumentCommand \STEXsymbol { m } {
1890   \stex_get_symbol:n { #1 }
1891   \exp_args:No
1892   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1893 }
```

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

symdecl arguments:

```
1894 \keys_define:nn { stex / symdecl } {
1895   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1896   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1897   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1898   type      .tl_set:N   = \l_stex_symdecl_type_tl ,
1899   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,
1900   align     .str_set:N   = \l_stex_symdecl_align_str , % TODO(?)
1901   gfc       .str_set:N   = \l_stex_symdecl_gfc_str , % TODO(?)
1902   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
1903   def       .tl_set:N   = \l_stex_symdecl_definiens_tl
```

```

1904 }
1905
1906 \bool_new:N \l_stex_symdecl_make_macro_bool
1907
1908 \cs_new_protected:Nn \__stex_symdecl_args:n {
1909   \str_clear:N \l_stex_symdecl_name_str
1910   \str_clear:N \l_stex_symdecl_args_str
1911   \str_clear:N \l_stex_symdecl_deprecate_str
1912   \bool_set_false:N \l_stex_symdecl_local_bool
1913   \tl_clear:N \l_stex_symdecl_type_tl
1914   \tl_clear:N \l_stex_symdecl_definiens_tl
1915
1916   \keys_set:nn { stex / symdecl } { #1 }
1917 }

```

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

```

1918
1919 \NewDocumentCommand \symdecl { s O{} m } {
1920   \__stex_symdecl_args:n { #2 }
1921   \IfBooleanTF #1 {
1922     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1923   } {
1924     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1925   }
1926   \stex_symdecl_do:n { #3 }
1927   \stex_smsmode_do:
1928 }
1929 \stex_deactivate_macro:Nn \symdecl {module~environments}

```

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

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

```

1930 \cs_new_protected:Nn \stex_symdecl_do:n {
1931   \stex_if_in_module:F {
1932     % TODO throw error? some default namespace?
1933   }
1934
1935   \str_if_empty:NT \l_stex_symdecl_name_str {
1936     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1937   }
1938
1939   \prop_if_exist:cT { l_stex_symdecl_
1940     \l_stex_current_module_str ?
1941     \l_stex_symdecl_name_str
1942   }_prop
1943 }{
1944   % TODO throw error (beware of circular dependencies)
1945 }
1946
1947 \prop_clear:N \l_tmpa_prop
1948 \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
1949 \seq_clear:N \l_tmpa_seq
1950 \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str

```

```

1951 \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1952
1953 \str_if_empty:NT \l_stex_symdecl_deprecate_str {
1954   \str_if_empty:NF \l_stex_module_deprecate_str {
1955     \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
1956   }
1957 }
1958 \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
1959
1960 \exp_args:No \stex_add_constant_to_current_module:n {
1961   \l_stex_symdecl_name_str
1962 }
1963
1964 % arity/args
1965 \int_zero:N \l_tmpb_int
1966
1967 \bool_set_true:N \l_tmpa_bool
1968 \str_map_inline:Nn \l_stex_symdecl_args_str {
1969   \token_case_meaning:NnF ##1 {
1970     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1971     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1972     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1973     {\tl_to_str:n a} {
1974       \bool_set_false:N \l_tmpa_bool
1975       \int_incr:N \l_tmpb_int
1976     }
1977     {\tl_to_str:n B} {
1978       \bool_set_false:N \l_tmpa_bool
1979       \int_incr:N \l_tmpb_int
1980     }
1981   }{
1982     \msg_set:nnn{stex}{error/wrongargs}{
1983       args~value~in~symbol~declaration~for~
1984       \l_stex_current_module_str ?
1985       \l_stex_symdecl_name_str ~
1986       needs~to~be~
1987       i,~a,~b~or~B,~but~##1~given
1988     }
1989     \msg_error:nn{stex}{error/wrongargs}
1990   }
1991 }
1992 \bool_if:NTF \l_tmpa_bool {
1993   % possibly numeric
1994   \str_if_empty:NTF \l_stex_symdecl_args_str {
1995     \prop_put:Nnn \l_tmpa_prop { args } {}
1996     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1997   }{
1998     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1999     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2000     \str_clear:N \l_tmpa_str
2001     \int_step_inline:nn \l_tmpa_int {
2002       \str_put_right:Nn \l_tmpa_str i
2003     }
2004     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }

```

```

2005     }
2006   } {
2007     \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2008     \prop_put:Nnx \l_tmpa_prop { arity }
2009     { \str_count:N \l_stex_symdecl_args_str }
2010   }
2011   \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2012
2013
2014   % semantic macro
2015
2016   \bool_if:NT \l_stex_symdecl_make_macro_bool {
2017     \exp_args:Nx \stex_do_aftergroup:n {
2018       \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2019         \l_stex_current_module_str ? \l_stex_symdecl_name_str
2020       }}
2021     }
2022
2023     \bool_if:NF \l_stex_symdecl_local_bool {
2024       \exp_args:Nx \stex_add_to_current_module:n {
2025         \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2026           \l_stex_current_module_str ? \l_stex_symdecl_name_str
2027         } }
2028       }
2029     }
2030   }
2031
2032   % add to all symbols
2033
2034   \bool_if:NF \l_stex_symdecl_local_bool {
2035     \exp_args:Nx \stex_add_to_current_module:n {
2036       \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2037         \l_stex_current_module_str ? \l_stex_symdecl_name_str
2038       }
2039     }
2040   %   \exp_args:Nx \stex_add_field_to_current_module:n {
2041   %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2042   %   }
2043   }
2044
2045   \stex_debug:nn{symbols}{New~symbol:~
2046     \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2047     Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2048     Args:~\prop_item:Nn \l_tmpa_prop { args }
2049   }
2050
2051   % circular dependencies require this:
2052
2053   \prop_if_exist:cF {
2054     l_stex_symdecl_
2055     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2056     _prop
2057   } {
2058     \prop_set_eq:cN {

```

```

2059     l_stex_symdecl_
2060     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2061     _prop
2062   } \l_tmpa_prop
2063 }
2064
2065 \seq_clear:c {
2066   l_stex_symdecl_
2067   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2068   _notations
2069 }
2070
2071 \bool_if:NF \l_stex_symdecl_local_bool {
2072   \exp_args:Nx
2073   \stex_add_to_current_module:n {
2074     \seq_clear:c {
2075       l_stex_symdecl_
2076       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2077       _notations
2078     }
2079     \prop_set_from_keyval:cn {
2080       l_stex_symdecl_
2081       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2082       _prop
2083     } {
2084       name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2085       module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2086       type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2087       args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2088       arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2089       assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
2090     }
2091   }
2092 }
2093
2094 \stex_if_smsmode:TF {
2095   \bool_if:NF \l_stex_symdecl_local_bool {
2096     % \exp_args:Nx \stex_add_to_sms:n {
2097     %   \prop_set_from_keyval:cn {
2098     %     l_stex_symdecl_
2099     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2100     %     _prop
2101     %   } {
2102     %     name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2103     %     module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2104     %     local     = \prop_item:Nn \l_tmpa_prop { local }     ,
2105     %     type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2106     %     args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2107     %     arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2108     %     assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
2109     %   }
2110     %   \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2111     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2112     %   }

```



```

2113 %      }
2114     }
2115   }{
2116     \exp_args:Nx \stex_do_aftergroup:n {
2117       \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2118         \l_stex_current_module_str ? \l_stex_symdecl_name_str
2119       }
2120     }
2121     \stex_if_do_html:T {
2122       \stex_annotate_invisible:nnn {symdecl} {
2123         \l_stex_current_module_str ? \l_stex_symdecl_name_str
2124       } {
2125         \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}{}{${\l_st
2126         \stex_annotate_invisible:nnn{args}{}{
2127           \prop_item:Nn \l_tmpa_prop { args }
2128         }
2129         \stex_annotate_invisible:nnn{macroname}{#1}{}
2130         \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2131           \stex_annotate_invisible:nnn{definiens}{}
2132             {${\l_stex_symdecl_definiens_tl$}
2133         }
2134       }
2135     }
2136   }
2137 }

```

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

`\stex_get_symbol:n`

```

2138 \str_new:N \l_stex_get_symbol_uri_str
2139
2140 \cs_new_protected:Nn \stex_get_symbol:n {
2141   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2142     \__stex_symdecl_get_symbol_from_cs:n { #1 }
2143   }{
2144     % argument is a string
2145     % is it a command name?
2146     \cs_if_exist:cTF { #1 }{
2147       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2148       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2149       \str_if_empty:NTF \l_tmpa_str {
2150         \exp_args:Nx \cs_if_eq:NNTF {
2151           \tl_head:N \l_tmpa_tl
2152         } \stex_invoke_symbol:n {
2153           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
2154         }{
2155           \__stex_symdecl_get_symbol_from_string:n { #1 }
2156         }
2157       } {
2158         \__stex_symdecl_get_symbol_from_string:n { #1 }
2159       }
2160     }{
2161       % argument is not a command name
2162       \__stex_symdecl_get_symbol_from_string:n { #1 }

```

```

2163     % \l_stex_all_symbols_seq
2164 }
2165 }
2166 \str_if_eq:eeF {
2167   \prop_item:cn {
2168     l_stex_symdecl\l_stex_get_symbol_uri_str _prop
2169   }{ deprecate }
2170 }{}{
2171   \msg_warning:nnxx{stex}{warning/deprecated}{
2172     Symbol~\l_stex_get_symbol_uri_str
2173   }{
2174     \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{ deprecate }
2175   }
2176 }
2177 }
2178
2179 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2180   \str_set:Nn \l_tmpa_str { #1 }
2181   \bool_set_false:N \l_tmpa_bool
2182   \stex_if_in_module:T {
2183     \exp_args:Nno \seq_if_in:cnT {c_stex_module\l_stex_current_module_str _constants} { \l_
2184       \bool_set_true:N \l_tmpa_bool
2185       \str_set:Nx \l_stex_get_symbol_uri_str {
2186         \l_stex_current_module_str ? #1
2187       }
2188     }
2189   }
2190   \bool_if:NF \l_tmpa_bool {
2191     \tl_set:Nn \l_tmpa_tl {
2192       \msg_set:nnn{stex}{error/unknownsymbol}{
2193         No~symbol~#1~found!
2194       }
2195       \msg_error:nn{stex}{error/unknownsymbol}
2196     }
2197     \str_set:Nn \l_tmpa_str { #1 }
2198     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2199     \seq_map_inline:Nn \l_stex_all_symbols_seq {
2200       \str_set:Nn \l_tmpb_str { ##1 }
2201       \str_if_eq:eeT { \l_tmpa_str } {
2202         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
2203       } {
2204         \seq_map_break:n {
2205           \tl_set:Nn \l_tmpa_tl {
2206             \str_set:Nn \l_stex_get_symbol_uri_str {
2207               ##1
2208             }
2209           }
2210         }
2211       }
2212     }
2213     \l_tmpa_tl
2214   }
2215 }
2216

```

```

2217 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
2218   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2219     { \tl_tail:N \l_tmpa_tl }
2220   \tl_if_single:NTF \l_tmpa_tl {
2221     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2222       \exp_after:wN \str_set:Nn \exp_after:wN
2223         \l_stex_get_symbol_uri_str \l_tmpa_tl
2224     }{
2225       % TODO
2226       % tail is not a single group
2227     }
2228   }{
2229     % TODO
2230     % tail is not a single group
2231   }
2232 }

```

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

## 30.2 Notations

```

2233 <@@=stex_notation>
      notation arguments:
2234 \keys_define:nn { stex / notation } {
2235   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2236   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2237   prec .str_set_x:N = \l__stex_notation_prec_str ,
2238   op .tl_set:N = \l__stex_notation_op_tl ,
2239   primary .bool_set:N = \l__stex_notation_primary_bool ,
2240   primary .default:n = {true} ,
2241   unknown .code:n = \str_set:Nx
2242     \l__stex_notation_variant_str \l_keys_key_str
2243 }
2244
2245 \cs_new_protected:Nn \_stex_notation_args:n {
2246   \str_clear:N \l__stex_notation_lang_str
2247   \str_clear:N \l__stex_notation_variant_str
2248   \str_clear:N \l__stex_notation_prec_str
2249   \tl_clear:N \l__stex_notation_op_tl
2250   \bool_set_false:N \l__stex_notation_primary_bool
2251
2252   \keys_set:nn { stex / notation } { #1 }
2253 }

```

**\notation**

```

2254 \NewDocumentCommand \notation { 0{ } m } {
2255   \_stex_notation_args:n { #1 }
2256   \tl_clear:N \l_stex_symdecl_definiens_tl
2257   \stex_get_symbol:n { #2 }
2258   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
2259 }
2260 \stex_deactivate_macro:Nn \notation {module~environments}

```

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

`\stex_notation_do:nn`

```

2261 \seq_new:N \l__stex_notation_precedences_seq
2262 \tl_new:N \l__stex_notation_opprec_tl
2263 \int_new:N \l__stex_notation_currarg_int
2264
2265 \cs_new_protected:Nn \stex_notation_do:nn {
2266   \let\l_stex_current_symbol_str\relax
2267   \str_set:Nx \l__stex_notation_symbol_str { #1 }
2268   \seq_clear:N \l__stex_notation_precedences_seq
2269   \tl_clear:N \l__stex_notation_opprec_tl
2270   \prop_get:cnN {
2271     l_stex_symdecl_ #1 _prop
2272   } { args } \l__stex_notation_args_str
2273
2274   % precedences
2275   \prop_get:cnN {
2276     l_stex_symdecl_ #1 _prop
2277   } { arity } \l__stex_notation_arity_str
2278   \str_if_empty:NTF \l__stex_notation_prec_str {
2279     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2280       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2281     }{
2282       \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2283     }
2284   } {
2285     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2286       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2287       \int_step_inline:nn { \l__stex_notation_arity_str } {
2288         \exp_args:NNo
2289         \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2290       }
2291     }{
2292       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2293       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2294         \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2295         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2296           \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2297             \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2298           \seq_map_inline:Nn \l_tmpa_seq {
2299             \seq_put_right:Nn \l_tmpb_seq { ##1 }
2300           }
2301         }
2302       }{
2303         \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2304           \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2305         }{
2306           \tl_set:No \l__stex_notation_opprec_tl { 0 }
2307         }
2308       }
2309     }
2310   }
2311
2312   \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2313   \int_step_inline:nn { \l__stex_notation_arity_str } {

```

```

2314 \seq_pop_left:NnF \l_tmpa_seq \l_tmpb_str {
2315 \exp_args:NNo
2316 \seq_put_right:No \l__stex_notation_precedences_seq {
2317 \l__stex_notation_opprec_tl
2318 }
2319 }
2320 }
2321
2322 \tl_clear:N \l__stex_notation_dummyargs_tl
2323
2324 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2325 \exp_args:NNe
2326 \cs_set:Npn \l__stex_notation_macrocode_cs {
2327 \stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2328 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2329 { \l__stex_notation_opprec_tl }
2330 { \exp_not:n { #2 } }
2331 }
2332 \__stex_notation_final:
2333 }{
2334 \str_if_in:NnTF \l__stex_notation_args_str b {
2335 \exp_args:Nne \use:nn
2336 {
2337 \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2338 \cs_set:Npn \l__stex_notation_arity_str } { {
2339 \stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2340 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2341 { \l__stex_notation_opprec_tl }
2342 { \exp_not:n { #2 } }
2343 } }
2344 }{
2345 \str_if_in:NnTF \l__stex_notation_args_str B {
2346 \exp_args:Nne \use:nn
2347 {
2348 \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2349 \cs_set:Npn \l__stex_notation_arity_str } { {
2350 \stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2351 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2352 { \l__stex_notation_opprec_tl }
2353 { \exp_not:n { #2 } }
2354 } }
2355 }{
2356 \exp_args:Nne \use:nn
2357 {
2358 \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2359 \cs_set:Npn \l__stex_notation_arity_str } { {
2360 \stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2361 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2362 { \l__stex_notation_opprec_tl }
2363 { \exp_not:n { #2 } }
2364 } }
2365 }
2366 }
2367

```

```

2368 \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2369 \int_zero:N \l__stex_notation_currarg_int
2370 \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2371 \__stex_notation_arguments:
2372 }
2373 }

```

(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

```

2374 \cs_new_protected:Nn \__stex_notation_arguments: {
2375 \int_incr:N \l__stex_notation_currarg_int
2376 \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2377 \__stex_notation_final:
2378 }{
2379 \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2380 \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2381 \str_if_eq:VnTF \l_tmpa_str a {
2382 \__stex_notation_argument_assoc:n
2383 }{
2384 \str_if_eq:VnTF \l_tmpa_str B {
2385 \__stex_notation_argument_assoc:n
2386 }{
2387 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2388 \tl_put_right:Nx \l__stex_notation_dummyargs_tl {
2389 { \stex_term_math_arg:nnn
2390 { \int_use:N \l__stex_notation_currarg_int }
2391 { \l_tmpa_str }
2392 { ###\int_use:N \l__stex_notation_currarg_int }
2393 }
2394 }
2395 \__stex_notation_arguments:
2396 }
2397 }
2398 }
2399 }

```

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

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

```

2400 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
2401
2402 \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2403 {\l__stex_notation_arity_str}{
2404 #1
2405 }
2406 \int_zero:N \l_tmpa_int
2407 \tl_clear:N \l_tmpa_tl
2408 \str_map_inline:Nn \l__stex_notation_args_str {
2409 \int_incr:N \l_tmpa_int
2410 \tl_put_right:Nx \l_tmpa_tl {
2411 \str_if_eq:nnTF {##1}{a}{ {} }{
2412 \str_if_eq:nnTF {##1}{B}{ {} }{
2413 {##### \int_use:N \l_tmpa_int}

```

```

2414     }
2415   }
2416 }
2417 }
2418 \exp_after:wN\exp_after:wN\exp_after:wN \def
2419 \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2420 \exp_after:wN\exp_after:wN\exp_after:wN ##
2421 \exp_after:wN\exp_after:wN\exp_after:wN 1
2422 \exp_after:wN\exp_after:wN\exp_after:wN ##
2423 \exp_after:wN\exp_after:wN\exp_after:wN 2
2424 \exp_after:wN\exp_after:wN\exp_after:wN {
2425   \exp_after:wN \exp_after:wN \exp_after:wN
2426   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2427     \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2428   }
2429 }
2430
2431 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2432 \tl_put_right:Nx \l__stex_notation_dummyargs_tl { {
2433   \stex_term_math_assoc_arg:nnnn
2434   { \int_use:N \l__stex_notation_currarg_int }
2435   { \l_tmpa_str }
2436   { ####\int_use:N \l__stex_notation_currarg_int }
2437   { \l_tmpa_cs {####1} {####2} }
2438 } }
2439 %\cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
2440 %\tl_put_right:Nx \l_tmpa_tl {
2441 % { \stex_term_math_assoc_arg:nnnn
2442 %   { \int_use:N \l_tmpa_int }
2443 %   { \l_tmpb_str }
2444 %   \exp_args:No \exp_not:n
2445 %   {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
2446 %   { ####\int_use:N \l_tmpa_int }
2447 % }
2448 %}
2449 \__stex_notation_arguments:
2450 }

```

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

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

```

2451 \cs_new_protected:Nn \__stex_notation_final: {
2452   \exp_args:Nne \use:nn
2453   {
2454     \cs_generate_from_arg_count:cNnn {
2455       stex_notation_ \l__stex_notation_symbol_str \c_hash_str
2456       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2457       _cs
2458     }
2459     \cs_set:Npn \l__stex_notation_arity_str } { {
2460       \exp_after:wN \exp_after:wN \exp_after:wN
2461       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2462       { \exp_after:wN \l__stex_notation_macrocode_cs \l__stex_notation_dummyargs_tl }
2463     } }

```

```

2464
2465 \tl_if_empty:NF \l__stex_notation_op_tl {
2466   \cs_set:cpx {
2467     stex_op_notation_ \l__stex_notation_symbol_str \c_hash_str
2468     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2469     _cs
2470   } {
2471     \stex_term_oms:nnn {
2472       \l__stex_notation_symbol_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2473       \l__stex_notation_lang_str
2474     }{
2475       \l__stex_notation_symbol_str
2476     }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2477   }
2478 }
2479
2480 \exp_args:Ne
2481 \stex_add_to_current_module:n {
2482   \cs_generate_from_arg_count:cNnn {
2483     stex_notation_ \l__stex_notation_symbol_str \c_hash_str
2484     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2485     _cs
2486   } \cs_set:Npn {\l__stex_notation_arity_str} {
2487     \exp_after:wN \exp_after:wN \exp_after:wN
2488     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2489     { \exp_after:wN \l__stex_notation_macrocode_cs \l__stex_notation_dummyargs_tl }
2490   }
2491   \tl_if_empty:NF \l__stex_notation_op_tl {
2492     \cs_set:cpn {
2493       stex_op_notation_ \l__stex_notation_symbol_str \c_hash_str
2494       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2495       _cs
2496     } {
2497       \stex_term_oms:nnn {
2498         \l__stex_notation_symbol_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2499         \l__stex_notation_lang_str
2500       }{
2501         \l__stex_notation_symbol_str
2502       }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2503     }
2504   }
2505 }
2506 \exp_args:Nx
2507 % \stex_do_aftergroup:n {
2508   \seq_put_right:cx {
2509     l_stex_symdecl_ \l__stex_notation_symbol_str
2510     _notations
2511   } {
2512     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2513   }
2514 % }
2515
2516 \stex_debug:nn{symbols}{
2517   Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str

```



```

2518 ~for~\l__stex_notation_symbol_str^^J
2519 Operator~precedence::~\l__stex_notation_opprec_tl^^J
2520 Argument~precedences::~
2521   \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2522 Notation: \cs_meaning:c {
2523   stex_notation_ \l__stex_notation_symbol_str \c_hash_str
2524   \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2525   _cs
2526 }
2527 }
2528
2529 %\prop_set_eq:cn {
2530 %   l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2531 %   \c_hash_str \l__stex_notation_lang_str _prop
2532 %} \l_tmpb_prop
2533
2534 \exp_args:Ne
2535 \stex_add_to_current_module:n {
2536   \seq_put_right:cn {
2537     l_stex_symdecl_ \l__stex_notation_symbol_str
2538     _notations
2539   } {
2540     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2541   }
2542   %\prop_set_from_keyval:cn {
2543   %   l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2544   %   \c_hash_str \l__stex_notation_lang_str _prop
2545   %} {
2546   %   symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2547   %   language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2548   %   variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,
2549   %   opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2550   %   argprecs    = \prop_item:Nn \l_tmpb_prop { argprecs }    ,
2551   %}
2552 }
2553
2554 \stex_if_smsmode:TF {
2555 %   \exp_args:Nx \stex_add_to_sms:n {
2556 %     \prop_set_from_keyval:cn {
2557 %       l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2558 %       \c_hash_str \l__stex_notation_lang_str _prop
2559 %     } {
2560 %       symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2561 %       language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2562 %       variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,
2563 %       opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2564 %       argprecs    = \prop_item:Nn \l_tmpb_prop { argprecs }    ,
2565 %     }
2566 %   }
2567 }{
2568
2569 % HTML annotations
2570 \stex_if_do_html:T {
2571   \stex_annotate_invisible:nnn { notation }

```

```

2572 { \l__stex_notation_symbol_str } {
2573   \stex_annotate_invisible:nnn { notationfragment }
2574   { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
2575   \stex_annotate_invisible:nnn { precedence }
2576   { \l__stex_notation_prec_str }{}
2577
2578   \int_zero:N \l_tmpa_int
2579   \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2580   \tl_clear:N \l_tmpa_tl
2581   \int_step_inline:nn { \l__stex_notation_arity_str }{
2582     \int_incr:N \l_tmpa_int
2583     \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2584     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_r
2585     \str_if_eq:VnTF \l_tmpb_str a {
2586       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2587         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2588         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2589       } }
2590     }{
2591       \str_if_eq:VnTF \l_tmpb_str B {
2592         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2593           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2594           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2595         } }
2596       }{
2597         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2598           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2599         } }
2600     }
2601   }
2602 }
2603 \stex_annotate_invisible:nnn { notationcomp }{}{
2604   \str_set:Nx \l_stex_current_symbol_str { \l__stex_notation_symbol_str }
2605   $ \exp_args:Nno \use:nn { \use:c {
2606     stex_notation_ \l_stex_current_symbol_str
2607     \c_hash_str \l__stex_notation_variant_str
2608     \c_hash_str \l__stex_notation_lang_str _cs
2609   } } { \l_tmpa_tl } $
2610 }
2611 }
2612 }
2613 }
2614 \stex_smsmode_do:
2615 }

```

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

\setnotation

```

2616 \keys_define:nn { stex / setnotation } {
2617   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2618   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2619   unknown .code:n = \str_set:Nx
2620     \l__stex_notation_variant_str \l_keys_key_str
2621 }

```

```

2622
2623 \cs_new_protected:Nn \stex_setnotation_args:n {
2624   \str_clear:N \l__stex_notation_lang_str
2625   \str_clear:N \l__stex_notation_variant_str
2626   \keys_set:nn { stex / setnotation } { #1 }
2627 }
2628
2629 \NewDocumentCommand \setnotation {m m} {
2630   \stex_get_symbol:n { #1 }
2631   \stex_setnotation_args:n { #2 }
2632   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl \l_stex_get_symbol_uri_str _notations }
2633     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2634     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl \l_stex_get_symbol_uri_str _notation
2635       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2636       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl \l_stex_get_symbol_uri_str _notation
2637         { \c_hash_str }
2638       \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl \l_stex_get_symbol_uri_str _notations
2639         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2640       \exp_args:Nx \stex_add_to_current_module:n {
2641         \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl \l_stex_get_symbol_uri_str _notati
2642         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2643         \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl \l_stex_get_symbol_uri_str _notation
2644         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2645         \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl \l_stex_get_symbol_uri_str _notati
2646         { \c_hash_str }
2647       }
2648       \stex_debug:nn {notations}{
2649         Setting~default~notation~
2650         {\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str}~for~
2651         \l_stex_get_symbol_uri_str \
2652         \expandafter\meaning\csname
2653         l_stex_symdecl \l_stex_get_symbol_uri_str _notations\endcsname
2654       }
2655     }{
2656       % todo throw error
2657     }
2658     \stex_smsmode_do:
2659   }
2660
2661 \cs_new_protected:Nn \stex_copy_notations:nn {
2662   \stex_debug:nn {notations}{
2663     Copying~notations~from~#2~to~#1\
2664     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
2665   }
2666   \tl_clear:N \l_tmpa_tl
2667   \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2668     \tl_put_right:Nn \l_tmpa_tl { {##} ##1} }
2669   }
2670   \seq_map_inline:cn {l_stex_symdecl_#2_notations}{
2671     \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2672     \edef \l_tmpa_tl {
2673       \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2674       \exp_after:wN\exp_after:wN\exp_after:wN {
2675         \exp_after:wN \l_tmpa_cs \l_tmpa_tl

```

```

2676     }
2677   }
2678   \exp_args:Nx
2679   \stex_do_aftergroup:n {
2680     \seq_put_right:cn{l_stex_symdecl_#1_notations}{##1}
2681     \cs_generate_from_arg_count:cNnn {
2682       stex_notation_ #1 \c_hash_str ##1 _cs
2683     } \cs_set:Npn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }{
2684       \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl}
2685     }
2686   }
2687 }
2688 }
2689
2690 \NewDocumentCommand \copynotation {m m} {
2691   \stex_get_symbol:n { #1 }
2692   \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2693   \stex_get_symbol:n { #2 }
2694   \exp_args:Noo
2695   \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2696   \exp_args:Nx \stex_add_import_to_current_module:n{
2697     \stex_copy_notations:nn {\l_tmpa_str} {\l_stex_get_symbol_uri_str}
2698   }
2699   \stex_smsmode_do:
2700 }
2701

```

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

**\symdef**

```

2702 \keys_define:nn { stex / symdef } {
2703   name .str_set_x:N = \l_stex_symdecl_name_str ,
2704   local .bool_set:N = \l_stex_symdecl_local_bool ,
2705   args .str_set_x:N = \l_stex_symdecl_args_str ,
2706   type .tl_set:N = \l_stex_symdecl_type_tl ,
2707   def .tl_set:N = \l_stex_symdecl_definiens_tl ,
2708   op .tl_set:N = \l__stex_notation_op_tl ,
2709   lang .str_set_x:N = \l__stex_notation_lang_str ,
2710   variant .str_set_x:N = \l__stex_notation_variant_str ,
2711   prec .str_set_x:N = \l__stex_notation_prec_str ,
2712   unknown .code:n = \str_set:Nx
2713     \l__stex_notation_variant_str \l_keys_key_str
2714 }
2715
2716 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2717   \str_clear:N \l_stex_symdecl_name_str
2718   \str_clear:N \l_stex_symdecl_args_str
2719   \bool_set_false:N \l_stex_symdecl_local_bool
2720   \tl_clear:N \l_stex_symdecl_type_tl
2721   \tl_clear:N \l_stex_symdecl_definiens_tl
2722   \str_clear:N \l__stex_notation_lang_str
2723   \str_clear:N \l__stex_notation_variant_str
2724   \str_clear:N \l__stex_notation_prec_str
2725   \tl_clear:N \l__stex_notation_op_tl

```

```

2726 \keys_set:nn { stex / symdef } { #1 }
2727 }
2728 }
2729
2730 \NewDocumentCommand \symdef { 0{} m } {
2731   \__stex_notation_symdef_args:n { #1 }
2732   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2733   \stex_symdecl_do:n { #2 }
2734   \exp_args:Nx \stex_notation_do:nn {
2735     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2736   }
2737 }
2738 \stex_deactivate_macro:Nn \symdef {module~environments}

```

*(End definition for \symdef. This function is documented on page [37](#).)*

```

2739 \endpackage

```

## Chapter 31

# STEX -Terms Implementation

```
2740 <*package>
2741
2742 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2743
2744 <@@=stex_terms>
2745
2746   Warnings and error messages
2747 \msg_new:nnn{stex}{error/nonotation}{
2748   Symbol~#1~invoked,~but~has~no~notation~#2!
2749 }
2750 \msg_new:nnn{stex}{error/notationarg}{
2751   Error~in~parsing~notation~#1
2752 }
2753 \msg_new:nnn{stex}{error/noop}{
2754   Symbol~#1~has~no~operator~notation~for~notation~#2
2755 }
```

### 31.1 Symbol Invocations

Arguments:

```
2755 \keys_define:nn { stex / terms } {
2756   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2757   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2758   unknown .code:n = \str_set:Nx
2759     \l__stex_terms_variant_str \l_keys_key_str
2760 }
2761
2762 \cs_new_protected:Nn \__stex_terms_args:n {
2763   \str_clear:N \l__stex_terms_lang_str
2764   \str_clear:N \l__stex_terms_variant_str
2765   \str_clear:N \l__stex_terms_prec_str
2766   \tl_clear:N \l__stex_terms_op_tl
2767
2768   \keys_set:nn { stex / terms } { #1 }
```

2769 }

`\stex_invoke_symbol:n` Invokes a semantic macro

```

2770 \cs_new_protected:Nn \stex_invoke_symbol:n {
2771   \str_if_eq:eeF {
2772     \prop_item:cn {
2773       l_stex_symdecl_#1_prop
2774     }{ deprecate }
2775   }{}{
2776     \msg_warning:nnxx{stex}{warning/deprecated}{
2777       Symbol~#1
2778     }{
2779       \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
2780     }
2781   }
2782   \if_mode_math:
2783     \exp_after:wN \__stex_terms_invoke_math:n
2784   \else:
2785     \exp_after:wN \__stex_terms_invoke_text:n
2786   \fi: { #1 }
2787 }
```

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

`\__stex_terms_invoke_math:n`

```

2788 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2789   \peek_charcode_remove:NTF ! {
2790     \peek_charcode:NTF [ {
2791       \__stex_terms_invoke_op:nw { #1 }
2792     }{
2793       \peek_charcode_remove:NTF ! {
2794         \peek_charcode:NTF [ {
2795           \__stex_terms_invoke_op_custom:nw
2796         }{
2797           % TODO throw error
2798         }
2799       }{
2800         \__stex_terms_invoke_op:nw { #1 } []
2801       }
2802     }
2803   }{
2804     \peek_charcode_remove:NTF * {
2805       \__stex_terms_invoke_text:n { #1 }
2806     }{
2807       \peek_charcode:NTF [ {
2808         \__stex_terms_invoke_math:nw { #1 }
2809       }{
2810         \__stex_terms_invoke_math:nw { #1 } []
2811       }
2812     }
2813   }
2814 }
```

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

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

```

2815 \cs_new_protected:Npn \_stex_terms_invoke_op_custom:nw #1 [#2] {
2816   \_stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
2817     \stex_highlight_term:nn{#1}{#2}
2818   }
2819 }

```

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

\\_stex\_terms\_invoke\_op:nw

```

2820 \cs_new_protected:Npn \_stex_terms_invoke_op:nw #1 [#2] {
2821   \_stex_terms_args:n { #2 }
2822   \cs_if_exist:cTF {
2823     stex_op_notation_ #1 \c_hash_str
2824     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str_cs
2825   }{
2826     \csname stex_op_notation_ #1 \c_hash_str
2827       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str_cs
2828     \endcsname
2829   }{
2830     \msg_error:nnxx{stex}{error/noop}{#1}{\l__stex_terms_variant_str \c_hash_str \l__stex_t
2831   }
2832 }

```

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

\\_stex\_terms\_invoke\_math:nw

```

2833 \cs_new_protected:Npn \_stex_terms_invoke_math:nw #1 [#2] {
2834   \_stex_terms_args:n { #2 }
2835   \seq_if_empty:cTF {
2836     l_stex_symdecl_ #1 _notations
2837   } {
2838     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
2839   } {
2840     \seq_if_in:cxTF {
2841       l_stex_symdecl_ #1 _notations
2842     }
2843     { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
2844       \str_set:Nn \l_stex_current_symbol_str { #1 }
2845       \stex_debug:nn{terms}{Using~
2846         #1\c_hash_str\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str \
2847         \expandafter\meaning\csname stex_notation_ #1 \c_hash_str
2848         \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2849         _cs\endcsname
2850       }
2851       \use:c{
2852         stex_notation_ #1 \c_hash_str
2853         \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2854         _cs
2855       }
2856     }{
2857       \str_if_empty:NTF \l__stex_terms_variant_str {
2858         \str_if_empty:NTF \l__stex_terms_lang_str {
2859           \seq_get_left:cN {

```



```

2860         l_stex_symdecl_ #1 _notations
2861     } \l_tmpa_str
2862     \str_set:Nn \l_stex_current_symbol_str { #1 }
2863     \stex_debug:nn{terms}{Using~
2864         #1\c_hash_str\l_tmpa_str \l
2865         \expandafter\meaning\csname stex_notation_ #1 \c_hash_str
2866         \l_tmpa_str
2867         _cs\endcsname
2868     }
2869     \use:c{
2870         stex_notation_ #1 \c_hash_str \l_tmpa_str
2871         _cs
2872     }
2873     ){
2874         \msg_error:nnxx{stex}{error/nonotation}{#1}{
2875             ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2876         }
2877     }
2878     ){
2879         \msg_error:nnxx{stex}{error/nonotation}{#1}{
2880             ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2881         }
2882     }
2883 }
2884 }
2885 }

```

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

\\_stex\_terms\_invoke\_text:n

```

2886 \cs_new_protected:Nn \_stex_terms_invoke_text:n {
2887     \peek_charcode_remove:NTF ! {
2888         \stex_term_custom:nn { #1 } { }
2889     }{
2890         \prop_set_eq:Nc \l_tmpa_prop {
2891             l_stex_symdecl_ #1 _prop
2892         }
2893         \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2894         \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2895     }
2896 }

```

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

## 31.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
2897 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2898 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2899 \int_new:N \l__stex_terms_downprec
2900 \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`

```
2901 \tl_set:Nn \l__stex_terms_left_bracket_str (
2902 \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

```
2903 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
2904   \bool_if:NTF \l__stex_terms_brackets_done_bool {
2905     \bool_set_false:N \l__stex_terms_brackets_done_bool
2906     #2
2907   } {
2908     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
2909       \bool_if:NTF \l__stex_inarray_bool { #2 } {
2910         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
2911         \dobrackets { #2 }
2912       }
2913     }{ #2 }
2914   }
2915 }
```

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

**`\dobrackets`**

```
2916 \bool_new:N \l__stex_terms_brackets_done_bool
2917 %\RequirePackage{scalerel}
2918 \cs_new_protected:Npn \dobrackets #1 {
2919   %\ThisStyle{\if D\m@switch
2920   %   \exp_args:Nnx \use:nn
2921   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
2922   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
2923   % \else
2924   \exp_args:Nnx \use:nn
2925   {
2926     \bool_set_true:N \l__stex_terms_brackets_done_bool
2927     \int_set:Nn \l__stex_terms_downprec \infprec
2928     \l__stex_terms_left_bracket_str
2929     #1
2930   }
2931   {
2932     \bool_set_false:N \l__stex_terms_brackets_done_bool
2933     \l__stex_terms_right_bracket_str
2934     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2935   }
2936   %\fi}
2937 }
```

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

**\withbrackets**

```
2938 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2939   \exp_args:Nnx \use:nn
2940   {
2941     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
2942     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
2943     #3
2944   }
2945   {
2946     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
2947       {\l__stex_terms_left_bracket_str}
2948     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
2949       {\l__stex_terms_right_bracket_str}
2950   }
2951 }
```

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

**\STEXinvisible**

```
2952 \cs_new_protected:Npn \STEXinvisible #1 {
2953   \stex_annotate_invisible:n { #1 }
2954 }
```

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

OMDoc terms:

**\\_stex\_term\_math\_oms:nnnn**

```
2955 \cs_new_protected:Nn \_stex_term_oms:nnn {
2956   \stex_annotate:nnn{ OMID }{ #2 }{
2957     \stex_highlight_term:nn { #1 } { #3 }
2958   }
2959 }
2960
2961 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2962   \__stex_terms_maybe_brackets:nn { #3 }{
2963     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2964   }
2965 }
```

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

**\\_stex\_term\_math\_oma:nnnn**

```
2966 \cs_new_protected:Nn \_stex_term_oma:nnn {
2967   \stex_annotate:nnn{ OMA }{ #2 }{
2968     \stex_highlight_term:nn { #1 } { #3 }
2969   }
2970 }
2971
2972 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2973   \__stex_terms_maybe_brackets:nn { #3 }{
2974     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2975   }
2976 }
```

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

`\_stex_term_math_omb:nnnn`

```
2977 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2978   \stex_annotate:nnn{ OMBIND }{ #2 }{
2979     \stex_highlight_term:nn { #1 } { #3 }
2980   }
2981 }
2982
2983 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2984   \__stex_terms_maybe_brackets:nn { #3 }{
2985     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2986   }
2987 }
```

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

`\_stex_term_math_arg:nnn`

```
2988 \cs_new_protected:Nn \_stex_term_arg:nn {
2989   \stex_unhighlight_term:n {
2990     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2991   }
2992 }
2993 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
2994   \exp_args:Nnx \use:nn
2995     { \int_set:Nn \l__stex_terms_downprec { #2 }
2996       \_stex_term_arg:nn { #1 }{ #3 }
2997     }
2998   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2999 }
```

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

`\_stex_term_math_assoc_arg:nnnn`

```
3000 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
3001   % TODO sequences
3002   \clist_set:Nn \l_tmpa_clist{ #3 }
3003   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3004     \tl_set:Nn \l_tmpa_tl { #3 }
3005   }{
3006     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3007     \clist_reverse:N \l_tmpa_clist
3008     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3009
3010     \clist_map_inline:Nn \l_tmpa_clist {
3011       \exp_args:NNNo \exp_args:NNNo \tl_set:No \l_tmpa_tl {
3012         \exp_args:Nno
3013         \l_tmpa_cs { ##1 } \l_tmpa_tl
3014       }
3015     }
3016   }
3017   \exp_args:Nnno
3018   \_stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
3019 }
```

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

`\stex_term_custom:nn`

```
3020 \cs_new_protected:Nn \stex_term_custom:nn {
3021   \str_set:Nn \l__stex_terms_custom_uri { #1 }
3022   \str_set:Nn \l_tmpa_str { #2 }
3023   \tl_clear:N \l_tmpa_tl
3024   \int_zero:N \l_tmpa_int
3025   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
3026   \__stex_terms_custom_loop:
3027 }
```

(End definition for `\stex_term_custom:nn`. This function is documented on page 39.)

`\__stex_terms_custom_loop:`

```
3028 \cs_new_protected:Nn \__stex_terms_custom_loop: {
3029   \bool_set_false:N \l_tmpa_bool
3030   \bool_while_do:nn {
3031     \str_if_eq_p:ee X {
3032       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
3033     }
3034   }{
3035     \int_incr:N \l_tmpa_int
3036   }
3037
3038   \peek_charcode:NTF [ {
3039     % notation/text component
3040     \__stex_terms_custom_component:w
3041   } {
3042     \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
3043       % all arguments read => finish
3044       \__stex_terms_custom_final:
3045     } {
3046       % arguments missing
3047       \peek_charcode_remove:NTF * {
3048         % invisible, specific argument position or both
3049         \peek_charcode:NTF [ {
3050           % visible specific argument position
3051           \__stex_terms_custom_arg:wn
3052         } {
3053           % invisible
3054           \peek_charcode_remove:NTF * {
3055             % invisible specific argument position
3056             \__stex_terms_custom_arg_inv:wn
3057           } {
3058             % invisible next argument
3059             \__stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
3060           }
3061         }
3062       } {
3063         % next normal argument
3064         \__stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
3065       }
3066     }
3067   }
3068 }
```

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

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

```

3069 \cs_new_protected:Npn \_stex_terms_custom_arg_inv:wn [ #1 ] #2 {
3070   \bool_set_true:N \l_tmpa_bool
3071   \_stex_terms_custom_arg:wn [ #1 ] { #2 }
3072 }

```

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

\\_stex\_terms\_custom\_arg:wn

```

3073 \cs_new_protected:Npn \_stex_terms_custom_arg:wn [ #1 ] #2 {
3074   \str_set:Nx \l_tmpb_str {
3075     \str_item:Nn \l_tmpa_str { #1 }
3076   }
3077   \str_case:VnTF \l_tmpb_str {
3078     { X } {
3079       \msg_error:nnx{stex}{error/notationarg}{\l_stex_terms_custom_uri}
3080     }
3081     { i } { \_stex_terms_custom_set_X:n { #1 } }
3082     { b } { \_stex_terms_custom_set_X:n { #1 } }
3083     { a } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
3084     { B } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
3085   }{ }{
3086     \msg_error:nnx{stex}{error/notationarg}{\l_stex_terms_custom_uri}
3087   }
3088
3089   \bool_if:nTF \l_tmpa_bool {
3090     \tl_put_right:Nx \l_tmpa_tl {
3091       \stex_annotate_invisible:n {
3092         \_stex_term_arg:nn { \int_eval:n { #1 } }
3093         \exp_not:n { { #2 } }
3094       }
3095     }
3096   } {
3097     \tl_put_right:Nx \l_tmpa_tl {
3098       \_stex_term_arg:nn { \int_eval:n { #1 } }
3099       \exp_not:n { { #2 } }
3100     }
3101   }
3102
3103   \_stex_terms_custom_loop:
3104 }

```

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

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

```

3105 \cs_new_protected:Nn \_stex_terms_custom_set_X:n {
3106   \str_set:Nx \l_tmpa_str {
3107     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
3108     X
3109     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
3110   }
3111 }

```

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

\\_stex\_terms\_custom\_component:

```

3112 \cs_new_protected:Npn \_stex_terms_custom_component:w [ #1 ] {
3113   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
3114   \_stex_terms_custom_loop:
3115 }

```

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

\\_stex\_terms\_custom\_final:

```

3116 \cs_new_protected:Nn \_stex_terms_custom_final: {
3117   \int_compare:nNnTF \l_tmpb_int = 0 {
3118     \exp_args:Nnno \_stex_term_oms:nnn
3119   }{
3120     \str_if_in:NnTF \l_tmpa_str {b} {
3121       \exp_args:Nnno \_stex_term_ombind:nnn
3122     } {
3123       \exp_args:Nnno \_stex_term_oma:nnn
3124     }
3125   }
3126   { \l_stex_terms_custom_uri } { \l_stex_terms_custom_uri } { \l_tmpa_tl }
3127 }

```

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

\symref  
\symname

```

3128 \NewDocumentCommand \symref { m m }{
3129   \let\compemph_uri_prev:\compemph@uri
3130   \let\compemph@uri\symrefemph@uri
3131   \STEXsymbol{#1}!{#2}
3132   \let\compemph@uri\compemph_uri_prev:
3133 }
3134
3135 \keys_define:nn { stex / symname } {
3136   post .str_set_x:N = \l_stex_symname_post_str
3137 }
3138
3139 \cs_new_protected:Nn \stex_symname_args:n {
3140   \str_clear:N \l_stex_symname_post_str
3141   \keys_set:nn { stex / symname } { #1 }
3142 }
3143
3144 \NewDocumentCommand \symname { 0{ } m }{
3145   \stex_symname_args:n { #1 }
3146   \stex_get_symbol:n { #2 }
3147   \str_set:Nx \l_tmpa_str {
3148     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3149   }
3150   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3151
3152   \let\compemph_uri_prev:\compemph@uri
3153   \let\compemph@uri\symrefemph@uri
3154   \exp_args:NNx \use:nn

```

```

3155 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }![
3156 \l_tmpa_str \l_stex_symname_post_str
3157 ] }
3158 \let\compemph@uri\compemph_uri_prev:
3159 }

```

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

## 31.3 Notation Components

```

3160 <@@=stex_notationcomps>

```

`\stex_highlight_term:nn`

```

3161
3162 \str_new:N \l_stex_current_symbol_str
3163 \cs_new_protected:Nn \stex_highlight_term:nn {
3164 \exp_args:Nnx
3165 \use:nn {
3166 \str_set:Nx \l_stex_current_symbol_str { #1 }
3167 #2
3168 } {
3169 \str_set:Nx \exp_not:N \l_stex_current_symbol_str
3170 { \l_stex_current_symbol_str }
3171 }
3172 }
3173
3174 \cs_new_protected:Nn \stex_unhighlight_term:n {
3175 % \latexml_if:TF {
3176 % #1
3177 % } {
3178 % \rustex_if:TF {
3179 % #1
3180 % } {
3181 % #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
3182 % }
3183 % }
3184 }

```

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

```

\comp
\compemph@uri
\compemph
\defemph
\defemph@uri
\symrefemph
\symrefemph@uri
3185 \cs_new_protected:Npn \comp #1 {
3186 \str_if_empty:NF \l_stex_current_symbol_str {
3187 \rustex_if:TF {
3188 \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
3189 }{
3190 \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3191 }
3192 }
3193 }
3194
3195 \cs_new_protected:Npn \compemph@uri #1 #2 {
3196 \compemph{ #1 }
3197 }

```



```

3198
3199
3200 \cs_new_protected:Npn \compemph #1 {
3201     #1
3202 }
3203
3204 \cs_new_protected:Npn \defemph@uri #1 #2 {
3205     \defemph{#1}
3206 }
3207
3208 \cs_new_protected:Npn \defemph #1 {
3209     \textbf{#1}
3210 }
3211
3212 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3213     \symrefemph{#1}
3214 }
3215
3216 \cs_new_protected:Npn \symrefemph #1 {
3217     \textbf{#1}
3218 }

```

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

## `\ellipses`

```

3219 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix 3220 \bool_new:N \l_stex_inarray_bool
\parrayline 3221 \bool_set_false:N \l_stex_inarray_bool
\parraylineh 3222 \NewDocumentCommand \parray { m m } {
\parraycell 3223     \begin{group}
3224     \bool_set_true:N \l_stex_inarray_bool
3225     \begin{array}{#1}
3226         #2
3227     \end{array}
3228     \end{group}
3229 }
3230
3231 \NewDocumentCommand \prmatrix { m } {
3232     \begin{group}
3233     \bool_set_true:N \l_stex_inarray_bool
3234     \begin{matrix}
3235         #1
3236     \end{matrix}
3237     \end{group}
3238 }
3239
3240 \def \maybepline {
3241     \bool_if:NT \l_stex_inarray_bool {\hline}
3242 }
3243
3244 \def \parrayline #1 #2 {

```

```

3245   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}\}
3246 }
3247
3248 \def \pmrow #1 { \parrayline{ }{ #1 } }
3249
3250 \def \parraylineh #1 #2 {
3251   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}\hline}
3252 }
3253
3254 \def \parraycell #1 {
3255   #1 \bool_if:NT \l_stex_inarray_bool {&}
3256 }

```

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

```

3257 \endpackage

```

## Chapter 32

# STEX -Structural Features Implementation

```
3258 <*package>
3259
3260 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3261
3262 <@@=stex_features>
3263
3264 Warnings and error messages
3265 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3266   Symbol~#1~can~not~be~assigned~in~copymodule~#2
3267 }
3268 \msg_new:nnn{stex}{error/interpretmodule/noddefinens}{
3269   Symbol~#1~not~assigned~in~interpretmodule~#2
3270 }
```

### 32.1 Imports with modification

```
3270 \cs_new_protected:Nn \stex_get_symbol_in_copymodule:n {
3271   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3272     \__stex_features_get_symbol_from_cs:n { #1 }
3273   }{
3274     % argument is a string
3275     % is it a command name?
3276     \cs_if_exist:cTF { #1 }{
3277       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3278       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3279       \str_if_empty:NNTF \l_tmpa_str {
3280         \exp_args:Nx \cs_if_eq:NNTF {
3281           \tl_head:N \l_tmpa_tl
3282         } \stex_invoke_symbol:n {
3283           \exp_args:No \__stex_features_get_symbol_from_cs:n { \use:c { #1 } }
3284         }{
3285           \__stex_features_get_symbol_from_string:n { #1 }
3286         }
3287       }
3288     }
3289   }
```

```

3286     }
3287   } {
3288     \__stex_features_get_symbol_from_string:n { #1 }
3289   }
3290   ){
3291     % argument is not a command name
3292     \__stex_features_get_symbol_from_string:n { #1 }
3293     % \l_stex_all_symbols_seq
3294   }
3295 }
3296 }
3297
3298 \cs_new_protected:Nn \__stex_features_get_symbol_from_string:n {
3299   \str_set:Nn \l_tmpa_str { #1 }
3300   \bool_set_false:N \l_tmpa_bool
3301   \bool_if:NF \l_tmpa_bool {
3302     \tl_set:Nn \l_tmpa_tl {
3303       \msg_set:nnn{stex}{error/unknownsymbol}{
3304         No~symbol~#1~found!
3305       }
3306       \msg_error:nn{stex}{error/unknownsymbol}
3307     }
3308     \str_set:Nn \l_tmpa_str { #1 }
3309     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3310     \seq_map_inline:Nn \l__stex_features_copymodule_fields_seq {
3311       \str_set:Nn \l_tmpb_str { ##1 }
3312       \str_if_eq:eeT { \l_tmpa_str } {
3313         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3314       } {
3315         \seq_map_break:n {
3316           \tl_set:Nn \l_tmpa_tl {
3317             \str_set:Nn \l_stex_get_symbol_uri_str {
3318               ##1
3319             }
3320             \__stex_features_get_symbol_check:
3321           }
3322         }
3323       }
3324     }
3325     \l_tmpa_tl
3326   }
3327 }
3328
3329 \cs_new_protected:Nn \__stex_features_get_symbol_from_cs:n {
3330   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3331   { \tl_tail:N \l_tmpa_tl }
3332   \tl_if_single:NTF \l_tmpa_tl {
3333     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3334       \exp_after:wN \str_set:Nn \exp_after:wN
3335       \l_stex_get_symbol_uri_str \l_tmpa_tl
3336       \__stex_features_get_symbol_check:
3337     }{
3338       % TODO
3339       % tail is not a single group

```

```

3340     }
3341   }{
3342     % TODO
3343     % tail is not a single group
3344   }
3345 }
3346
3347 \cs_new_protected:Nn \__stex_features_get_symbol_check: {
3348   \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq {?} \l_stex_get_symbol_uri_str
3349   \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 3 {
3350     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
3351     \str_set:Nx \l_tmpa_str {\seq_use:Nn \l_tmpa_seq ?}
3352     \seq_if_in:NoF \l__stex_features_copymodule_modules_seq \l_tmpa_str {
3353       \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3354         \l_stex_current_copymodule_name_str\Allowed:~\seq_use:Nn \l__stex_features_copymodule_modules_seq \l_tmpa_str
3355       }
3356     }
3357   }{
3358     \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3359       \l_stex_current_copymodule_name_str~(inexplicably)
3360     }
3361   }
3362 }
3363
3364 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
3365   \stex_import_module_uri:nn { #1 } { #2 }
3366   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3367   \stex_import_require_module:nnnn
3368     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3369     { \l_stex_import_path_str } { \l_stex_import_name_str }
3370   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3371   \seq_set_eq:NN \l__stex_features_copymodule_modules_seq \l_stex_collect_imports_seq
3372   \seq_clear:N \l__stex_features_copymodule_fields_seq
3373   \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3374     \seq_map_inline:cn {c_stex_module_###1_constants}{
3375       \exp_args:NNx \seq_put_right:Nn \l__stex_features_copymodule_fields_seq {
3376         ###1 ? #####1
3377       }
3378     }
3379   }
3380   \seq_clear:N \l_tmpa_seq
3381   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3382     name      = \l_stex_current_copymodule_name_str ,
3383     module    = \l_stex_current_module_str ,
3384     from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3385     includes  = \l_tmpa_seq ,
3386     fields    = \l_tmpa_seq
3387   }
3388   \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3389     as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3390   \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_features_copymodule_modules_seq
3391     \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_features_copymodule_fields_seq {,~}
3392   \stex_if_smsmode:F {
3393     \begin{stex_annotate_env} {#4} {

```

```

3394     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3395   }
3396   \stex_annotate_invisible:nnn{from}{\l_stex_import_ns_str ?\l_stex_import_name_str}{\}
3397 }
3398 \bool_set_eq:NN \l__stex_features_oldhtml_bool \l_stex_html_do_output_bool
3399 \bool_set_false:N \l_stex_html_do_output_bool
3400 }
3401 \cs_new_protected:Nn \stex_copymodule_end:n {
3402   \def \l_tmpa_cs ##1 ##2 {#1}
3403   \bool_set_eq:NN \l_stex_html_do_output_bool \l__stex_features_oldhtml_bool
3404   \tl_clear:N \l_tmpa_tl
3405   \tl_clear:N \l_tmpb_tl
3406   \prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3407   \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3408     \seq_map_inline:cn {c_stex_module_##1_constants}{
3409       \tl_clear:N \l_tmpc_tl
3410       \l_tmpa_cs{##1}{####1}
3411       \str_if_exist:cTF {l__stex_features_copymodule_##1?####1_name_str} {
3412         \tl_put_right:Nx \l_tmpa_tl {
3413           \prop_set_from_keyval:cn {
3414             l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_#
3415           }{
3416             \exp_after:wN \prop_to_keyval:N \csname
3417               l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_#
3418             \endcsname
3419           }
3420           \seq_clear:c {
3421             l_stex_symdecl_
3422             \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_name_s
3423             _notations
3424           }
3425         }
3426         \tl_put_right:Nx \l_tmpc_tl {
3427           \stex_copy_notations:nn {\l_stex_current_module_str ? \use:c{l__stex_features_cop
3428           \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_features_copymodule_##1?####1_n
3429         }
3430         \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{l__stex_features_
3431         \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3432           \tl_put_right:Nx \l_tmpc_tl {
3433             \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?#
3434           }
3435           \tl_put_right:Nx \l_tmpa_tl {
3436             \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{\}
3437             \stex_invoke_symbol:n {
3438               \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_na
3439             }
3440           }
3441         }
3442       }
3443     }{
3444       \tl_put_right:Nx \l_tmpc_tl {
3445         \stex_copy_notations:nn {\l_stex_current_module_str ? \l_stex_current_copymodule_n
3446       }
3447       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1_prop}

```

```

3448 \prop_put:Nnx \l_tmpa_prop { name }{ \l_stex_current_copymodule_name_str / ####1 }
3449 \prop_put:Nnx \l_tmpa_prop { module }{ \l_stex_current_module_str }
3450 \tl_put_right:Nx \l_tmpa_tl {
3451   \prop_set_from_keyval:cn {
3452     l_stex_symdecl\l_stex_current_module_str ? \l_stex_current_copymodule_name_str
3453   }{
3454     \prop_to_keyval:N \l_tmpa_prop
3455   }
3456   \seq_clear:c {
3457     l_stex_symdecl_
3458     \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3459     _notations
3460   }
3461 }
3462 \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \l_stex_current_copymodule
3463 \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3464   \tl_put_right:Nx \l_tmpc_tl {
3465     \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?#
3466   }
3467   \tl_put_right:Nx \l_tmpa_tl {
3468     \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3469     \stex_invoke_symbol:n {
3470       \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3471     }
3472   }
3473 }
3474 }
3475 }
3476 \tl_if_exist:cT {l__stex_features_copymodule_##1?####1_def_tl}{
3477   \tl_put_right:Nx \l_tmpc_tl {
3478     \stex_annotate_invisible:nnn{definiens}{\use:c{l__stex_features_copymodule_##1?#
3479   }
3480 }
3481 \tl_put_right:Nx \l_tmpb_tl {
3482   \stex_annotate:nnn{assignment} {##1?####1} { \l_tmpc_tl }
3483 }
3484 }
3485 }
3486 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3487 \tl_put_left:Nx \l_tmpa_tl {
3488   \prop_set_from_keyval:cn {
3489     l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _pro
3490   }{
3491     \prop_to_keyval:N \l_stex_current_copymodule_prop
3492   }
3493 }
3494 \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl
3495 \stex_debug:nn{copymodule}{result:\meaning \l_tmpa_tl}
3496 \exp_args:Nx \stex_do_aftergroup:n {
3497   \exp_args:No \exp_not:n \l_tmpa_tl
3498 }
3499 \l_tmpb_tl
3500 \stex_if_smsmode:F {
3501   \end{stex_annotate_env}

```

```

3502 }
3503 }
3504
3505 \NewDocumentEnvironment {copymodule} { 0{} m m}{
3506   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ structure }
3507   \stex_deactivate_macro:Nn \symdecl {module~environments}
3508   \stex_deactivate_macro:Nn \symdef {module~environments}
3509   \stex_deactivate_macro:Nn \notation {module~environments}
3510   \stex_reactivate_macro:N \assign
3511   \stex_reactivate_macro:N \renamedec1
3512   \stex_reactivate_macro:N \donotcopy
3513   \stex_smsmode_do:
3514 }{
3515   \stex_copymodule_end:n {}
3516 }
3517
3518 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
3519   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ realization }
3520   \stex_deactivate_macro:Nn \symdecl {module~environments}
3521   \stex_deactivate_macro:Nn \symdef {module~environments}
3522   \stex_deactivate_macro:Nn \notation {module~environments}
3523   \stex_reactivate_macro:N \assign
3524   \stex_reactivate_macro:N \renamedec1
3525   \stex_reactivate_macro:N \donotcopy
3526   \stex_smsmode_do:
3527 }{
3528   \stex_copymodule_end:n {
3529     \tl_if_exist:cF {
3530       l__stex_features_copymodule_##1?##2_def_tl
3531     }{
3532       \msg_error:nxxx{stex}{error/interpretmodule/nodedefiniens}{
3533         ##1?##2
3534       }{\l_stex_current_copymodule_name_str}
3535     }
3536   }
3537 }
3538
3539 \NewDocumentCommand \donotcopy { 0{} m}{
3540   \stex_import_module_uri:nn { #1 } { #2 }
3541   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3542   \seq_map_inline:Nn \l_stex_collect_imports_seq {
3543     \seq_remove_all:Nn \l_stex_features_copymodule_modules_seq { ##1 }
3544     \seq_map_inline:cn {c_stex_module_##1_constants}{
3545       \seq_remove_all:Nn \l_stex_features_copymodule_fields_seq { ##1 ? #####1 }
3546       \bool_lazy_any_p:nT {
3547         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_name_str}}
3548         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_macroname_str}}
3549         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_def_tl}}
3550       }{
3551         % TODO throw error
3552       }
3553     }
3554   }
3555 }

```



```

3556 \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
3557 \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_ns_str ?\l_stex_import_name_str }
3558 \prop_put:Nnx \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
3559 }
3560
3561 \NewDocumentCommand \assign { m m }{
3562   \stex_get_symbol_in_copymodule:n {#1}
3563   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
3564   \tl_set:cn {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _def_tl}{#2}
3565 }
3566
3567 \keys_define:nn { stex / renamedec1 } {
3568   name .str_set_x:N = \l_stex_renamedec1_name_str
3569 }
3570 \cs_new_protected:Nn \__stex_features_renamedec1_args:n {
3571   \str_clear:N \l_stex_renamedec1_name_str
3572
3573   \keys_set:nn { stex / renamedec1 } { #1 }
3574 }
3575
3576 \NewDocumentCommand \renamedec1 { O{} m m }{
3577   \__stex_features_renamedec1_args:n { #1 }
3578   \stex_get_symbol_in_copymodule:n {#2}
3579   \stex_debug:nn{renamedec1}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
3580   \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
3581   \str_if_empty:NTF \l_stex_renamedec1_name_str {
3582     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3583       \l_stex_get_symbol_uri_str
3584     } }
3585   } {
3586     \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex_r
3587     \stex_debug:nn{renamedec1}{@~\l_stex_current_module_str ? \l_stex_renamedec1_name_str}
3588     \prop_set_eq:cc {l_stex_symdecl_
3589       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3590     _prop
3591     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
3592     \seq_set_eq:cc {l_stex_symdecl_
3593       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3594     _notations
3595     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
3596     \prop_put:cnx {l_stex_symdecl_
3597       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3598     _prop
3599     }{ name }{ \l_stex_renamedec1_name_str }
3600     \prop_put:cnx {l_stex_symdecl_
3601       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3602     _prop
3603     }{ module }{ \l_stex_current_module_str }
3604     \exp_args:NNx \seq_put_left:Nn \l__stex_features_copymodule_fields_seq {
3605       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3606     }
3607     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3608       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3609     } }

```

```

3610 }
3611 }
3612 %\NewDocumentCommand \notation_in_copymodules: { 0{} m } {
3613 % \_stex_notation_args:n { #1 }
3614 % \tl_clear:N \l_stex_symdecl_definiens_tl
3615 % \stex_get_symbol_in_copymodule:n { #2 }
3616 % \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
3617 % % todo
3618 %}
3619 \stex_deactivate_macro:Nn \assign {copymodules}
3620 \stex_deactivate_macro:Nn \renamedekl {copymodules}
3621 \stex_deactivate_macro:Nn \donotcopy {copymodules}
3622
3623
3624 \seq_new:N \l_stex_implicit_morphisms_seq
3625 \NewDocumentCommand \implicitmorphism { 0{} m m }{
3626 \stex_import_module_uri:nn { #1 } { #2 }
3627 \stex_debug:nn{implicits}{
3628 Implicit~morphism:~
3629 \l_stex_module_ns_str ? \l__stex_features_name_str
3630 }
3631 \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
3632 \l_stex_module_ns_str ? \l__stex_features_name_str
3633 }{
3634 \msg_error:nnn{stex}{error/conflictingmodules}{
3635 \l_stex_module_ns_str ? \l__stex_features_name_str
3636 }
3637 }
3638
3639 % TODO
3640
3641
3642
3643 \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
3644 \l_stex_module_ns_str ? \l__stex_features_name_str
3645 }
3646 }
3647

```

## 32.2 The feature environment

structural@feature

```

3648
3649 \NewDocumentEnvironment{structural@feature}{ m m m }{
3650 \stex_if_in_module:F {
3651 \msg_set:nnn{stex}{error/nomodule}{
3652 Structural~Feature~has~to~occur~in~a~module:\\
3653 Feature~#2~of~type~#1\\
3654 In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
3655 }
3656 \msg_error:nn{stex}{error/nomodule}
3657 }
3658

```

```

3659 \str_set:Nx \l_stex_module_name_str {
3660   \prop_item:Nn \l_stex_current_module_prop
3661     { name } / #2 - feature
3662 }
3663
3664 \str_set:Nx \l_stex_module_ns_str {
3665   \prop_item:Nn \l_stex_current_module_prop
3666     { ns }
3667 }
3668
3669
3670 \str_clear:N \l_tmpa_str
3671 \seq_clear:N \l_tmpa_seq
3672 \tl_clear:N \l_tmpa_tl
3673 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
3674   origname = #2,
3675   name      = \l_stex_module_name_str ,
3676   ns        = \l_stex_module_ns_str ,
3677   imports   = \exp_not:o { \l_tmpa_seq } ,
3678   constants = \exp_not:o { \l_tmpa_seq } ,
3679   content   = \exp_not:o { \l_tmpa_tl } ,
3680   file      = \exp_not:o { \g_stex_currentfile_seq } ,
3681   lang      = \l_stex_module_lang_str ,
3682   sig       = \l_tmpa_str ,
3683   meta      = \l_tmpa_str ,
3684   feature   = #1 ,
3685 }
3686
3687 \stex_if_smsmode:F {
3688   \begin{stex_annotate_env}{ feature:#1 }{}
3689   \stex_annotate_invisible:nnn{header}{}{ #3 }
3690 }
3691 }{
3692   \str_set:Nx \l_tmpa_str {
3693     c_stex_feature_
3694     \prop_item:Nn \l_stex_current_module_prop { ns } ?
3695     \prop_item:Nn \l_stex_current_module_prop { name }
3696     _prop
3697   }
3698   \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
3699   \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
3700   \stex_if_smsmode:TF {
3701     \exp_args:Nx \stex_add_to_sms:n {
3702       \prop_gset_from_keyval:cn {
3703         c_stex_feature_
3704         \prop_item:Nn \l_stex_current_module_prop { ns } ?
3705         \prop_item:Nn \l_stex_current_module_prop { name }
3706         _prop
3707       } {
3708         origname = #2,
3709         name      = \prop_item:cn { \l_tmpa_str } { name } ,
3710         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
3711         imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
3712         constants = \prop_item:cn { \l_tmpa_str } { constants } ,

```

```

3713         content = \prop_item:cn { \l_tmpa_str } { content } ,
3714         file     = \prop_item:cn { \l_tmpa_str } { file } ,
3715         lang     = \prop_item:cn { \l_tmpa_str } { lang } ,
3716         sig      = \prop_item:cn { \l_tmpa_str } { sig } ,
3717         meta     = \prop_item:cn { \l_tmpa_str } { meta } ,
3718         feature  = \prop_item:cn { \l_tmpa_str } { feature }
3719     }
3720 }
3721 } {
3722     \end{stex_annotate_env}
3723 }
3724 }
3725

```

## 32.3 Features

structure

```

3726
3727 \prop_new:N \l_stex_all_structures_prop
3728
3729 \keys_define:nn { stex / features / structure } {
3730     name .str_set_x:N = \l__stex_features_structure_name_str ,
3731 }
3732
3733 \cs_new_protected:Nn \__stex_features_structure_args:n {
3734     \str_clear:N \l__stex_features_structure_name_str
3735     \keys_set:nn { stex / features / structure } { #1 }
3736 }
3737
3738 %\stex_new_feature:nnnn { structure } { 0{ } m } {
3739 % \__stex_features_structure_args:n { ##1 }
3740 % \str_if_empty:NT \l__stex_features_structure_name_str {
3741 %     \str_set:Nx \l__stex_features_structure_name_str { ##2 }
3742 % }
3743 %} {
3744 %
3745 %}
3746
3747 \NewDocumentEnvironment{mathstructure}{0{ } m }{
3748     \__stex_features_structure_args:n { #1 }
3749     \str_if_empty:NT \l__stex_features_structure_name_str {
3750         \str_set:Nx \l__stex_features_structure_name_str { #2 }
3751     }
3752     \exp_args:Nnnx
3753     \begin{structural@feature}{ structure }
3754         { \l__stex_features_structure_name_str }{}
3755         \seq_clear:N \l_tmpa_seq
3756         \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
3757     \stex_smsmode_do:
3758 }{
3759     \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
3760     \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
3761     \str_set:Nx \l_tmpa_str {

```

```

3762     \prop_item:Nn \l_stex_current_module_prop { ns } ?
3763     \prop_item:Nn \l_stex_current_module_prop { name }
3764   }
3765   \seq_map_inline:Nn \l_tmpa_seq {
3766     \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
3767   }
3768   \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
3769   \exp_args:Nnx
3770   \AddToHookNext { env / mathstructure / after }{
3771     \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}}{
3772       \_stex_term_math_oms:nnnn { \l_tmpa_str }{}{0}{}
3773     }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
3774     \STEXexport {
3775       \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3776       {\prop_item:Nn \l_stex_current_module_prop { origname }}
3777       {\l_tmpa_str}
3778       \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3779       {#2}{\l_tmpa_str}
3780     %   \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3781     %     \prop_item:Nn \l_stex_current_module_prop { origname },
3782     %     \l_tmpa_str
3783     %   }
3784     %   \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3785     %     #2,\l_tmpa_str
3786     %   }
3787     %   \tl_set:cx { #2 } {
3788     %     \stex_invoke_structure:n { \l_tmpa_str }
3789   }
3790 }
3791
3792 \end{structural@feature}
3793 % \g_stex_last_feature_prop
3794 }

```

\instantiate

```

3795 \seq_new:N \l__stex_features_structure_field_seq
3796 \str_new:N \l__stex_features_structure_field_str
3797 \str_new:N \l__stex_features_structure_def_tl
3798 \prop_new:N \l__stex_features_structure_prop
3799 \NewDocumentCommand \instantiate { m O{} m }{
3800   \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
3801   \prop_set_eq:Nc \l__stex_features_structure_prop {
3802     c_stex_feature_\l_tmpa_str _prop
3803   }
3804   \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
3805   \seq_map_inline:Nn \l__stex_features_structure_field_seq {
3806     \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
3807     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3808       \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
3809       \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
3810         {!} \l_tmpa_tl
3811       \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3812         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
3813         \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl

```

```

3814     \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3815   }{
3816     \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
3817     \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3818     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
3819       \l_tmpa_tl
3820     \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3821       \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
3822       \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3823     }{
3824       \tl_clear:N \l_tmpb_tl
3825     }
3826   }
3827 }{
3828   \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
3829   \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3830     \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
3831     \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
3832     \tl_clear:N \l_tmpa_tl
3833   }{
3834     % TODO throw error
3835   }
3836 }
3837 % \l_tmpa_str: name
3838 % \l_tmpa_tl: definiens
3839 % \l_tmpb_tl: notation
3840 \tl_if_empty:NT \l__stex_features_structure_field_str {
3841   % TODO throw error
3842 }
3843 \str_clear:N \l_tmpb_str
3844
3845 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3846 \seq_map_inline:Nn \l_tmpa_seq {
3847   \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
3848   \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
3849   \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
3850     \seq_map_break:n {
3851       \str_set:Nn \l_tmpb_str { ####1 }
3852     }
3853   }
3854 }
3855 \prop_get:cnN { l_stex_symdecl_ \l_tmpb_str _prop } {args}
3856   \l_tmpb_str
3857
3858 \tl_if_empty:NNTF \l_tmpb_tl {
3859   \tl_if_empty:NF \l_tmpa_tl {
3860     \exp_args:Nx \use:n {
3861       \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fe
3862     }
3863   }
3864 }{
3865   \tl_if_empty:NNTF \l_tmpa_tl {
3866     \exp_args:Nx \use:n {
3867       \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\

```

```

3868     }
3869
3870   }{
3871     \exp_args:Nx \use:n {
3872       \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
3873       \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
3874     }
3875   }
3876 }
3877 % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
3878 % \prop_item:Nn \l_stex_current_module_prop {name} ?
3879 % #3/\l__stex_features_structure_field_str
3880 % \par
3881 % \expandafter\present\csname
3882 %   l_stex_symdecl_
3883 %   \prop_item:Nn \l_stex_current_module_prop {ns} ?
3884 %   \prop_item:Nn \l_stex_current_module_prop {name} ?
3885 %   #3/\l__stex_features_structure_field_str
3886 %   _prop
3887 % \endcsname
3888 }
3889
3890 \tl_clear:N \l__stex_features_structure_def_tl
3891
3892 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3893 \seq_map_inline:Nnn \l_tmpa_seq {
3894   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3895   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3896   \exp_args:Nx \use:n {
3897     \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
3898
3899     }
3900   }
3901
3902   \prop_if_exist:cF {
3903     l_stex_symdecl_
3904     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3905     \prop_item:Nn \l_stex_current_module_prop {name} ?
3906     #3/\l_tmpa_str
3907     _prop
3908   }{
3909     \prop_get:cnN { l_stex_symdecl_ ##1 _prop } {args}
3910     \l_tmpb_str
3911     \exp_args:Nx \use:n {
3912       \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
3913     }
3914   }
3915 }
3916
3917 \symdecl*[type={\STEXsymbol{module-type}}{
3918   \_stex_term_math_oms:nnnn {
3919     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3920     \prop_item:Nn \l__stex_features_structure_prop {name}
3921   }-}{0}{-}

```

```

3922   }}]{#3}
3923
3924   % TODO: -> sms file
3925
3926   \tl_set:cx{ #3 }{
3927     \stex_invoke_structure:nnn {
3928       \prop_item:Nn \l_stex_current_module_prop {ns} ?
3929       \prop_item:Nn \l_stex_current_module_prop {name} ? #3
3930     } {
3931       \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3932       \prop_item:Nn \l__stex_features_structure_prop {name}
3933     }
3934   }
3935   \stex_smsmode_do:
3936 }

```

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

\stex\_invoke\_structure:nnn

```

3937 % #1: URI of the instance
3938 % #2: URI of the instantiated module
3939 \cs_new_protected:Nn \stex_invoke_structure:nnn {
3940   \tl_if_empty:nTF{ #3 }{
3941     \prop_set_eq:Nc \l__stex_features_structure_prop {
3942       c_stex_feature_ #2 _prop
3943     }
3944     \tl_clear:N \l_tmpa_tl
3945     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3946     \seq_map_inline:Nn \l_tmpa_seq {
3947       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3948       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3949       \cs_if_exist:cT {
3950         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
3951       }{
3952         \tl_if_empty:NF \l_tmpa_tl {
3953           \tl_put_right:Nn \l_tmpa_tl {,}
3954         }
3955         \tl_put_right:Nx \l_tmpa_tl {
3956           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
3957         }
3958       }
3959     }
3960     \exp_args:No \mathstrut \l_tmpa_tl
3961   }{
3962     \stex_invoke_symbol:n{#1/#3}
3963   }
3964 }

```

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

3965 </package>



## Chapter 33

# STEX -Statements Implementation

```
3966 <*package>
3967
3968 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3969
3970 <@@=stex_statements>
    Warnings and error messages
3971
\titleemph
3972 \def\titleemph#1{\textbf{#1}}
    (End definition for \titleemph. This function is documented on page ??.)
```

### 33.1 Definitions

```
definiendum
3973 \keys_define:nn {stex / definiendum }{
3974   post      .tl_set:N      = \l__stex_statements_definiendum_post_tl,
3975   root      .str_set_x:N   = \l__stex_statements_definiendum_root_str,
3976   gfa       .str_set_x:N   = \l__stex_statements_definiendum_gfa_str
3977 }
3978 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
3979   \str_clear:N \l__stex_statements_definiendum_root_str
3980   \tl_clear:N \l__stex_statements_definiendum_post_tl
3981   \str_clear:N \l__stex_statements_definiendum_gfa_str
3982   \keys_set:nn { stex / definiendum }{ #1 }
3983 }
3984 \NewDocumentCommand \definiendum { 0{} m m } {
3985   \__stex_statements_definiendum_args:n { #1 }
3986   \stex_get_symbol:n { #2 }
3987   \stex_ref_new_sym_target:n \l__stex_get_symbol_uri_str
3988   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
3989     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
3990       \tl_set:Nn \l_tmpa_tl { #3 }

```

```

3991   } {
3992     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
3993     \tl_set:Nn \l_tmpa_tl {
3994       \l__stex_statements_definiendum_root_str\l__stex_statements_definiendum_post_tl
3995     }
3996   }
3997 } {
3998   \tl_set:Nn \l_tmpa_tl { #3 }
3999 }
4000
4001 % TODO root
4002 \rustex_if:TF {
4003   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
4004 } {
4005   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
4006 }
4007 }
4008 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

**definame**

```

4009
4010 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
4011
4012 \NewDocumentCommand \definame { 0{ } m } {
4013   \__stex_statements_definiendum_args:n { #1 }
4014   % TODO: root
4015   \stex_get_symbol:n { #2 }
4016   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4017   \str_set:Nx \l_tmpa_str {
4018     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4019   }
4020   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4021   \rustex_if:TF {
4022     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4023       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4024     }
4025   } {
4026     \defemph@uri {
4027       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4028     } { \l_stex_get_symbol_uri_str }
4029   }
4030 }
4031 \stex_deactivate_macro:Nn \definame {definition~environments}
4032
4033 \NewDocumentCommand \Definame { 0{ } m } {
4034   \__stex_statements_definiendum_args:n { #1 }
4035   \stex_get_symbol:n { #2 }
4036   \str_set:Nx \l_tmpa_str {
4037     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4038   }
4039   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4040   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str

```

```

4041 \rustex_if:TF {
4042   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4043     \l_tmpa_str\l__stex_statements_definiendum_post_tl
4044   }
4045 } {
4046   \defemph@uri {
4047     \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
4048   } { \l_stex_get_symbol_uri_str }
4049 }
4050 }
4051 \stex_deactivate_macro:Nn \Definame {definition-environments}
4052
4053 \NewDocumentCommand \Symname { 0{ } m }{
4054   \stex_symname_args:n { #1 }
4055   \stex_get_symbol:n { #2 }
4056   \str_set:Nx \l_tmpa_str {
4057     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4058   }
4059   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4060   \let\compemph_uri_prev:\compemph@uri
4061   \let\compemph@uri\symrefemph@uri
4062   \exp_args:NNx \use:nn
4063   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }![
4064     \exp_after:wN \stex_capitalize:n \l_tmpa_str
4065     \l_stex_symname_post_str
4066   ] }
4067   \let\compemph@uri\compemph_uri_prev:
4068 }

```

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

#### sdefinition

```

4069
4070 \keys_define:nn {stex / sdefinition }{
4071   type      .str_set_x:N = \sdefinitiontype,
4072   id        .str_set_x:N = \sdefinitionid,
4073   name      .str_set_x:N = \sdefinitionname,
4074   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4075   title     .tl_set:N     = \sdefinitiontitle
4076 }
4077 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
4078   \str_clear:N \sdefinitiontype
4079   \str_clear:N \sdefinitionid
4080   \str_clear:N \sdefinitionname
4081   \clist_clear:N \l__stex_statements_sdefinition_for_clist
4082   \tl_clear:N \sdefinitiontitle
4083   \keys_set:nn { stex / sdefinition }{ #1 }
4084 }
4085
4086 \NewDocumentEnvironment{sdefinition}{0{}}{
4087   \__stex_statements_sdefinition_args:n{ #1 }
4088   \stex_reactivate_macro:N \definiendum
4089   \stex_reactivate_macro:N \definame
4090   \stex_reactivate_macro:N \Definame

```

```

4091 \stex_if_smsmode:F{
4092   \seq_clear:N \l_tmpa_seq
4093   \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4094     \str_if_eq:nnF{ ##1 }{}{
4095       \stex_get_symbol:n { ##1 }
4096       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4097         \l_stex_get_symbol_uri_str
4098       }
4099     }
4100   }
4101   \exp_args:Nnnx
4102   \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {,}}
4103   \str_if_empty:NF \sdefinitiontype {
4104     \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{}
4105   }
4106   \clist_set:No \l_tmpa_clist \sdefinitiontype
4107   \tl_clear:N \l_tmpa_tl
4108   \clist_map_inline:Nn \l_tmpa_clist {
4109     \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
4110       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
4111     }
4112   }
4113   \tl_if_empty:NTF \l_tmpa_tl {
4114     \__stex_statements_sdefinition_start:
4115   }{
4116     \l_tmpa_tl
4117   }
4118 }
4119 \stex_ref_new_doc_target:n \sdefinitionid
4120 \stex_smsmode_do:
4121 }{
4122   \str_if_empty:NF \sdefinitionname { \symdecl*{\sdefinitionname} }
4123   \stex_if_smsmode:F {
4124     \clist_set:No \l_tmpa_clist \sdefinitiontype
4125     \tl_clear:N \l_tmpa_tl
4126     \clist_map_inline:Nn \l_tmpa_clist {
4127       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
4128         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
4129       }
4130     }
4131     \tl_if_empty:NTF \l_tmpa_tl {
4132       \__stex_statements_sdefinition_end:
4133     }{
4134       \l_tmpa_tl
4135     }
4136     \end{stex_annotate_env}
4137   }
4138 }

```

\stexpatchdefinition

```

4139 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
4140   \par\noindent\titllemph{Definition\tl_if_empty:NF \sdefinitiontitle {
4141     ~(\sdefinitiontitle)
4142   }~}

```

```

4143 }
4144 \cs_new_protected:Nn \__stex_statements_sdefinition_end: { \par \medskip }
4145
4146 \newcommand\stexpatchdefinition[3] [] {
4147   \str_set:Nx \l_tmpa_str{ #1 }
4148   \str_if_empty:NTF \l_tmpa_str {
4149     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
4150     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
4151   }{
4152     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
4153     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
4154   }
4155 }

```

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

\inlinedef inline:

```

4156 \keys_define:nn {stex / inlinedef }{
4157   type      .str_set_x:N = \sdefinitiontype,
4158   id        .str_set_x:N = \sdefinitionid,
4159   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4160   name      .str_set_x:N = \sdefinitionname
4161 }
4162 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
4163   \str_clear:N \sdefinitiontype
4164   \str_clear:N \sdefinitionid
4165   \str_clear:N \sdefinitionname
4166   \clist_clear:N \l__stex_statements_sdefinition_for_clist
4167   \keys_set:nn { stex / inlinedef }{ #1 }
4168 }
4169 \NewDocumentCommand \inlinedef { 0{} m } {
4170   \begingroup
4171   \__stex_statements_inlinedef_args:n{ #1 }
4172   \stex_ref_new_doc_target:n \sdefinitionid
4173   \stex_reactivate_macro:N \definiendum
4174   \stex_reactivate_macro:N \definame
4175   \stex_reactivate_macro:N \Definame
4176   \stex_if_smsmode:TF{
4177     \str_if_empty:NF \sdefinitionname { \symdecl*{\sdefinitionname} }
4178   }{
4179     \seq_clear:N \l_tmpa_seq
4180     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4181       \str_if_eq:nnF{ ##1 }{ }{
4182         \stex_get_symbol:n { ##1 }
4183         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4184           \l_stex_get_symbol_uri_str
4185         }
4186       }
4187     }
4188     \exp_args:Nnx
4189     \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
4190       \str_if_empty:NF \sdefinitiontype {
4191         \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{ }
4192       }
4193     }

```

```

4193     #2
4194     \str_if_empty:NF \sdefinitionname { \symdecl*{\sdefinitionname} }
4195   }
4196 }
4197 \endgroup
4198 \stex_smsmode_do:
4199 }

```

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

## 33.2 Assertions

sassertion

```

4200
4201 \keys_define:nn {stex / sassertion }{
4202   type      .str_set_x:N = \sassertiontype,
4203   id        .str_set_x:N = \sassertionid,
4204   title     .tl_set:N    = \sassertiontitle ,
4205   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4206   name      .str_set_x:N = \sassertionname
4207 }
4208 \cs_new_protected:Nn \l__stex_statements_sassertion_args:n {
4209   \str_clear:N \sassertiontype
4210   \str_clear:N \sassertionid
4211   \str_clear:N \sassertionname
4212   \clist_clear:N \l__stex_statements_sassertion_for_clist
4213   \tl_clear:N \sassertiontitle
4214   \keys_set:nn { stex / sassertion }{ #1 }
4215 }
4216
4217 %\tl_new:N \g__stex_statements_aftergroup_tl
4218
4219 \NewDocumentEnvironment{sassertion}{0{}}{
4220   \l__stex_statements_sassertion_args:n{ #1 }
4221   \stex_if_smsmode:F {
4222     \seq_clear:N \l_tmpa_seq
4223     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4224       \str_if_eq:nnF{ ##1 }{}{
4225         \stex_get_symbol:n { ##1 }
4226         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4227           \l_stex_get_symbol_uri_str
4228         }
4229       }
4230     }
4231     \exp_args:Nnnx
4232     \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}
4233     \str_if_empty:NF \sassertiontype {
4234       \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
4235     }
4236     \clist_set:Nn \l_tmpa_clist \sassertiontype
4237     \tl_clear:N \l_tmpa_tl
4238     \clist_map_inline:Nn \l_tmpa_clist {
4239       \tl_if_exist:cT {\l__stex_statements_sassertion_##1_start:}{

```

```

4240         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
4241     }
4242 }
4243 \tl_if_empty:NTF \l_tmpa_tl {
4244     \__stex_statements_sassertion_start:
4245 }{
4246     \l_tmpa_tl
4247 }
4248 }
4249 \stex_ref_new_doc_target:n \sassertionid
4250 \stex_smsmode_do:
4251 }{
4252     \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
4253     \stex_if_smsmode:F {
4254         \clist_set:Nn \l_tmpa_clist \sassertiontype
4255         \tl_clear:N \l_tmpa_tl
4256         \clist_map_inline:Nn \l_tmpa_clist {
4257             \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
4258                 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
4259             }
4260         }
4261         \tl_if_empty:NTF \l_tmpa_tl {
4262             \__stex_statements_sassertion_end:
4263         }{
4264             \l_tmpa_tl
4265         }
4266         \end{stex_annotate_env}
4267     }
4268 }

```

`\stexpatchassertion`

```

4269
4270 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
4271     \par\noindent\titllemph{Assertion~\tl_if_empty:NF \sassertiontitle {
4272         (\sassertiontitle)
4273     }~}
4274 }
4275 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
4276
4277 \newcommand\stexpatchassertion[3] [] {
4278     \str_set:Nx \l_tmpa_str{ #1 }
4279     \str_if_empty:NTF \l_tmpa_str {
4280         \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
4281         \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
4282     }{
4283         \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
4284         \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
4285     }
4286 }

```

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

`\inlineass inline:`

```

4287 \keys_define:nn {stex / inlineass }{

```

```

4288 type      .str_set_x:N = \sassertiontype,
4289 id        .str_set_x:N = \sassertionid,
4290 for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4291 name      .str_set_x:N = \sassertionname
4292 }
4293 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
4294   \str_clear:N \sassertiontype
4295   \str_clear:N \sassertionid
4296   \str_clear:N \sassertionname
4297   \clist_clear:N \l__stex_statements_sassertion_for_clist
4298   \keys_set:nn { stex / inlineass }{ #1 }
4299 }
4300 \NewDocumentCommand \inlineass { 0{} m } {
4301   \beginngroup
4302     \__stex_statements_inlineass_args:n{ #1 }
4303     \stex_ref_new_doc_target:n \sassertionid
4304     \stex_if_smsmode:TF{
4305       \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
4306     }{
4307       \seq_clear:N \l_tmpa_seq
4308       \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4309         \str_if_eq:nnF{ ##1 }{ }{
4310           \stex_get_symbol:n { ##1 }
4311           \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4312             \l_stex_get_symbol_uri_str
4313           }
4314         }
4315       }
4316       \exp_args:Nnx
4317       \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{
4318         \str_if_empty:NF \sassertiontype {
4319           \stex_annotate_invisible:nnn{type}{\sassertiontype}{ }
4320         }
4321         #2
4322         \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
4323       }
4324     }
4325     \endgroup
4326     \stex_smsmode_do:
4327   }

```

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

### 33.3 Examples

`sexample`

```

4328
4329 \keys_define:nn {stex / sexample }{
4330   type      .str_set_x:N = \exampletype,
4331   id        .str_set_x:N = \sexampleid,
4332   title     .tl_set:N     = \sexampltitle,
4333   for       .clist_set:N = \l__stex_statements_sexample_for_clist,
4334 }

```



```

4335 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
4336   \str_clear:N \sexamplotype
4337   \str_clear:N \sexampleid
4338   \tl_clear:N \sexampletile
4339   \clist_clear:N \l__stex_statements_sexample_for_clist
4340   \keys_set:nn { stex / sexample }{ #1 }
4341 }
4342
4343 \NewDocumentEnvironment{sexample}{0{}}{
4344   \__stex_statements_sexample_args:n{ #1 }
4345   \stex_if_smsmode:F {
4346     \seq_clear:N \l_tmpa_seq
4347     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4348       \str_if_eq:nnF{ ##1 }{{}{
4349         \stex_get_symbol:n { ##1 }
4350         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4351           \l_stex_get_symbol_uri_str
4352         }
4353       }
4354     }
4355     \exp_args:Nnnx
4356     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
4357     \str_if_empty:NF \sexamplotype {
4358       \stex_annotate_invisible:nnn{type}{\sexamplotype}{}
4359     }
4360     \clist_set:No \l_tmpa_clist \sexamplotype
4361     \tl_clear:N \l_tmpa_tl
4362     \clist_map_inline:Nn \l_tmpa_clist {
4363       \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
4364         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
4365       }
4366     }
4367     \tl_if_empty:NTF \l_tmpa_tl {
4368       \__stex_statements_sexample_start:
4369     }{
4370       \l_tmpa_tl
4371     }
4372   }
4373   \stex_ref_new_doc_target:n \sexampleid
4374   \stex_smsmode_do:
4375 }{
4376   \str_if_empty:NF \sexamplename { \symdecl*{\sexamplename} }
4377   \stex_if_smsmode:F {
4378     \clist_set:No \l_tmpa_clist \sexamplotype
4379     \tl_clear:N \l_tmpa_tl
4380     \clist_map_inline:Nn \l_tmpa_clist {
4381       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
4382         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
4383       }
4384     }
4385     \tl_if_empty:NTF \l_tmpa_tl {
4386       \__stex_statements_sexample_end:
4387     }{
4388       \l_tmpa_tl

```

```

4389     }
4390     \end{stex_annotate_env}
4391   }
4392 }

```

\stexpatchexample

```

4393
4394 \cs_new_protected:Nn \__stex_statements_sexample_start: {
4395   \par\noindent\titllemph{Example~\tl_if_empty:NF \sexampltitle {
4396     (\sexampltitle)
4397   }~}
4398 }
4399 \cs_new_protected:Nn \__stex_statements_sexample_end: { \par\medskip}
4400
4401 \newcommand\stexpatchexample[3] [] {
4402   \str_set:Nx \l_tmpa_str{ #1 }
4403   \str_if_empty:NTF \l_tmpa_str {
4404     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
4405     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
4406   }{
4407     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
4408     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
4409   }
4410 }

```

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

\inlineex inline:

```

4411 \keys_define:nn {stex / inlineex }{
4412   type      .str_set_x:N = \sexampltype,
4413   id        .str_set_x:N = \sexampleid,
4414   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
4415   name      .str_set_x:N = \sexamplname
4416 }
4417 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
4418   \str_clear:N \sexampltype
4419   \str_clear:N \sexampleid
4420   \str_clear:N \sexamplname
4421   \clist_clear:N \l__stex_statements_sexample_for_clist
4422   \keys_set:nn { stex / inlineex }{ #1 }
4423 }
4424 \NewDocumentCommand \inlineex { 0{} m } {
4425   \begingroup
4426   \__stex_statements_inlineex_args:n{ #1 }
4427   \stex_ref_new_doc_target:n \sexampleid
4428   \stex_if_smsmode:TF{
4429     \str_if_empty:NF \sexamplname { \symdecl*{\examplename} }
4430   }{
4431     \seq_clear:N \l_tmpa_seq
4432     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4433       \str_if_eq:nnF{ ##1 }{}{
4434         \stex_get_symbol:n { ##1 }
4435         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4436           \l_stex_get_symbol_uri_str

```

```

4437     }
4438   }
4439 }
4440 \exp_args:Nnx
4441 \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpa_seq {,}}{
4442   \str_if_empty:NF \sexamplotype {
4443     \stex_annotate_invisible:nnn{type}{\sexamplotype}{ }
4444   }
4445   #2
4446   \str_if_empty:NF \sexamplename { \symdecl*{\sexamplename} }
4447 }
4448 }
4449 \endgroup
4450 \stex_smsmode_do:
4451 }

```

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

## 33.4 Logical Paragraphs

`sparagraph`

```

4452 \keys_define:nn { stex / sparagraph } {
4453   id      .str_set:N = \sparagraphid ,
4454   title   .tl_set:N  = \l_stex_sparagraph_title_tl ,
4455   type    .str_set:N  = \sparagraphtype ,
4456   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
4457   from    .tl_set:N    = \sparagraphfrom ,
4458   to      .tl_set:N    = \sparagraphto ,
4459   start   .tl_set:N    = \l_stex_sparagraph_start_tl ,
4460   name    .str_set:N    = \sparagraphname
4461 }
4462
4463 \cs_new_protected:Nn \stex_sparagraph_args:n {
4464   \tl_clear:N \l_stex_sparagraph_title_tl
4465   \tl_clear:N \sparagraphfrom
4466   \tl_clear:N \sparagraphto
4467   \tl_clear:N \l_stex_sparagraph_start_tl
4468   \str_clear:N \sparagraphid
4469   \str_clear:N \sparagraphtype
4470   \clist_clear:N \l__stex_statements_sparagraph_for_clist
4471   \str_clear:N \sparagraphname
4472   \keys_set:nn { stex / sparagraph } { #1 }
4473 }
4474 \newif\if@in@omtext\@in@omtextfalse
4475
4476 \NewDocumentEnvironment {sparagraph} { 0{ } } {
4477   \stex_sparagraph_args:n { #1 }
4478   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4479     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
4480   }{
4481     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
4482   }
4483   \@in@omtexttrue

```

```

4484 \stex_if_smsmode:F {
4485   \seq_clear:N \l_tmpa_seq
4486   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
4487     \str_if_eq:nnF{ ##1 }{}{
4488       \stex_get_symbol:n { ##1 }
4489       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4490         \l_stex_get_symbol_uri_str
4491       }
4492     }
4493   }
4494   \exp_args:Nnnx
4495   \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
4496   \str_if_empty:NF \sparagraphtype {
4497     \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}
4498   }
4499   \str_if_empty:NF \sparagraphfrom {
4500     \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
4501   }
4502   \str_if_empty:NF \sparagraphto {
4503     \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
4504   }
4505   \clist_set:No \l_tmpa_clist \sparagraphtype
4506   \tl_clear:N \l_tmpa_tl
4507   \clist_map_inline:Nn \sparagraphtype {
4508     \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
4509       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
4510     }
4511   }
4512   \tl_if_empty:NTF \l_tmpa_tl {
4513     \__stex_statements_sparagraph_start:
4514   }{
4515     \l_tmpa_tl
4516   }
4517 }
4518 \stex_ref_new_doc_target:n \sparagraphid
4519 \stex_smsmode_do:
4520 \ignorespacesandpars
4521 }{
4522   \stex_if_smsmode:F {
4523     \clist_set:No \l_tmpa_clist \sparagraphtype
4524     \tl_clear:N \l_tmpa_tl
4525     \clist_map_inline:Nn \l_tmpa_clist {
4526       \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
4527         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
4528       }
4529     }
4530     \str_if_empty:NF \sparagraphname { \symdecl*{\sparagraphname} }
4531     \tl_if_empty:NTF \l_tmpa_tl {
4532       \__stex_statements_sparagraph_end:
4533     }{
4534       \l_tmpa_tl
4535     }
4536     \end{stex_annotate_env}
4537   }

```

```

4538 }
\stexpatchparagraph

```

```

4539
4540 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
4541   \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4542     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
4543       \titleemph{\l_stex_sparagraph_title_tl}:~
4544     }
4545   }{
4546     \titleemph{\l_stex_sparagraph_start_tl}~
4547   }
4548 }
4549 \cs_new_protected:Nn \__stex_statements_sparagraph_end: { \par\medskip}
4550
4551 \newcommand\stexpatchparagraph[3] [] {
4552   \str_set:Nx \l_tmpa_str{ #1 }
4553   \str_if_empty:NTF \l_tmpa_str {
4554     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
4555     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
4556   }{
4557     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
4558     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
4559   }
4560 }
4561
4562 \keys_define:nn { stex / inlinepara } {
4563   id      .str_set_x:N = \sparagraphid ,
4564   type    .str_set_x:N = \sparagraphtype ,
4565   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
4566   from    .tl_set:N    = \sparagraphfrom ,
4567   to      .tl_set:N    = \sparagraphto ,
4568   name    .str_set:N   = \sparagraphname
4569 }
4570 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
4571   \tl_clear:N \sparagraphfrom
4572   \tl_clear:N \sparagraphto
4573   \str_clear:N \sparagraphid
4574   \str_clear:N \sparagraphtype
4575   \clist_clear:N \l__stex_statements_sparagraph_for_clist
4576   \str_clear:N \sparagraphname
4577   \keys_set:nn { stex / inlinepara } { #1 }
4578 }
4579 \NewDocumentCommand \inlinepara { 0{} m } {
4580   \begingroup
4581   \__stex_statements_inlinepara_args:n{ #1 }
4582   \stex_ref_new_doc_target:n \sparagraphid
4583   \stex_if_smsmode:TF{
4584     \str_if_empty:NF \sparagraphname { \symdecl*\sparagraphname } }
4585   }{
4586     \seq_clear:N \l_tmpa_seq
4587     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
4588       \str_if_eq:nnF{ ##1 }{{}{
4589         \stex_get_symbol:n { ##1 }

```

```

4590     \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4591         \l_stex_get_symbol_uri_str
4592     }
4593 }
4594 }
4595 \exp_args:Nnx
4596 \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
4597     \str_if_empty:NF \sparagraphtype {
4598         \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}
4599     }
4600     \str_if_empty:NF \sparagraphfrom {
4601         \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
4602     }
4603     \str_if_empty:NF \sparagraphto {
4604         \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
4605     }
4606     #2
4607     \str_if_empty:NF \sparagraphname { \symdecl*{\sparagraphname} }
4608 }
4609 }
4610 \endgroup
4611 \stex_smsmode_do:
4612 }
4613

```

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

`symboldoc`

```

4614 \NewDocumentEnvironment{symboldoc}{m}{
4615     \seq_set_split:Nnn \l_tmpa_seq , { #1 }
4616     \seq_clear:N \l_tmpb_seq
4617     \seq_map_inline:Nn \l_tmpa_seq {
4618         \str_if_eq:nnF{ ##1 }{}{
4619             \stex_get_symbol:n { ##1 }
4620             \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
4621                 \l_stex_get_symbol_uri_str
4622             }
4623         }
4624     }
4625     \par
4626     \exp_args:Nnnx
4627     \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
4628 }{
4629     \end{stex_annotate_env}
4630 }
4631 \</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>

```
4632 <*package>
4633 <@@=stex_sproof>
4634
4635 %%%%%%%%%%% sproof.dtx %%%%%%%%%%%
4636
```

### 34.2 Proofs

We first define some keys for the proof environment.

```
4637 \keys_define:nn { stex / spf } {
4638   id          .str_set:N = \l__stex_sproof_spf_id_str,
4639   display     .tl_set:N  = \l__stex_sproof_spf_display_tl,
4640   for         .tl_set:N  = \l__stex_sproof_spf_for_tl ,
4641   from       .tl_set:N  = \l__stex_sproof_spf_from_tl ,
4642   proofend    .tl_set:N  = \l__stex_sproof_spf_proofend_tl,
4643   type        .tl_set:N  = \l__stex_sproof_spf_type_tl,
4644   title       .tl_set:N  = \l__stex_sproof_spf_title_tl,
4645   continues   .tl_set:N  = \l__stex_sproof_spf_continues_tl,
4646   functions   .tl_set:N  = \l__stex_sproof_spf_functions_tl,
4647   method      .tl_set:N  = \l__stex_sproof_spf_method_tl
4648 }
4649 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
4650   \str_clear:N \l__stex_sproof_spf_id_str
4651   \tl_clear:N \l__stex_sproof_spf_display_tl
4652   \tl_clear:N \l__stex_sproof_spf_for_tl
4653   \tl_clear:N \l__stex_sproof_spf_from_tl
4654   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
4655   \tl_clear:N \l__stex_sproof_spf_type_tl
4656   \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

```

4657 \tl_clear:N \l__stex_sproof_spf_continues_tl
4658 \tl_clear:N \l__stex_sproof_spf_functions_tl
4659 \tl_clear:N \l__stex_sproof_spf_method_tl
4660 \keys_set:nn { stex / spf }{ #1 }
4661 }

```

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

```

4662 \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.

```

4663 \newcount\count_ten
4664 \newenvironment{pst@with@label}[1]{
4665   \edef\pst@label{#1}
4666   \advance\count_ten by 1\relax
4667   \count_ten=1
4668 }{
4669   \advance\count_ten by -1\relax
4670 }

```

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

```

4671 \def\the@pst@label{
4672   \pst@make@label\pst@label{\number\count_ten}\l__stex_sproof_pstlabel_postfix_tl
4673 }

```

(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.

```

4674 \keys_define:nn { stex / pstlabel }{
4675   prefix      .tl_set:N   = \l__stex_sproof_pstlabel_prefix_tl,
4676   delimiter   .tl_set:N   = \l__stex_sproof_pstlabel_delimiter_tl,
4677   postfix     .tl_set:N   = \l__stex_sproof_pstlabel_postfix_tl
4678 }
4679 \cs_new_protected:Nn \__stex_sproof_pstlabel_args:n {

```

---

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



```

4680 \tl_set:Nn \l__stex_sproof_pstlabel_prefix_tl {P}
4681 \tl_set:Nn \l__stex_sproof_pstlabel_delimiter_tl {.}
4682 \tl_clear:N \l__stex_sproof_pstlabel_postfix_tl
4683 }
4684 \__stex_sproof_pstlabel_args:n {}
4685 \newcommand\setpstlabelstyle[1]{
4686   \__stex_sproof_pstlabel_args:n {#1}
4687 }
4688 \newcommand\setpstlabelstyledefault{%
4689   \__stex_sproof_pstlabel_args:n{prefix=P,delimiter=.,postfix={}}
4690 }

```

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

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

```

4691 \ExplSyntaxOff
4692 \def\pst@make@label@long#1#2{\@for\@I:=#1\do{\expandafter\expandafter\expandafter\@I\csname
4693 \def\pst@make@label@angles#1#2{\ensuremath{\@for\@I:=#1\do{\rangle}}#2}
4694 \def\pst@make@label@short#1#2{#2}
4695 \def\pst@make@label@empty#1#2{}
4696 \ExplSyntaxOn
4697 \def\pstlabelstyle#1{%
4698   \def\pst@make@label{\use:c{pst@make@label@#1}}%
4699 }%
4700 \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.

```

4701 \def\next@pst@label{%
4702   \global\advance\count\count10 by 1%
4703 }%

```

(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

```

4704 \def\sproof@box{
4705   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
4706 }
4707 \def\spf@proofend{\sproof@box}
4708 \def\sproofend{
4709   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
4710     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
4711   }
4712 }
4713 \def\sProofEndSymbol#1{\def\sproof@box{#1}}

```

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

**spf@\*@kw**

```

4714 \def\spf@proofsketch@kw{Proof Sketch}
4715 \def\spf@proof@kw{Proof}
4716 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

4717 \AddToHook{begindocument}{
4718   \ltx@ifpackageloaded{babel}{
4719     \makeatletter
4720     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4721     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4722       \input{sproof-ngerman.ldf}
4723     }
4724     \clist_if_in:NnT \l_tmpa_clist {finnish}{
4725       \input{sproof-finnish.ldf}
4726     }
4727     \clist_if_in:NnT \l_tmpa_clist {french}{
4728       \input{sproof-french.ldf}
4729     }
4730     \clist_if_in:NnT \l_tmpa_clist {russian}{
4731       \input{sproof-russian.ldf}
4732     }
4733     \makeatother
4734   }{}
4735 }

```

`spfsketch`

```

4736 \newcommand\spfsketch[2][]{
4737   \__stex_sproof_spf_args:n{#1}
4738   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4739     \titleemph{
4740       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
4741         \spf@proofsketch@kw
4742       }{
4743         \l__stex_sproof_spf_type_tl
4744       }
4745     }:
4746   }
4747   {~#2}
4748   %\sref@label@id{this \ifx\spf@type\@empty\spf@proofsketch@kw\else\spf@type\fi}
4749   \sproofend
4750 }

```

(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>

```

4751 \newenvironment{spfeq}[2][]{
4752   \__stex_sproof_spf_args:n{#1}
4753   %\sref@target
4754   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4755     \titleemph{
4756       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
4757         \spf@proof@kw
4758       }{

```

<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

```

4759     \l__stex_sproof_spf_type_tl
4760   }
4761   }:
4762 }
4763 {-#2}
4764 \begin{displaymath}\begin{array}{rcll}
4765 }{
4766   \end{array}\end{displaymath}
4767 }

```

(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.

```

4768 \newenvironment{spf@proof}[2] []{
4769   \l__stex_sproof_spf_args:n{#1}
4770   %\sref@target
4771   \count_ten=10
4772   \par\noindent
4773   \tl_if_eq:NNTF \l__stex_sproof_spf_display_tl\spf@flow{
4774     \titleemph{
4775       \tl_if_empty:NNTF \l__stex_sproof_spf_type_tl {
4776         \spf@proof@kw
4777       }{
4778         \l__stex_sproof_spf_type_tl
4779       }
4780     }:
4781   }
4782   {-#2}
4783   %\sref@label{id{this \ifx\spf@type\@empty\spf@proof@kw\else\spf@type\fi}
4784   \def\pst@label{}
4785   \newcount\pst@count% initialize the labeling mechanism
4786   \begin{description}\begin{pst@with@label}{\l__stex_sproof_pstlabel_prefix_tl}
4787   }{
4788     \end{pst@with@label}\end{description}
4789   }
4790   \newenvironment{sproof}[2] []{\begin{spf@proof}[#1]{#2}}{\sproofend\end{spf@proof}}
4791   \newenvironment{sProof}[2] []{\begin{spf@proof}[#1]{#2}}{\end{spf@proof}}

```

**\spfidea**

```

4792 \newcommand\spfidea[2] []{
4793   \l__stex_sproof_spf_args:n{#1}
4794   \titleemph{
4795     \tl_if_empty:NNTF \l__stex_sproof_spf_type_tl {Proof~Idea}{
4796       \l__stex_sproof_spf_type_tl
4797     }:
4798   }-#2
4799   \sproofend
4800 }

```

(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
4801 \newenvironment{spfstep}[1][]{
4802   \_stex_sproof_spf_args:n{#1}
4803   \@in@omtexttrue
4804   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4805     \item[\the@pst@label]
4806   }
4807   \tl_if_empty:NF \l__stex_sproof_spf_title_tl {
4808     {(\titleemph{\l__stex_sproof_spf_title_tl})\enspace}
4809   }
4810   %\sref@label{id{\pst@label}
4811   \ignorespacesandpars
4812 }{
4813   \next@pst@label\ignorespacesandpars
4814 }

```

**sproofcomment**

```

4815 \newenvironment{sproofcomment}[1][]{
4816   \_stex_sproof_spf_args:n{#1}
4817   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4818     \item[\the@pst@label]
4819   }
4820 }{
4821   \next@pst@label
4822 }

```

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.

```

4823 \newenvironment{subproof}[2][]{
4824   \_stex_sproof_spf_args:n{#1}
4825   \def\@test{#2}
4826   \ifx\@test\empty\else
4827     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4828       \item[\the@pst@label]
4829     }{#2}
4830   \fi
4831   \begin{pst@with@label}{\pst@label,\number\count_ten}
4832 }{
4833   \end{pst@with@label}\next@pst@label
4834 }

```

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

```

4835 \newenvironment{spfcases}[2][]{
4836   \def\@test{#1}
4837   \ifx\@test\empty
4838     \begin{subproof}[method=by-cases]{#2}

```

---

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

```

4839 \else
4840   \begin{subproof}[#1,method=by-cases]{#2}
4841 \fi
4842 }{
4843   \end{subproof}
4844 }

```

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

```

4845 \newenvironment{spfcase}[2] [] {
4846   \__stex_sproof_spf_args:n{#1}
4847   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4848     \item[\the@pst@label]
4849   }
4850   \def\@test{#2}
4851   \ifx\@test\@empty
4852   \else
4853     {\titleemph{#2}:~}
4854   \fi
4855   \begin{pst@with@label}{\pst@label,\number\count_ten}
4856 }{
4857   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4858     \sproofend
4859   }
4860   \end{pst@with@label}
4861   \next@pst@label
4862 }

```

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

```

4863 \newcommand\spfcasesketch[3] [] {
4864   \__stex_sproof_spf_args:n{#1}
4865   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4866     \item[\the@pst@label]
4867   }
4868   \def\@test{#2}
4869   \ifx\@test\@empty
4870   \else
4871     {\titleemph{#2}:~}
4872   \fi#3
4873   \next@pst@label
4874 }%

```

### 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.

```

4875 \keys_define:nn { stex / just }{
4876   id      .str_set:x:N = \l__stex_sproof_just_id_str,
4877   method  .tl_set:N    = \l__stex_sproof_just_method_tl,
4878   premises .tl_set:N    = \l__stex_sproof_just_premises_tl,
4879   args    .tl_set:N    = \l__stex_sproof_just_args_tl
4880 }

```

EdN:17

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`

```
4881 \newenvironment{justification}[1] [] {}{}
```

`\premise`

```
4882 \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.

```
4883 \newcommand\justarg[2] [] {#2}
```

```
4884 \end{package}
```

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

Some auxiliary code, and clean up to be executed at the end of the package.

---

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

## Chapter 35

# STEX -Others Implementation

```
4885 <*package>
4886
4887 %%%%%%%%%%% others.dtx %%%%%%%%%%%
4888
4889 <@@=stex_others>
      Warnings and error messages
4890 % None

\MSC Math subject classifier

4891 \NewDocumentCommand \MSC {m} {
4892   % TODO
4893 }

(End definition for \MSC. This function is documented on page 21.)
      Patching tikzinput, if loaded
4894 \@ifpackageloaded{tikzinput}{
4895   \RequirePackage{stex-tikzinput}
4896 }{}
4897 </package>
```

## Chapter 36

# STEX -Metatheory Implementation

```
4898 \*package>
4899 \@@=stex_modules>
4900
4901 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
4902
4903 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
4904 \begingroup
4905 \stex_module_setup:nn{
4906   ns=\c_stex_metatheory_ns_str,
4907   meta=NONE
4908 }{Metatheory}
4909 \stex_reactivate_macro:N \symdecl
4910 \stex_reactivate_macro:N \notation
4911 \stex_reactivate_macro:N \symdef
4912 \ExplSyntaxOff
4913 \csname stex_suppress_html:n\endcsname{
4914   % is-a (a:A, a \in A, a is an A, etc.)
4915   \symdecl[args=ai]{isa}
4916   \notation[typed]{isa}{#1 \comp{:} #2}{##1 \comp, ##2}
4917   \notation[in]{isa}{#1 \comp\in #2}{##1 \comp, ##2}
4918   \notation[pred]{isa}{#2\comp(#1 \comp)}{##1 \comp, ##2}
4919
4920   % bind (\forall, \Pi, \lambda etc.)
4921   \symdecl[args=Bi]{bind}
4922   \notation[forall]{bind}{\comp\forall #1.\;#2}{##1 \comp, ##2}
4923   \notation[\Pi]{bind}{\comp\prod_{#1}#2}{##1 \comp, ##2}
4924   \notation[deffun]{bind}{\comp( #1 \comp{ }\;\to\; ) #2}{##1 \comp, ##2}
4925
4926   % dummy variable
4927   \symdecl{dummyvar}
4928   \notation[underscore]{dummyvar}{\comp\_}
4929   \notation[dot]{dummyvar}{\comp\cdot}
4930   \notation[dash]{dummyvar}{\comp{\rm --}}
4931
4932   %fromto (function space, Hom-set, implication etc.)
```



```

4933 \symdecl[args=ai]{fromto}
4934 \notation[xarrow]{fromto}{#1 \comp\to #2}{##1 \comp\times ##2}
4935 \notation[arrow]{fromto}{#1 \comp\to #2}{##1 \comp\to ##2}
4936
4937 % mapto (lambda etc.)
4938 %\symdecl[args=Bi]{mapto}
4939 %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
4940 %\notation[lambda]{mapto}{\comp\lambda #1 \comp.; #2}{#1 \comp, #2}
4941 %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp.; #2}{#1 \comp, #2}
4942
4943 % function/operator application
4944 \symdecl[args=ia]{apply}
4945 \notation[prec=0;0x\infpres,parens]{apply}{#1 \comp( #2 \comp)}{##1 \comp, ##2}
4946 \notation[prec=0;0x\infpres,lambda]{apply}{#1 \; #2 }{##1 \; ; ##2}
4947
4948 % ‘type’ of all collections (sets, classes, types, kinds)
4949 \symdecl{collection}
4950 \notation[U]{collection}{\comp{\mathcal{U}}}
4951 \notation[set]{collection}{\comp{\textsf{Set}}}
4952
4953 % sequences
4954 \symdecl[args=1]{seqtype}
4955 \notation[kleene]{seqtype}{#1^{\comp\ast}}
4956
4957 \symdef[args=2,li,prec=nobrackets]{sequence-index}{#1}_{#2}
4958 \notation[ui,prec=nobrackets]{sequence-index}{#1}^{#2}
4959
4960 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{,\ellipses}}{##1\comp,##2}
4961 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{,\ellipses},#2}{##1\comp,##2}
4962 \symdef[args=aui,prec=nobrackets]{aseqfromtovia}{#1\comp{,\ellipses},#2\comp{,\ellipses},#3}
4963
4964 % letin (‘let’, local definitions, variable substitution)
4965 \symdecl[args=bii]{letin}
4966 \notation[let]{letin}{\comp{\rm let}}{#1\comp{=}#2;\comp{\rm in}}{#3}
4967 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
4968 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
4969
4970 % structures
4971 \symdecl*[args=1]{module-type}
4972 \notation{module-type}{\mathtt{MOD} #1}
4973 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
4974 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{##1 \comp, ##2}
4975
4976 }
4977 \ExplSyntaxOn
4978 \stex_add_to_current_module:n{
4979   \let\appa\apply
4980   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
4981   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
4982   \def\livar{\csname sequence-index\endcsname[li]}
4983   \def\uivar{\csname sequence-index\endcsname[ui]}
4984   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
4985   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
4986   \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}

```

```
4987 }  
4988 \__stex_modules_end_module:  
4989 \endgroup  
4990 </package>
```

## Chapter 37

# Tikzinput Implementation

```
4991 <*package>
4992
4993 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
4994
4995 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
4996 \RequirePackage{l3keys2e}
4997
4998 \keys_define:nn { tikzinput } {
4999   image .bool_set:N = \c_tikzinput_image_bool,
5000   image .default:n = false ,
5001   unknown .code:n = {}
5002 }
5003
5004 \ProcessKeysOptions { tikzinput }
5005
5006 \bool_if:NTF \c_tikzinput_image_bool {
5007   \RequirePackage{graphicx}
5008
5009   \providecommand\usetikzlibrary[]{}
5010   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
5011 }{
5012   \RequirePackage{tikz}
5013   \RequirePackage{standalone}
5014
5015   \newcommand \tikzinput [2] [] {
5016     \setkeys{Gin}{#1}
5017     \ifx \Gin@ewidth \Gin@exclamation
5018       \ifx \Gin@eheight \Gin@exclamation
5019         \input { #2 }
5020       \else
5021         \resizebox{!}{ \Gin@eheight }{
5022           \input { #2 }
5023         }
5024       \fi
5025     \else
5026       \ifx \Gin@eheight \Gin@exclamation
5027         \resizebox{ \Gin@ewidth }{!}{
5028           \input { #2 }
```

```

5029     }
5030     \else
5031         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
5032             \input { #2 }
5033         }
5034     \fi
5035 \fi
5036 }
5037 }
5038
5039 \newcommand \ctikzinput [2] [] {
5040     \begin{center}
5041         \tikzinput [1] {#2}
5042     \end{center}
5043 }
5044
5045 \@ifpackageloaded{stex}{
5046     \RequirePackage{stex-tikzinput}
5047 }{}
5048
5049 </package>
5050 <*stex>
5051 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
5052 \RequirePackage{stex}
5053 \RequirePackage{tikzinput}
5054
5055 \newcommand\mhtikzinput [2] [] {%
5056     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
5057     \stex_in_repository:nn\Gin@mhrepos{
5058         \tikzinput[#1]{\mhpath{##1}{#2}}
5059     }
5060 }
5061 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [1] {#2}\end{center}}
5062 </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.

```
5063 \*cls)
5064 \<@@=document_structure>
5065 \ProvidesExplClass{document-structure}{2022/02/10}{3.0}{Modular Document Structure Class}
5066 \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`

```
5067 \keys_define:nn{ document-structure / pkg }{
5068   class      .str_set_x:N = \c_document_structure_class_str,
5069   minimal    .bool_set:N = \c_document_structure_minimal_bool,
5070   report     .code:n      = {
5071     \ClassWarning{document-structure}{the option 'report' is deprecated, use 'class=report',
5072     \str_set:Nn \c_document_structure_class_str {report}
5073   },
5074   book       .code:n      = {
5075     \ClassWarning{document-structure}{the option 'book' is deprecated, use 'class=book', ins
5076     \str_set:Nn \c_document_structure_class_str {book}
5077   },
5078   bookpart   .code:n      = {
5079     \ClassWarning{document-structure}{the option 'bookpart' is deprecated, use 'class=book,t
5080     \str_set:Nn \c_document_structure_class_str {book}
5081     \str_set:Nn \c_document_structure_topsect_str {chapter}
5082   },
```

```

5083 docopt      .str_set_x:N = \c_document_structure_docopt_str,
5084 unknown     .code:n      = {
5085   \PassOptionsToPackage{ \CurrentOption }{ document-structure }
5086 }
5087 }
5088 \ProcessKeysOptions{ document-structure / pkg }
5089 \str_if_empty:NT \c_document_structure_class_str {
5090   \str_set:Nn \c_document_structure_class_str {article}
5091 }
5092 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
5093   {\c_document_structure_class_str}
5094

```

### 38.3 Beefing up the document environment

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

```

5095 \RequirePackage{document-structure}
5096 \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>

```

5097 \keys_define:nn { document-structure / document }{
5098   id .str_set_x:N = \c_document_structure_document_id_str
5099 }
5100 \let\__document_structure_orig_document=\document
5101 \renewcommand{\document}[1][]{
5102   \keys_set:nn{ document-structure / document }{ #1 }
5103   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
5104   \__document_structure_orig_document
5105 }

```

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

```

5106 }
5107 \</cls>

```

### 38.4 Implementation: document-structure Package

```

5108 \<*package>
5109 \ProvidesExplPackage{document-structure}{2022/02/10}{3.0}{Modular Document Structure}
5110 \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

```

5111
5112 \keys_define:nn{ document-structure / pkg }{
5113   class      .str_set_x:N = \c_document_structure_class_str,
5114   topsect    .str_set_x:N = \c_document_structure_topsect_str,
5115   % showignores .bool_set:N = \c_document_structure_showignores_bool,
5116 }
5117 \ProcessKeysOptions{ document-structure / pkg }
5118 \str_if_empty:NT \c_document_structure_class_str {
5119   \str_set:Nn \c_document_structure_class_str {article}
5120 }
5121 \str_if_empty:NT \c_document_structure_topsect_str {
5122   \str_set:Nn \c_document_structure_topsect_str {section}
5123 }

```

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

```

5124 \RequirePackage{xspace}
5125 \RequirePackage{comment}
5126 \AddToHook{begindocument}{
5127   \ltx@ifpackageloaded{babel}{
5128     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5129     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5130       \makeatletter\input{omdoc-ngerman.ldf}\makeatother
5131     }
5132   }{}
5133 }

```

`\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.

```

5134 \int_new:N \l_document_structure_section_level_int
5135 \str_case:VnF \c_document_structure_topsect_str {
5136   {part}}{
5137     \int_set:Nn \l_document_structure_section_level_int {0}
5138   }
5139   {chapter}}{
5140     \int_set:Nn \l_document_structure_section_level_int {1}
5141   }
5142 }{
5143   \str_case:VnF \c_document_structure_class_str {
5144     {book}}{
5145       \int_set:Nn \l_document_structure_section_level_int {0}
5146     }
5147     {report}}{
5148       \int_set:Nn \l_document_structure_section_level_int {0}
5149     }
5150   }{
5151     \int_set:Nn \l_document_structure_section_level_int {2}
5152   }
5153 }

```

## 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

```
5154 \def\current@section@level{document}%
5155 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
5156 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
5157 \cs_new_protected:Npn \skipomgroup {
5158   \ifcase\l_document_structure_section_level_int
5159   \or\stepcounter{part}
5160   \or\stepcounter{chapter}
5161   \or\stepcounter{section}
5162   \or\stepcounter{subsection}
5163   \or\stepcounter{subsubsection}
5164   \or\stepcounter{paragraph}
5165   \or\stepcounter{subparagraph}
5166   \fi
5167 }
```

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

`blindomgroup`

```
5168 \newcommand\at@begin@blindomgroup[1]{%
5169 \newenvironment{blindomgroup}
5170 {
5171   \int_incr:N\l_document_structure_section_level_int
5172   \at@begin@blindomgroup\l_document_structure_section_level_int
5173 }{}}
```

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

```
5174 \newcommand\omgroup@nonum[2]{
5175   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
5176   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
5177 }
```

*(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.

```
5178 \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.



```

5179 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
5180   \@nameuse{#1}{#2}
5181 }{
5182   \cs_if_exist:NTF\rdfmata@sectioning{
5183     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
5184   }{
5185     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
5186   }
5187 }
5188 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
5189 }

```

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

omgroup

```

5190 \keys_define:nn { document-structure / omgroup }{
5191   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
5192   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
5193   creators     .clist_set:N = \l__document_structure_omgroup_creators_clist,
5194   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
5195   srccite      .tl_set:N   = \l__document_structure_omgroup_srccite_tl,
5196   type         .tl_set:N   = \l__document_structure_omgroup_type_tl,
5197   short        .tl_set:N   = \l__document_structure_omgroup_short_tl,
5198   display      .tl_set:N   = \l__document_structure_omgroup_display_tl,
5199   intro        .tl_set:N   = \l__document_structure_omgroup_intro_tl,
5200   loadmodules  .bool_set:N = \l__document_structure_omgroup_loadmodules_bool
5201 }
5202 \cs_new_protected:Nn \__document_structure_omgroup_args:n {
5203   \str_clear:N \l__document_structure_omgroup_id_str
5204   \str_clear:N \l__document_structure_omgroup_date_str
5205   \clist_clear:N \l__document_structure_omgroup_creators_clist
5206   \clist_clear:N \l__document_structure_omgroup_contributors_clist
5207   \tl_clear:N \l__document_structure_omgroup_srccite_tl
5208   \tl_clear:N \l__document_structure_omgroup_type_tl
5209   \tl_clear:N \l__document_structure_omgroup_short_tl
5210   \tl_clear:N \l__document_structure_omgroup_display_tl
5211   \tl_clear:N \l__document_structure_omgroup_intro_tl
5212   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
5213   \keys_set:nn { document-structure / omgroup } { #1 }
5214 }

```

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.

```

5215 \newif\if@mainmatter\@mainmattertrue
5216 \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.

```

5217 \keys_define:nn { document-structure / sectioning }{
5218   name .str_set_x:N = \l__document_structure_sect_name_str ,
5219   ref   .str_set_x:N = \l__document_structure_sect_ref_str   ,
5220   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
5221   clear .default:n = {true} ,
5222   num   .bool_set:N = \l__document_structure_sect_num_bool   ,

```

```

5223   num      .default:n      = {true}
5224 }
5225 \cs_new_protected:Nn \__document_structure_sect_args:n {
5226   \str_clear:N \l__document_structure_sect_name_str
5227   \str_clear:N \l__document_structure_sect_ref_str
5228   \bool_set_false:N \l__document_structure_sect_clear_bool
5229   \bool_set_false:N \l__document_structure_sect_num_bool
5230   \keys_set:nn { document-structure / sectioning } { #1 }
5231 }
5232 \newcommand\omdoc@sectioning[3][]{
5233   \__document_structure_sect_args:n {#1}
5234   \let\omdoc@sect@name\l__document_structure_sect_name_str
5235   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
5236   \if@mainmatter% numbering not overridden by frontmatter, etc.
5237     \bool_if:NTF \l__document_structure_sect_num_bool {
5238       \omgroup@num{#2}{#3}
5239     }{
5240       \omgroup@nonum{#2}{#3}
5241     }
5242     \def\current@section@level{\omdoc@sect@name}
5243   \else
5244     \omgroup@nonum{#2}{#3}
5245   \fi
5246 }% 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.

```

5247 \newcommand\omgroup@redefine@addtocontents[1]{%
5248 %\edef\__document_structureimport{#1}%
5249 %\@for\@I:=\__document_structureimport\do{%
5250 %\edef\@path{\csname module@\@I @path\endcsname}%
5251 %\@ifundefined{tf@toc}\relax%
5252 %   {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
5253 %\ifx\hyper@anchor\undefined% hyperref.sty loaded?
5254 %\def\addcontentsline##1##2##3{%
5255 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
5256 %\else% hyperref.sty not loaded
5257 %\def\addcontentsline##1##2##3{%
5258 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
5259 %\fi
5260 }% hypreref.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.

```

5261 \newenvironment{omgroup}[2][]{% keys, title
5262 {
5263   \__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.

```

5264 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
5265   \omgroup@redefine@addtocontents{
5266     \@ifundefined{module@id}\used@modules%

```

```

5267     %{\ifundefined{module@\module@id @path}{\used@modules}\module@id}
5268   }
5269 }

```

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

```

5270 \int_incr:N\l_document_structure_section_level_int
5271 \ifcase\l_document_structure_section_level_int
5272   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
5273   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
5274   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
5275   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
5276   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
5277   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
5278   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
5279 \fi
5280 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
5281 \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
5282 }% for customization
5283 {}

```

and finally, we localize the sections

```

5284 \newcommand\omdoc@part@kw{Part}
5285 \newcommand\omdoc@chapter@kw{Chapter}
5286 \newcommand\omdoc@section@kw{Section}
5287 \newcommand\omdoc@subsection@kw{Subsection}
5288 \newcommand\omdoc@subsubsection@kw{Subsubsection}
5289 \newcommand\omdoc@paragraph@kw{paragraph}
5290 \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`

```

5291 \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).

```

5292 \cs_if_exist:NTF\frontmatter{
5293   \let\__document_structure_orig_frontmatter\frontmatter
5294   \let\frontmatter\relax
5295 }{
5296   \tl_set:Nn\__document_structure_orig_frontmatter{
5297     \clearpage
5298     \@mainmatterfalse
5299     \pagenumbering{roman}
5300   }
5301 }

```

Using these, we can now define the `frontmatter` and `backmatter` environments

```

5312 \newenvironment{frontmatter}{
5313   \__document_structure_orig_frontmatter
5314 }{
5315   \cs_if_exist:NTF\mainmatter{
5316     \mainmatter
5317   }{
5318     \clearpage
5319     \@mainmattertrue
5320     \pagenumbering{arabic}
5321   }
5322 }

```

```

5323 \newenvironment{backmatter}{
5324   \__document_structure_orig_backmatter
5325 }{
5326   \cs_if_exist:NTF\mainmatter{
5327     \mainmatter
5328   }{
5329     \clearpage
5330     \@mainmattertrue
5331     \pagenumbering{arabic}
5332   }
5333 }

```

5334 \@mainmattertrue\pagenumbering{arabic}

```

5335 \def \c__document_structure_document_str{document}
5336 \newcommand\afterprematurestop{}
5337 \def\prematurestop@endomgroup{
5338   \unless\ifx\@currentvir\c__document_structure_document_str
5339     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
5340     \expandafter\prematurestop@endomgroup
5341   \fi
5342 }

```

```

5343 \providecommand\prematurestop{
5344   \message{Stopping~sTeX~processing~prematurely}
5345   \prematurestop@endomgroup
5346   \afterprematurestop
5347   \end{document}
5348 }

```

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

## 38.8 Global Variables

`\setSGvar` set a global variable

```

5349 \RequirePackage{etoolbox}
5350 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

`\useSGvar` use a global variable

```

5351 \newrobustcmd\useSGvar[1]{%
5352   \@ifundefined{sTeX@Gvar@#1}
5353   {\PackageError{document-structure}
5354     {The sTeX Global variable #1 is undefined}
5355     {set it with \protect\setSGvar}}
5356   \@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.

```

5357 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
5358   \@ifundefined{sTeX@Gvar@#1}
5359   {\PackageError{document-structure}
5360     {The sTeX Global variable #1 is undefined}
5361     {set it with \protect\setSGvar}}
5362   {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}

```

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

## Chapter 39

# NotesSlides – Implementation

### 39.1 Class and Package Options

We define some Package Options and switches for the `notesslides` class and activate them by passing them on to `beamer.cls` and `omdoc.cls` and the `notesslides` package. We pass the `nontheorem` option to the `statements` package when we are not in notes mode, since the `beamer` package has its own (overlay-aware) theorem environments.

```
5363 \*cls)
5364 \@@=notesslides)
5365 \ProvidesExplClass{notesslides}{2022/02/10}{3.0}{notesslides Class}
5366 \RequirePackage{l3keys2e,expl-keystr-compatible}
5367
5368 \keys_define:nn{notesslides / cls}{
5369   class .code:n = {
5370     \PassOptionsToClass{\CurrentOption}{document-structure}
5371     \str_if_eq:nnT{#1}{book}{
5372       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5373     }
5374     \str_if_eq:nnT{#1}{report}{
5375       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5376     }
5377   },
5378   notes .bool_set:N = \c__notesslides_notes_bool ,
5379   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
5380   unknown .code:n = {
5381     \PassOptionsToClass{\CurrentOption}{document-structure}
5382     \PassOptionsToClass{\CurrentOption}{beamer}
5383     \PassOptionsToPackage{\CurrentOption}{notesslides}
5384   }
5385 }
5386 \ProcessKeysOptions{ notesslides / cls }
5387 \bool_if:NTF \c__notesslides_notes_bool {
5388   \PassOptionsToPackage{notes=true}{notesslides}
5389 }{
5390   \PassOptionsToPackage{notes=false}{notesslides}
5391 }
5392 \</cls)
```

now we do the same for the notesslides package.

```

5393 <*package>
5394 \ProvidesExplPackage{notesslides}{2022/02/10}{3.0}{notesslides Package}
5395 \RequirePackage{l3keys2e,expl-keystr-compat}
5396
5397 \keys_define:nn{notesslides / pkg}{
5398   topsect          .str_set_x:N = \c__notesslides_topsect_str,
5399   defaulttopsect   .str_set_x:N = \c__notesslides_defaulttopsec_str,
5400   notes            .bool_set:N = \c__notesslides_notes_bool ,
5401   slides           .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
5402   sectocframes     .bool_set:N = \c__notesslides_sectocframes_bool ,
5403   frameimages      .bool_set:N = \c__notesslides_frameimages_bool ,
5404   fiboxed          .bool_set:N = \c__notesslides_fiboxed_bool ,
5405   noproblems       .bool_set:N = \c__notesslides_noproblems_bool,
5406   unknown          .code:n      = {
5407     \PassOptionsToClass{\CurrentOption}{stex}
5408     \PassOptionsToClass{\CurrentOption}{tikzinput}
5409   }
5410 }
5411 \ProcessKeysOptions{ notesslides / pkg }
5412 \newif\ifnotes
5413 \bool_if:NTF \c__notesslides_notes_bool {
5414   \notesttrue
5415 }{
5416   \notesfalse
5417 }
5418

```

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

```

5419 \str_if_empty:NTF \c__notesslides_topsect_str {
5420   \str_set_eq:NN \__notesslides_topsect \c__notesslides_defaulttopsec_str
5421 }{
5422   \str_set_eq:NN \__notesslides_topsect \c__notesslides_topsect_str
5423 }
5424 </package>

```

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

```

5425 <*cls>
5426 \bool_if:NTF \c__notesslides_notes_bool {
5427   \LoadClass{document-structure}
5428 }{
5429   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
5430   \newcounter{Item}
5431   \newcounter{paragraph}
5432   \newcounter{subparagraph}
5433   \newcounter{Hfootnote}
5434   \RequirePackage{document-structure}
5435 }

```

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

```

5436 \RequirePackage{notesslides}
5437 </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.

```

5438 \*package>
5439 \bool_if:NT \c__notesslides_notes_bool {
5440   \RequirePackage{a4wide}
5441   \RequirePackage{marginnote}
5442   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
5443   \RequirePackage{mdframed}
5444   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
5445   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
5446 }
5447 \RequirePackage{stex-tikzinput}
5448 \RequirePackage{etoolbox}
5449 \RequirePackage{amssymb}
5450 \RequirePackage{amsmath}
5451 \RequirePackage{comment}
5452 \RequirePackage{textcomp}
5453 \RequirePackage{url}
5454 \RequirePackage{graphicx}
5455 \RequirePackage{pgf}

```

## 39.2 Notes and Slides

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

```

5456 \bool_if:NT \c__notesslides_notes_bool {
5457   \renewcommand\usetheme[2][\]{\usepackage[#1]{beamernotestheme#2}}
5458 }

```

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

```

5459 \newcounter{slide}
5460 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
5461 \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.

```

5462 \bool_if:NTF \c__notesslides_notes_bool {
5463   \renewenvironment{note}{\ignorespaces}{\ignorespaces}{}
5464 }{
5465   \excludcomment{note}
5466 }

```

---

<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.

```
5467 \bool_if:NT \c__notesslides_notes_bool {
5468   \newlength{\slideframewidth}
5469   \setlength{\slideframewidth}{1.5pt}
```

**frame** We first define the keys.

```
5470 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
5471   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
5472     \bool_set_true:N #1
5473   }{
5474     \bool_set_false:N #1
5475   }
5476 }
5477 \keys_define:nn{notesslides / frame}{
5478   label .str_set_x:N = \l__notesslides_frame_label_str,
5479   allowframebreaks .code:n = {
5480     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
5481   },
5482   allowdisplaybreaks .code:n = {
5483     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
5484   },
5485   fragile .code:n = {
5486     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
5487   },
5488   shrink .code:n = {
5489     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
5490   },
5491   squeeze .code:n = {
5492     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
5493   },
5494   t .code:n = {
5495     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
5496   },
5497 }
5498 \cs_new_protected:Nn \__notesslides_frame_args:n {
5499   \str_clear:N \l__notesslides_frame_label_str
5500   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
5501   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
5502   \bool_set_true:N \l__notesslides_frame_fragile_bool
5503   \bool_set_true:N \l__notesslides_frame_shrink_bool
5504   \bool_set_true:N \l__notesslides_frame_squeeze_bool
5505   \bool_set_true:N \l__notesslides_frame_t_bool
5506   \keys_set:nn { notesslides / frame }{ #1 }
5507 }
```

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

```
5508 \renewenvironment{frame}[1][ ]{
5509   \__notesslides_frame_args:n{#1}
5510   \sffamily
5511   \stepcounter{slide}
5512   \def\@currentlabel{\theslide}
5513   \str_if_empty:NF \l__notesslides_frame_label_str {
5514     \label{\l__notesslides_frame_label_str}
```

5515 }

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

```

5516 \def\itemize@level{outer}
5517 \def\itemize@outer{outer}
5518 \def\itemize@inner{inner}
5519 \renewcommand\newpage{\addtocounter{framenum}{1}}
5520 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
5521 \renewenvironment{itemize}{
5522   \ifx\itemize@level\itemize@outer
5523     \def\itemize@label{$\rhd$}
5524   \fi
5525   \ifx\itemize@level\itemize@inner
5526     \def\itemize@label{$\scriptstyle\rhd$}
5527   \fi
5528   \begin{list}
5529     {\itemize@label}
5530     {\setlength{\labelsep}{.3em}
5531      \setlength{\labelwidth}{.5em}
5532      \setlength{\leftmargin}{1.5em}
5533     }
5534   \edef\itemize@level{\itemize@inner}
5535 }{
5536   \end{list}
5537 }

```

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

```

5538 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth]
5539 }{
5540   \medskip\miko@slidelabel\end{mdframed}
5541 }

```

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

`\frametitle`

```

5542 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
5543 }

```

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

EdN:21

`\pause` 21

```

5544 \bool_if:NT \c__notesslides_notes_bool {
5545   \newcommand\pause{}
5546 }

```

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

`nparagraph`

```

5547 \bool_if:NTF \c__notesslides_notes_bool {
5548   \newenvironment{nparagraph}[1][\begin{sparagraph}[#1]}\end{sparagraph}}
5549 }{
5550   \excludecomment{nparagraph}
5551 }

```

---

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

nomgroup

```
5552 \bool_if:NTF \c__notesslides_notes_bool {
5553   \newenvironment{nomgroup}[2] [] {\begin{omgroup}[#1]{#2}}{\end{omgroup}}
5554 }{
5555   \excludecomment{nomgroup}
5556 }
```

ndefinition

```
5557 \bool_if:NTF \c__notesslides_notes_bool {
5558   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}{\end{sdefinition}}
5559 }{
5560   \excludecomment{ndefinition}
5561 }
```

nassertion

```
5562 \bool_if:NTF \c__notesslides_notes_bool {
5563   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}{\end{sassertion}}
5564 }{
5565   \excludecomment{nassertion}
5566 }
```

nsproof

```
5567 \bool_if:NTF \c__notesslides_notes_bool {
5568   \newenvironment{nproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
5569 }{
5570   \excludecomment{nproof}
5571 }
```

nexample

```
5572 \bool_if:NTF \c__notesslides_notes_bool {
5573   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}{\end{sexample}}
5574 }{
5575   \excludecomment{nexample}
5576 }
```

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

```
5577 \def\inputref@preskip{\smallskip}
5578 \def\inputref@postskip{\medskip}
```

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

\inputref\*

```
5579 \let\orig@inputref\inputref
5580 \def\inputref{\@ifstar\ninputref\orig@inputref}
5581 \newcommand\ninputref[2] [] {
5582   \bool_if:NT \c__notesslides_notes_bool {
5583     \orig@inputref[#1]{#2}
5584   }
5585 }
```

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

## 39.3 Header and Footer Lines

Now, we set up the infrastructure for the footer line of the slides, we use boxes for the logos, so that they are only loaded once, that considerably speeds up processing.

**\setslidelogo** The default logo is the  $\text{\TeX}$  logo. Customization can be done by `\setslidelogo{<logo name>}`.

```

5586 \newlength{\slidelogoheight}
5587
5588 \bool_if:NTF \c__notesslides_notes_bool {
5589   \setlength{\slidelogoheight}{.4cm}
5590 }{
5591   \setlength{\slidelogoheight}{1cm}
5592 }
5593 \newsavebox{\slidelogo}
5594 \sbox{\slidelogo}{\TeX}
5595 \newrobustcmd{\setslidelogo}[1]{
5596   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
5597 }
```

(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.

```

5598 \def\source{Michael Kohlhase}% customize locally
5599 \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.

```

5600 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
5601 \newsavebox{\cclogo}
5602 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
5603 \newif\ifcchref\cchreffalse
5604 \AtBeginDocument{
5605   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
5606 }
5607 \def\licensing{
5608   \ifcchref
5609     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
5610   \else
5611     {\usebox{\cclogo}}
5612   \fi
5613 }
5614 \newrobustcmd{\setlicensing}[2][]{
5615   \def@url{#1}
5616   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
5617   \ifx@url@empty
5618     \def\licensing{{\usebox{\cclogo}}}
5619   \else
5620     \def\licensing{
```

```

5621     \ifcchref
5622     \href{#1}{\usebox{\cclogo}}
5623     \else
5624     {\usebox{\cclogo}}
5625     \fi
5626   }
5627 \fi
5628 }

```

(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>

```

5629 \newrobustcmd\miko@slidelabel{
5630   \vbox to \slidelogoheight{
5631     \vss\hbox to \slidewidth
5632     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
5633   }
5634 }

```

(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.

```

5635 \def\Gin@mhrepos{}
5636 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
5637 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
5638 \newrobustcmd\frameimage[2][]{
5639   \stepcounter{slide}
5640   \bool_if:NT \c__notesslides_frameimages_bool {
5641     \def\Gin@ewidth{}\setkeys{Gin}{#1}
5642     \bool_if:NF \c__notesslides_notes_bool { \vfill }
5643     \begin{center}
5644       \bool_if:NTF \c__notesslides_fiboxed_bool {
5645         \fbox{
5646           \ifx\Gin@ewidth\@empty
5647             \ifx\Gin@mhrepos\@empty
5648               \mhgraphics[width=\slidewidth,#1]{#2}
5649             \else
5650               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5651             \fi
5652           \else% Gin@ewidth empty
5653             \ifx\Gin@mhrepos\@empty
5654               \mhgraphics[#1]{#2}
5655             \else
5656               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5657             \fi
5658           \fi% Gin@ewidth empty
5659         }
5660       }{
5661         \ifx\Gin@ewidth\@empty

```

---

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

```

5662         \ifx\Gin@mhrepos\@empty
5663             \mhgraphics[width=\slidewidth,#1]{#2}
5664         \else
5665             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5666         \fi
5667         \ifx\Gin@mhrepos\@empty
5668             \mhgraphics[#1]{#2}
5669         \else
5670             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5671         \fi
5672     \fi% Gin@ewidth empty
5673 }
5674 \end{center}
5675 \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
5676 \bool_if:NF \c__notesslides_notes_bool { \vfill }
5677 }
5678 } % 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.

```

5679 \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

```

5680 \AddToHook{begindocument}{
5681     \definecolor{green}{rgb}{0,.5,0}
5682     \definecolor{purple}{cmk}{.3,1,0,.17}
5683 }

```

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`.

```

5684 % \def\STpresent#1{\textcolor{blue}{#1}}
5685 \def\defemph#1{\textcolor{magenta}{#1}}
5686 \def\symrefemph#1{\textcolor{cyan}{#1}}
5687 \def\compemph#1{\textcolor{blue}{#1}}
5688 \def\titleemph#1{\textcolor{blue}{#1}}
5689 \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.

```

5690 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
5691 \def\smalltextwarning{
5692     \pgfuseimage{miko@small@dbend}
5693     \xspace
5694 }
5695 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}

```

```

5696 \newrobustcmd\textwarning{
5697   \raisebox{-0.05cm}{\pgfuseimage{miko@dbend}}
5698   \xspace
5699 }
5700 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}
5701 \newrobustcmd\bigtextwarning{
5702   \raisebox{-0.05cm}{\pgfuseimage{miko@big@dbend}}
5703   \xspace
5704 }

```

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

```

5705 \newrobustcmd\putgraphicsat[3]{
5706   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
5707 }
5708 \newrobustcmd\putat[2]{
5709   \begin{picture}(0,0)\put(#1){#2}\end{picture}
5710 }

```

## 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`.

```

5711 \bool_if:NT \c__notesslides_sectocframes_bool {
5712   \str_if_eq:VnTF \__notesslidesstopsect{part}{
5713     \newcounter{chapter}\counterwithin*{section}{chapter}
5714   }{
5715     \str_if_eq:VnT\__notesslidesstopsect{chapter}{
5716       \newcounter{chapter}\counterwithin*{section}{chapter}
5717     }
5718   }
5719 }

```

`\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
5720 \def\part@prefix{}
5721 \@ifpackageloaded{document-structure}{\{
5722   \str_case:VnF \__notesslidesstopsect {
5723     {part}{
5724       \int_set:Nn \l_document_structure_section_level_int {0}
5725       \def\thesection{\arabic{chapter}.\arabic{section}}
5726       \def\part@prefix{\arabic{chapter}.}
5727     }
5728     {chapter}{
5729       \int_set:Nn \l_document_structure_section_level_int {1}
5730       \def\thesection{\arabic{chapter}.\arabic{section}}
5731       \def\part@prefix{\arabic{chapter}.}
5732     }
5733   }{
5734     \int_set:Nn \l_document_structure_section_level_int {2}
5735     \def\part@prefix{}

```

```

5736 }
5737 }
5738
5739 \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

```

5740 \renewenvironment{omgroup}[2][]{
5741   \__document_structure_omgroup_args:n { #1 }
5742   \int_incr:N \l_document_structure_section_level_int
5743   \bool_if:NT \c__notesslides_sectocframes_bool {
5744     \stepcounter{slide}
5745     \begin{frame}[noframenumbering]
5746       \vfill\Large\centering
5747       \red{
5748         \ifcase\l_document_structure_section_level_int\or
5749           \stepcounter{part}
5750           \def\__notesslideslabel{\omdoc@part@kw~\Roman{part}}
5751           \def\currentsectionlevel{\omdoc@part@kw}
5752         \or
5753           \stepcounter{chapter}
5754           \def\__notesslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
5755           \def\currentsectionlevel{\omdoc@chapter@kw}
5756         \or
5757           \stepcounter{section}
5758           \def\__notesslideslabel{\part@prefix\arabic{section}}
5759           \def\currentsectionlevel{\omdoc@section@kw}
5760         \or
5761           \stepcounter{subsection}
5762           \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
5763           \def\currentsectionlevel{\omdoc@subsection@kw}
5764         \or
5765           \stepcounter{subsubsection}
5766           \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
5767           \def\currentsectionlevel{\omdoc@subsubsection@kw}
5768         \or
5769           \stepcounter{paragraph}
5770           \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}.\arabic{paragraph}}
5771           \def\currentsectionlevel{\omdoc@paragraph@kw}
5772         \else
5773           \def\__notesslideslabel{}
5774           \def\currentsectionlevel{\omdoc@paragraph@kw}
5775         \fi% end ifcase
5776         \__notesslideslabel%\sref@label@id\__notesslideslabel
5777         \quad #2%
5778       }%
5779       \vfill%
5780     \end{frame}%
5781   }
5782   \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str%
5783 }{}

```



5784 }

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

```
5785 \def\inserttheorembodyfont{\normalfont}
5786 %\bool_if:NF \c__notesslides_notes_bool {
5787 % \defbeamertemplate{theorem begin}{miko}
5788 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
5789 % \ifx\inserttheoremaddition@empty\else\ (\inserttheoremaddition)\fi%
5790 % \inserttheorempunctuation\inserttheorembodyfont\hspace}
5791 % \defbeamertemplate{theorem end}{miko}{}
```

and we set it as the default one.

```
5792 % \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.

```
5793 % \expandafter\def\csname Parent2\endcsname{}
5794 %}
5795
5796 \AddToHook{begindocument}{% this does not work for some reasons
5797 \setbeamertemplate{theorems}[ams style]
5798 }
5799 \bool_if:NT \c__notesslides_notes_bool {
5800 \renewenvironment{columns}[1][{}]{%
5801 \par\noindent%
5802 \begin{minipage}%
5803 \slidewidth\centering\leavevmode%
5804 }{%
5805 \end{minipage}\par\noindent%
5806 }%
5807 \newsavebox\columnbox%
5808 \renewenvironment<>{column}[2][{}]{%
5809 \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
5810 }{%
5811 \end{minipage}\end{lrbox}\usebox\columnbox%
5812 }%
5813 }
5814 \bool_if:NTF \c__notesslides_noproblems_bool {
5815 \newenvironment{problems}{}{}
5816 }{
5817 \excludecomment{problems}
5818 }
```

## 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.

```
5819 \gdef\printexcursions{}
5820 \newcommand\excursionref[2]{% label, text
5821 \bool_if:NT \c__notesslides_notes_bool {
5822 \begin{sparagraph}[title=Excursion]
```

```

5823     #2 \sref[fallback=the appendix]{#1}.
5824 \end{sparagraph}
5825 }
5826 }
5827 \newcommand\activate@excursion[2][]{
5828 \gappto\printexcursions{\inputref[#1]{#2}}
5829 }
5830 \newcommand\excursion[4][]{% repos, label, path, text
5831 \bool_if:NT \c__notesslides_notes_bool {
5832 \activate@excursion[#1]{#3}\excursionref{#2}{#4}
5833 }
5834 }

```

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

\excursiongroup

```

5835 \keys_define:nn{notesslides / excursiongroup }{
5836 id .str_set_x:N = \l__notesslides_excursion_id_str,
5837 intro .tl_set:N = \l__notesslides_excursion_intro_tl,
5838 mhrepos .str_set_x:N = \l__notesslides_excursion_mhrepos_str
5839 }
5840 \cs_new_protected:Nn \__notesslides_excursion_args:n {
5841 \tl_clear:N \l__notesslides_excursion_intro_tl
5842 \str_clear:N \l__notesslides_excursion_id_str
5843 \str_clear:N \l__notesslides_excursion_mhrepos_str
5844 \keys_set:nn {notesslides / excursiongroup }{ #1 }
5845 }
5846 \newcommand\excursiongroup[1][]{
5847 \__notesslides_excursion_args:n{ #1 }
5848 \ifdefempty\printexcursions{}% only if there are excursions
5849 {\begin{note}
5850 \begin{omgroup}[#1]{Excursions}%
5851 \ifdefempty\l__notesslides_excursion_intro_tl{\{
5852 \inputref[\l__notesslides_excursion_mhrepos_str]{
5853 \l__notesslides_excursion_intro_tl
5854 }
5855 }
5856 \printexcursions%
5857 \end{omgroup}
5858 \end{note}}
5859 }
5860 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{\fi
5861 \</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.

```
5862 <*package>
5863 <@@=problems>
5864 \ProvidesExplPackage{problem}{2019/03/20}{1.3}{Semantic Markup for Problems}
5865 \RequirePackage{l3keys2e,expl-keystr-compatible}
5866
5867 \keys_define:nn { problem / pkg }{
5868   notes      .default:n   = { true },
5869   notes      .bool_set:N  = \c__problems_notes_bool,
5870   gnotes     .default:n   = { true },
5871   gnotes     .bool_set:N  = \c__problems_gnotes_bool,
5872   hints      .default:n   = { true },
5873   hints      .bool_set:N  = \c__problems_hints_bool,
5874   solutions  .default:n   = { true },
5875   solutions  .bool_set:N  = \c__problems_solutions_bool,
5876   pts        .default:n   = { true },
5877   pts        .bool_set:N  = \c__problems_pts_bool,
5878   min        .default:n   = { true },
5879   min        .bool_set:N  = \c__problems_min_bool,
5880   boxed      .default:n   = { true },
5881   boxed      .bool_set:N  = \c__problems_boxed_bool,
5882   unknown    .code:n      = {}
5883 }
5884 \newif\ifsolutions
5885
5886 \ProcessKeysOptions{ problem / pkg }
5887 \bool_if:NTF \c__problems_solutions_bool {
5888   \solutionstrue
5889 }{
5890   \solutionsfalse
5891 }
```

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

```
5892 \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.

```
5893 \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.

```
5894 \def\prob@problem@kw{Problem}
5895 \def\prob@solution@kw{Solution}
5896 \def\prob@hint@kw{Hint}
5897 \def\prob@note@kw{Note}
5898 \def\prob@gnote@kw{Grading}
5899 \def\prob@pt@kw{pt}
5900 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
5901 \AddToHook{begindocument}{
5902   \ltx@ifpackageloaded{babel}{
5903     \makeatletter
5904     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5905     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5906       \input{problem-ngerman.ldf}
5907     }
5908     \clist_if_in:NnT \l_tmpa_clist {finnish}{
5909       \input{problem-finnish.ldf}
5910     }
5911     \clist_if_in:NnT \l_tmpa_clist {french}{
5912       \input{problem-french.ldf}
5913     }
5914     \clist_if_in:NnT \l_tmpa_clist {russian}{
5915       \input{problem-russian.ldf}
5916     }
5917     \makeatother
5918   }{ }
5919 }
```

## 40.2 Problems and Solutions

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

```
5920 \keys_define:nn{ problem / problem }{
5921   id      .str_set:x:N = \l__problems_prob_id_str,
5922   pts     .tl_set:N    = \l__problems_prob_pts_tl,
5923   min     .tl_set:N    = \l__problems_prob_min_tl,
5924   title   .tl_set:N    = \l__problems_prob_title_tl,
5925   type    .tl_set:N    = \l__problems_prob_type_tl,
5926   refnum  .int_set:N    = \l__problems_prob_refnum_int
5927 }
5928 \cs_new_protected:Nn \__problems_prob_args:n {
```

```

5929 \str_clear:N \l__problems_prob_id_str
5930 \tl_clear:N \l__problems_prob_pts_tl
5931 \tl_clear:N \l__problems_prob_min_tl
5932 \tl_clear:N \l__problems_prob_title_tl
5933 \tl_clear:N \l__problems_prob_type_tl
5934 \int_zero_new:N \l__problems_prob_refnum_int
5935 \keys_set:nn { problem / problem }{ #1 }
5936 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
5937   \let\l__problems_prob_refnum_int\undefined
5938 }
5939 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

5940 \newcounter{problem}
5941 \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.

```

5942 \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

```

5943 \newcommand\prob@number{
5944   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
5945     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
5946   }{
5947     \int_if_exist:NTF \l__problems_prob_refnum_int {
5948       \prob@label{\int_use:N \l__problems_prob_refnum_int }
5949     }{
5950       \prob@label\theproblem
5951     }
5952   }
5953 }

```

(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.

```

5954 \newcommand\prob@title[3]{%
5955   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
5956     #2 \l__problems_inclprob_title_tl #3
5957   }{
5958     \tl_if_exist:NTF \l__problems_prob_title_tl {
5959       #2 \l__problems_prob_title_tl #3
5960     }{
5961       #1
5962     }
5963   }
5964 }

```

(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.

```

5965 \def\prob@heading{
5966   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
5967   %\sref@label{id}\prob@problem@kw~\prob@number}{~}
5968 }

```

(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`

```

5969 \newenvironment{sproblem}[1][]{
5970   \__problems_prob_args:n{#1}%\sref@target%
5971   \@in@omtexttrue% we are in a statement (for inline definitions)
5972   \stepcounter{problem}\record@problem
5973   \def\current@section@level{\prob@problem@kw}
5974   \tl_if_exist:NTF \l__problems_inclprob_type_tl {
5975     \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
5976   }{
5977     \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
5978   }
5979   \str_if_exist:NTF \l__problems_inclprob_id_str {
5980     \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
5981   }{
5982     \str_set_eq:NN \sproblemid \l__problems_prob_id_str
5983   }
5984
5985
5986   \clist_set:No \l_tmpa_clist \sproblemtype
5987   \tl_clear:N \l_tmpa_tl
5988   \clist_map_inline:Nn \l_tmpa_clist {
5989     \tl_if_exist:cT {\__problems_sproblem_##1_start:}{
5990       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_start:}}
5991     }
5992   }
5993   \tl_if_empty:NTF \l_tmpa_tl {
5994     \__problems_sproblem_start:
5995   }{
5996     \l_tmpa_tl
5997   }
5998   \stex_ref_new_doc_target:n \sproblemid
5999 }{
6000   \clist_set:No \l_tmpa_clist \sproblemtype
6001   \tl_clear:N \l_tmpa_tl
6002   \clist_map_inline:Nn \l_tmpa_clist {
6003     \tl_if_exist:cT {\__problems_sproblem_##1_end:}{
6004       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_end:}}
6005     }

```

```

6006 }
6007 \tl_if_empty:NTF \l_tmpa_tl {
6008   \__problems_sproblem_end:
6009 }{
6010   \l_tmpa_tl
6011 }
6012
6013
6014 \smallskip
6015 }
6016
6017
6018 \cs_new_protected:Nn \__problems_sproblem_start: {
6019   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
6020 }
6021 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
6022
6023 \newcommand\stexpatchproblem[3][] {
6024   \str_set:Nx \l_tmpa_str{ #1 }
6025   \str_if_empty:NTF \l_tmpa_str {
6026     \tl_set:Nn \__problems_sproblem_start: { #2 }
6027     \tl_set:Nn \__problems_sproblem_end: { #3 }
6028   }{
6029     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
6030     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
6031   }
6032 }
6033
6034
6035 \bool_if:NT \c__problems_boxed_bool {
6036   \surroundwithmdframed{problem}
6037 }

```

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

```

6038 \def\record@problem{
6039   \protected@write\@auxout{}
6040   {
6041     \string\@problem{\prob@number}
6042     {
6043       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
6044         \l__problems_inclprob_pts_tl
6045       }{
6046         \l__problems_prob_pts_tl
6047       }
6048     }%
6049     {
6050       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
6051         \l__problems_inclprob_min_tl
6052       }{
6053         \l__problems_prob_min_tl
6054       }
6055     }
6056   }
6057 }

```

(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).

```
6058 \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.

```
6059 \keys_define:nn { problem / solution }{
6060   id                .str_set_x:N = \l__problems_solution_id_str ,
6061   for               .tl_set:N   = \l__problems_solution_for_tl ,
6062   height            .dim_set:N   = \l__problems_solution_height_dim ,
6063   creators          .clist_set:N = \l__problems_solution_creators_clist ,
6064   contributors      .clist_set:N = \l__problems_solution_contributors_clist ,
6065   srccite           .tl_set:N    = \l__problems_solution_srccite_tl
6066 }
6067 \cs_new_protected:Nn \__problems_solution_args:n {
6068   \str_clear:N \l__problems_solution_id_str
6069   \tl_clear:N \l__problems_solution_for_tl
6070   \tl_clear:N \l__problems_solution_srccite_tl
6071   \clist_clear:N \l__problems_solution_creators_clist
6072   \clist_clear:N \l__problems_solution_contributors_clist
6073   \dim_zero:N \l__problems_solution_height_dim
6074   \keys_set:nn { problem / solution }{ #1 }
6075 }
```

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

```
6076 \newcommand\@startsolution[1][{}]{
6077   \__problems_solution_args:n { #1 }
6078   \@in@omtexttrue% we are in a statement.
6079   \bool_if:NF \c__problems_boxed_bool { \hrule }
6080   \smallskip\noindent
6081   {\textbf\prob@solution@kw : \enspace}
6082   \begin{small}
6083   \def\current@section@level{\prob@solution@kw}
6084   \ignorespacesandpars
6085 }
```

**\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.

```
6086 \newcommand\startsolutions{
6087   \specialcomment{solution}{\@startsolution}{
6088     \bool_if:NF \c__problems_boxed_bool {
6089       \hrule\medskip
6090     }
6091     \end{small}%
6092   }
6093   \bool_if:NT \c__problems_boxed_bool {
6094     \surroundwithmdframed{solution}
6095   }
6096 }
```



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

\stopsolutions

```
6097 \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.

```
6098 \ifsolutions
6099   \startsolutions
6100 \else
6101   \stopsolutions
6102 \fi
```

exnote

```
6103 \bool_if:NTF \c__problems_notes_bool {
6104   \newenvironment{exnote}[1][]{
6105     \par\smallskip\hrule\smallskip
6106     \noindent\textbf{\prob@note@kw : }\small
6107   }{
6108     \smallskip\hrule
6109   }
6110 }{
6111   \excludecomment{exnote}
6112 }
```

hint

```
6113 \bool_if:NTF \c__problems_notes_bool {
6114   \newenvironment{hint}[1][]{
6115     \par\smallskip\hrule\smallskip
6116     \noindent\textbf{\prob@hint@kw :~ }\small
6117   }{
6118     \smallskip\hrule
6119   }
6120 \newenvironment{exhint}[1][]{
6121   \par\smallskip\hrule\smallskip
6122   \noindent\textbf{\prob@hint@kw :~ }\small
6123 }{
6124   \smallskip\hrule
6125 }
6126 }{
6127   \excludecomment{hint}
6128   \excludecomment{exhint}
6129 }
```

gnote

```
6130 \bool_if:NTF \c__problems_notes_bool {
6131   \newenvironment{gnote}[1][]{
6132     \par\smallskip\hrule\smallskip
6133     \noindent\textbf{\prob@gnote@kw : }\small
6134   }{
6135     \smallskip\hrule
6136   }
6137 }{
6138   \excludecomment{gnote}
6139 }
```

## 40.3 Multiple Choice Blocks

```

6140 \newenvironment{mcb}{
6141   \begin{enumerate}
6142 }{
6143   \end{enumerate}
6144 }

```

we define the keys for the mcc macro

```

6145 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
6146   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
6147     \bool_set_true:N #1
6148   }{
6149     \bool_set_false:N #1
6150   }
6151 }
6152 \keys_define:nn { problem / mcc }{
6153   id          .str_set:x:N = \l__problems_mcc_id_str ,
6154   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
6155   T           .default:n   = { true } ,
6156   T           .bool_set:N   = \l__problems_mcc_t_bool ,
6157   F           .default:n   = { true } ,
6158   F           .bool_set:N   = \l__problems_mcc_f_bool ,
6159   Ttext       .code:n       = {
6160     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
6161   } ,
6162   Ftext       .code:n       = {
6163     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
6164   }
6165 }
6166 \cs_new_protected:Nn \l__problems_mcc_args:n {
6167   \str_clear:N \l__problems_mcc_id_str
6168   \tl_clear:N \l__problems_mcc_feedback_tl
6169   \bool_set_true:N \l__problems_mcc_t_bool
6170   \bool_set_true:N \l__problems_mcc_f_bool
6171   \bool_set_true:N \l__problems_mcc_Ttext_bool
6172   \bool_set_false:N \l__problems_mcc_Ftext_bool
6173   \keys_set:nn { problem / mcc }{ #1 }
6174 }

```

\mcc

```

6175 \newcommand\mcc[2][] {
6176   \l__problems_mcc_args:n{ #1 }
6177   \item #2
6178   \ifsolutions
6179     \\\
6180     \bool_if:NT \l__problems_mcc_t_bool {
6181       % TODO!
6182       % \ifcsstring{mcc@T}{T}{\mcc@Ttext}%
6183     }
6184     \bool_if:NT \l__problems_mcc_f_bool {

```

---

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

```

6185         % TODO!
6186         % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
6187     }
6188     \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
6189         !
6190     }{
6191         \l__problems_mcc_feedback_tl
6192     }
6193     \fi
6194 } %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.

```

6195
6196 \keys_define:nn{ problem / inclproblem }{
6197     id      .str_set:N = \l__problems_inclprob_id_str,
6198     pts     .tl_set:N  = \l__problems_inclprob_pts_tl,
6199     min     .tl_set:N  = \l__problems_inclprob_min_tl,
6200     title   .tl_set:N  = \l__problems_inclprob_title_tl,
6201     refnum  .int_set:N  = \l__problems_inclprob_refnum_int,
6202     type    .tl_set:N  = \l__problems_inclprob_type_tl,
6203     mhrepos .str_set:N  = \l__problems_inclprob_mhrepos_str
6204 }
6205 \cs_new_protected:Nn \l__problems_inclprob_args:n {
6206     \str_clear:N \l__problems_prob_id_str
6207     \tl_clear:N \l__problems_inclprob_pts_tl
6208     \tl_clear:N \l__problems_inclprob_min_tl
6209     \tl_clear:N \l__problems_inclprob_title_tl
6210     \tl_clear:N \l__problems_inclprob_type_tl
6211     \int_zero_new:N \l__problems_inclprob_refnum_int
6212     \str_clear:N \l__problems_inclprob_mhrepos_str
6213     \keys_set:nn { problem / inclproblem }{ #1 }
6214     \tl_if_empty:NT \l__problems_inclprob_pts_tl {
6215         \let\l__problems_inclprob_pts_tl\undefined
6216     }
6217     \tl_if_empty:NT \l__problems_inclprob_min_tl {
6218         \let\l__problems_inclprob_min_tl\undefined
6219     }
6220     \tl_if_empty:NT \l__problems_inclprob_title_tl {
6221         \let\l__problems_inclprob_title_tl\undefined
6222     }
6223     \tl_if_empty:NT \l__problems_inclprob_type_tl {
6224         \let\l__problems_inclprob_type_tl\undefined
6225     }
6226     \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
6227         \let\l__problems_inclprob_refnum_int\undefined
6228     }
6229 }

```

```

6230
6231 \cs_new_protected:Nn \__problems_inclprob_clear: {
6232   \let\l__problems_inclprob_id_str\undefined
6233   \let\l__problems_inclprob_pts_tl\undefined
6234   \let\l__problems_inclprob_min_tl\undefined
6235   \let\l__problems_inclprob_title_tl\undefined
6236   \let\l__problems_inclprob_type_tl\undefined
6237   \let\l__problems_inclprob_refnum_int\undefined
6238   \let\l__problems_inclprob_mhrepos_str\undefined
6239 }
6240 \__problems_inclprob_clear:
6241
6242 \newcommand\includeproblem[2][ ]{
6243   \__problems_inclprob_args:n{ #1 }
6244   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
6245     \input{#2}
6246   }{
6247     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
6248       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
6249     }
6250   }
6251   \__problems_inclprob_clear:
6252 }

```

(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.

```

6253 \AddToHook{enddocument}{
6254   \bool_if:NT \c__problems_pts_bool {
6255     \message{Total:~\arabic{pts}~points}
6256   }
6257   \bool_if:NT \c__problems_min_bool {
6258     \message{Total:~\arabic{min}~minutes}
6259   }
6260 }

```

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

```

6261 \def\pts#1{
6262   \bool_if:NT \c__problems_pts_bool {
6263     \marginpar{#1~\prob@pt@kw}
6264   }
6265 }
6266 \def\min#1{
6267   \bool_if:NT \c__problems_min_bool {
6268     \marginpar{#1~\prob@min@kw}
6269   }
6270 }

```

`\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.

```

6271 \newcounter{pts}
6272 \def\show@pts{
6273   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
6274     \bool_if:NT \c__problems_pts_bool {
6275       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
6276       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
6277     }
6278   }{
6279     \tl_if_exist:NT \l__problems_prob_pts_tl {
6280       \bool_if:NT \c__problems_pts_bool {
6281         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
6282         \addtocounter{pts}{\l__problems_prob_pts_tl}
6283       }
6284     }
6285   }
6286 }

```

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

`\show@min`

```

6287 \newcounter{min}
6288 \def\show@min{
6289   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
6290     \bool_if:NT \c__problems_min_bool {
6291       \marginpar{\l__problems_inclprob_min_tl\ min}
6292       \addtocounter{min}{\l__problems_inclprob_min_tl}
6293     }
6294   }{
6295     \tl_if_exist:NT \l__problems_prob_min_tl {
6296       \bool_if:NT \c__problems_min_bool {
6297         \marginpar{\l__problems_prob_min_tl\ min}
6298         \addtocounter{min}{\l__problems_prob_min_tl}
6299       }
6300     }
6301   }
6302 }
6303 </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.

```
6304 <@@=hwexam>
6305 <*cls>
6306 \ProvidesExplClass{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
6307 \RequirePackage{l3keys2e,expl-keystr-compatible}
6308 \DeclareOption*{
6309   \PassOptionsToClass{\CurrentOption}{document-structure}
6310   \PassOptionsToPackage{\CurrentOption}{stex}
6311   \PassOptionsToPackage{\CurrentOption}{hwexam}
6312   \PassOptionsToPackage{\CurrentOption}{tikzinput}
6313 }
6314 \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.

```
6315 \LoadClass{document-structure}
6316 \RequirePackage{stex}
6317 \RequirePackage{hwexam}
6318 \RequirePackage{tikzinput}
6319 \RequirePackage{graphicx}
6320 \RequirePackage{a4wide}
6321 \RequirePackage{amssymb}
6322 \RequirePackage{amstext}
6323 \RequirePackage{amsmath}
```

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

```

6324 \newcommand\assig@default@type{\hwexam@assignment@kw}
6325 \def\document@hwexamtype{\assig@default@type}
6326 <@@=document_structure>
6327 \keys_define:nn { document-structure / document }{
6328 id .str_set_x:N = \c_document_structure_document_id_str,
6329 hwexamtype .tl_set:N = \document@hwexamtype
6330 }
6331 <@@=hwexam>
6332 </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.

```
6333 \*package>
6334 \ProvidesExplPackage{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
6335 \RequirePackage{l3keys2e,expl-keystr-compat}
6336
6337 \newif\iftest\testfalse
6338 \DeclareOption{test}{\testtrue}
6339 \newif\ifmultiple\multiplefalse
6340 \DeclareOption{multiple}{\multipletrue}
6341 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
6342 \ProcessOptions
```

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

```
6343 \RequirePackage{keyval}[1997/11/10]
6344 \RequirePackage{problem}
```

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

```
6345 \newcommand\hwexam@assignment@kw{Assignment}
6346 \newcommand\hwexam@given@kw{Given}
6347 \newcommand\hwexam@due@kw{Due}
6348 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
6349 blank~for~extra~space}
6350 \def\hwexam@minutes@kw{minutes}
6351 \newcommand\correction@probs@kw{prob.}
6352 \newcommand\correction@pts@kw{total}
6353 \newcommand\correction@reached@kw{reached}
6354 \newcommand\correction@sum@kw{Sum}
6355 \newcommand\correction@grade@kw{grade}
6356 \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

```

6357 \AddToHook{begindocument}{
6358 \ltx@ifpackageloaded{babel}{
6359 \makeatletter
6360 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6361 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6362 \input{hwexam-ngerman.ldf}
6363 }
6364 \clist_if_in:NnT \l_tmpa_clist {finnish}{
6365 \input{hwexam-finnish.ldf}
6366 }
6367 \clist_if_in:NnT \l_tmpa_clist {french}{
6368 \input{hwexam-french.ldf}
6369 }
6370 \clist_if_in:NnT \l_tmpa_clist {russian}{
6371 \input{hwexam-russian.ldf}
6372 }
6373 \makeatother
6374 }{}
6375 }
6376

```

## 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.

```

6377 \newcounter{assignment}
6378 \numberproblemsin{assignment}
6379 \renewcommand\prob@label[1]{\assignment@number.#1}

```

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

```

6380 \keys_define:nn { hwexam / assignment } {
6381 id .str_set:N = \l__hwexam_assign_id_str,
6382 number .int_set:N = \l__hwexam_assign_number_int,
6383 title .tl_set:N = \l__hwexam_assign_title_tl,
6384 type .tl_set:N = \l__hwexam_assign_type_tl,
6385 given .tl_set:N = \l__hwexam_assign_given_tl,
6386 due .tl_set:N = \l__hwexam_assign_due_tl,
6387 loadmodules .code:n = {
6388 \bool_set_true:N \l__hwexam_assign_loadmodules_bool
6389 }
6390 }
6391 \cs_new_protected:Nn \__hwexam_assignment_args:n {
6392 \str_clear:N \l__hwexam_assign_id_str
6393 \int_set:Nn \l__hwexam_assign_number_int {-1}
6394 \tl_clear:N \l__hwexam_assign_title_tl
6395 \tl_clear:N \l__hwexam_assign_type_tl
6396 \tl_clear:N \l__hwexam_assign_given_tl
6397 \tl_clear:N \l__hwexam_assign_due_tl
6398 \bool_set_false:N \l__hwexam_assign_loadmodules_bool

```

```

6399 \keys_set:nn { hwexam / assignment }{ #1 }
6400 }

```

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.

```

6401 \newcommand\given@due[2]{
6402 \bool_lazy_all:nF {
6403 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
6404 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
6405 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
6406 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
6407 }{ #1 }
6408
6409 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
6410 \tl_if_empty:NF \l__hwexam_assign_given_tl {
6411 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
6412 }
6413 }{
6414 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
6415 }
6416
6417 \bool_lazy_or:nnF {
6418 \bool_lazy_and_p:nn {
6419 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
6420 }{
6421 \tl_if_empty_p:V \l__hwexam_assign_due_tl
6422 }
6423 }{
6424 \bool_lazy_and_p:nn {
6425 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
6426 }{
6427 \tl_if_empty_p:V \l__hwexam_assign_due_tl
6428 }
6429 }{ ,~ }
6430
6431 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
6432 \tl_if_empty:NF \l__hwexam_assign_due_tl {
6433 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
6434 }
6435 }{
6436 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
6437 }
6438
6439 \bool_lazy_all:nF {
6440 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
6441 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
6442 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
6443 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
6444 }{ #2 }
6445 }

```

`\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.

```

6446 \newcommand\assignment@title[3]{
6447 \tl_if_empty:NTF \l__hwexam_inclasssign_title_tl {
6448 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
6449 #1
6450 }{
6451 #2\l__hwexam_assign_title_tl#3
6452 }
6453 }{
6454 #2\l__hwexam_inclasssign_title_tl#3
6455 }
6456 }

```

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

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

```

6457 \newcommand\assignment@number{
6458 \int_compare:nNnTF \l__hwexam_inclasssign_number_int = {-1} {
6459 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
6460 \arabic{assignment}
6461 } {
6462 \int_use:N \l__hwexam_assign_number_int
6463 }
6464 }{
6465 \int_use:N \l__hwexam_inclasssign_number_int
6466 }
6467 }

```

(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.

```

6468 \newenvironment{assignment}[1][]{
6469 \__hwexam_assignment_args:n { #1 }
6470 %\sref@target
6471 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
6472 \global\stepcounter{assignment}
6473 }{
6474 \global\setcounter{assignment}{\int_use:N\l__hwexam_assign_number_int}
6475 }
6476 \setcounter{problem}{0}
6477 \def\current@section@level{\document@hwexamtype}
6478 %\sref@label@id{\document@hwexamtype \thesection}
6479 \begin{@assignment}
6480 }{
6481 \end{@assignment}
6482 }

```

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

```

6483 \def\ass@title{
6484 \protect\document@hwexamtype~\arabic{assignment}
6485 \assignment@title{}\{;\}{} -- \given@due{}\}{}
6486 }
6487 \ifmultiple
6488 \newenvironment{@assignment}{
6489 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
6490 \begin{omgroup}[loadmodules]{\ass@title}
6491 }{
6492 \begin{omgroup}{\ass@title}
6493 }
6494 }{
6495 \end{omgroup}
6496 }

```

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

```

6497 \else
6498 \newenvironment{@assignment}{
6499 \begin{center}\bf
6500 \Large@title\strut\\
6501 \document@hwexamtype~\arabic{assignment}\assignment@title{}\{;\}{}{\}{}
6502 \large\given@due{--;\}{}\}{}
6503 \end{center}
6504 }{}
6505 \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.

```

6506 \keys_define:nn { hwexam / inclassignment } {
6507 %id .str_set_x:N = \l__hwexam_assign_id_str,
6508 number .int_set:N = \l__hwexam_inclassign_number_int,
6509 title .tl_set:N = \l__hwexam_inclassign_title_tl,
6510 type .tl_set:N = \l__hwexam_inclassign_type_tl,
6511 given .tl_set:N = \l__hwexam_inclassign_given_tl,
6512 due .tl_set:N = \l__hwexam_inclassign_due_tl,
6513 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
6514 }
6515 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
6516 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
6517 \tl_clear:N \l__hwexam_inclassign_title_tl
6518 \tl_clear:N \l__hwexam_inclassign_type_tl
6519 \tl_clear:N \l__hwexam_inclassign_given_tl
6520 \tl_clear:N \l__hwexam_inclassign_due_tl
6521 \str_clear:N \l__hwexam_inclassign_mhrepos_str
6522 \keys_set:nn { hwexam / inclassignment }{ #1 }
6523 }
6524 \__hwexam_inclassignment_args:n {}
6525
6526 \newcommand\inputassignment[2][{}]{

```

```

6527 \_hwexam_inclassnment_args:n { #1 }
6528 \str_if_empty:NTF \l__hwexam_inclassn_mhrepos_str {
6529 \input{#2}
6530 }{
6531 \stex_in_repository:nn{\l__hwexam_inclassn_mhrepos_str}{
6532 \input{\mhp{path}\l__hwexam_inclassn_mhrepos_str}{#2}}
6533 }
6534 }
6535 \_hwexam_inclassnment_args:n {}
6536 }
6537 \newcommand\includeassignment[2][]{
6538 \newpage
6539 \inputassignment[#1]{#2}
6540 }

```

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

## 42.4 Typesetting Exams

\quizheading

```

6541 \ExplSyntaxOff
6542 \newcommand\quizheading[1]{%
6543 \def\@tas{#1}%
6544 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
6545 \ifx\@tas\@empty\else%
6546 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
6547 \fi%
6548 }
6549 \ExplSyntaxOn

```

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

\testheading

```

6550
6551 \def\hwexamheader{\input{hwexam-default.header}}
6552
6553 \def\hwexamminutes{
6554 \tl_if_empty:NTF \testheading@duration {
6555 {\testheading@min}~\hwexam@minutes@kw
6556 }{
6557 \testheading@duration
6558 }
6559 }
6560
6561 \keys_define:nn { hwexam / testheading } {
6562 min .tl_set:N = \testheading@min,
6563 duration .tl_set:N = \testheading@duration,
6564 reqpts .tl_set:N = \testheading@reqpts,
6565 tools .tl_set:N = \testheading@tools
6566 }
6567 \cs_new_protected:Nn \_hwexam_testheading_args:n {
6568 \tl_clear:N \testheading@min
6569 \tl_clear:N \testheading@duration

```

```

6570 \tl_clear:N \testheading@reqpts
6571 \tl_clear:N \testheading@tools
6572 \keys_set:nn { hwexam / testheading }{ #1 }
6573 }
6574 \newenvironment{testheading}[1][]{
6575   \_hwexam_testheading_args:n{ #1 }
6576   \newcount\check@time\check@time=\testheading@min
6577   \advance\check@time by -\theassignment@totalmin
6578   \newif\if@bonuspoints
6579   \tl_if_empty:NTF \testheading@reqpts {
6580     \@bonuspointsfalse
6581   }{
6582     \newcount\bonus@pts
6583     \bonus@pts=\theassignment@totalpts
6584     \advance\bonus@pts by -\testheading@reqpts
6585     \edef\bonus@pts{\the\bonus@pts}
6586     \@bonuspointstrue
6587   }
6588   \edef\check@time{\the\check@time}
6589
6590   \makeatletter\hwexamheader\makeatother
6591 }{
6592   \newpage
6593 }

```

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

\testspace

```

6594 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

6595 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

6596 \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.

```

6597 <@=problems>
6598 \renewcommand\@problem[3]{
6599   \stepcounter{assignment@probs}
6600   \def\__problemspts{#2}
6601   \ifx\__problemspts\@empty\else
6602     \addtocounter{assignment@totalpts}{#2}
6603   \fi
6604   \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
6605   \xdef\correction@probs{\correction@probs & #1}%
6606   \xdef\correction@pts{\correction@pts & #2}
6607   \xdef\correction@reached{\correction@reached &}

```

```

6608 }
6609 <@@=hwexam>

```

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

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

```

6610 \newcounter{assignment@probs}
6611 \newcounter{assignment@totalpts}
6612 \newcounter{assignment@totalmin}
6613 \def\correction@probs{\correction@probs@kw}
6614 \def\correction@pts{\correction@pts@kw}
6615 \def\correction@reached{\correction@reached@kw}
6616 \stepcounter{assignment@probs}
6617 \newcommand\correction@table{
6618 \resizebox{\textwidth}{!}{%
6619 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
6620 &\multicolumn{\theassignment@probs}{c|}||%|
6621 {\footnotesize\correction@forgrading@kw} &\\ \hline
6622 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
6623 \correction@pts & \theassignment@totalpts & \\ \hline
6624 \correction@reached & & \[.7cm]\hline
6625 \end{tabular}}
6626 </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}

```