

# The sTeX3 Package \*

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

2022-02-13

## Abstract

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

# Contents

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

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

<b>21 document-structure: Semantic Markup for Open Mathematical Documents in <math>\text{\LaTeX}</math></b>	<b>53</b>
21.1 Introduction . . . . .	53
21.2 The User Interface . . . . .	54
21.2.1 Package and Class Options . . . . .	54
21.2.2 Document Structure . . . . .	54
21.2.3 Ignoring Inputs . . . . .	56
21.2.4 Structure Sharing . . . . .	56
21.2.5 Global Variables . . . . .	56
21.2.6 Colors . . . . .	57
21.3 Limitations . . . . .	57
<b>22 NotesSlides – Slides and Course Notes</b>	<b>58</b>
22.1 Introduction . . . . .	58
22.2 The User Interface . . . . .	58
22.2.1 Package Options . . . . .	58
22.2.2 Notes and Slides . . . . .	59
22.2.3 Header and Footer Lines of the Slides . . . . .	60
22.2.4 Frame Images . . . . .	60
22.2.5 Colors and Highlighting . . . . .	61
22.2.6 Front Matter, Titles, etc. . . . .	61
22.2.7 Excursions . . . . .	61
22.2.8 Miscellaneous . . . . .	62
22.3 Limitations . . . . .	62
<b>23 problem.sty: An Infrastructure for formatting Problems</b>	<b>63</b>
23.1 Introduction . . . . .	63
23.2 The User Interface . . . . .	63
23.2.1 Package Options . . . . .	63
23.2.2 Problems and Solutions . . . . .	64
23.2.3 Multiple Choice Blocks . . . . .	65
23.2.4 Including Problems . . . . .	65
23.2.5 Reporting Metadata . . . . .	65
23.3 Limitations . . . . .	65
<b>24 hwexam.sty/cls: An Infrastructure for formatting Assignments and Exams</b>	<b>67</b>
24.1 Introduction . . . . .	68
24.2 The User Interface . . . . .	68
24.2.1 Package and Class Options . . . . .	68
24.2.2 Assignments . . . . .	68
24.2.3 Typesetting Exams . . . . .	68
24.2.4 Including Assignments . . . . .	69
24.3 Limitations . . . . .	69
<b>IV Implementation</b>	<b>71</b>

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

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

**Part I**  
**Manual**

# Chapter 1

## What is sTeX?

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

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

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

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

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

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



# Chapter 2

## Quickstart

### 2.1 Setup

#### 2.1.1 The sTeX IDE

TODO: VSCode Plugin

#### 2.1.2 Manual Setup

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

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

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

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

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

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

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

---

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

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

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

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

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

The document we will consider is the following:

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

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

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

Compiling this document with `pdflatex` should yield the output

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

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

### \usemodule

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

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

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

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

---

`\symref`  
`\symname`

---

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

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

---

`\importmodule`

---

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

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

## Chapter 3

# Using Semantic Macros

TODO

## Chapter 4

# TeX Archives

### 4.1 The Local MathHub-Directory

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

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

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

### 4.2 The Structure of TeX Archives

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

Each such archive needs two subdirectories:

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

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

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

### 4.3 MANIFEST.MF-Files

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

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

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

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

`source-base` or

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

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

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

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

## Chapter 5

# Creating New Modules and Symbols

TODO

### 5.1 Advanced Structuring Mechanisms

Given modules:

#### Example 1

```
\begin{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}{{#1}^{\comp{-1}}}
\end{smodule}
```

Module 1:  
Module 2:  
Module 3:

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

## Example 2

```
\begin{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 4:  
Test:  $a \cdot (c + d \cdot e)$

TODO: explain donotclone

## Example 3

```
\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 5:

## 5.2 Primitive Symbols (The $\text{\texttt{STEX}}$ Metatheory)



## Chapter 6

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

## Chapter 7

# Additional Packages

7.1 Modular Document Structuring

7.2 Slides and Course Notes

7.3 Homework, Problems and Exams

# Chapter 8

# Stuff

## 8.1 Modules

---

`\sTeX` Both print this  $\text{\TeX}$  logo.  
`\stex`

---

### 8.1.1 Semantic Macros and Notations

Semantic macros invoke a formally declared symbol.

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

**Module 6:** For example, to introduce binary multiplication, we can do `\symdecl[args=2]{mult}`. We can then supply the semantic macro with arbitrarily many notations, such as `\notation{mult}{#1 #2}`.

#### Example 4

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

$ab$

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

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

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

### Example 5

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

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

.

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

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

### Example 6

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

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

.

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

### Example 7

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

Multiplying again by  $b$  yields...

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

### Example 8

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

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

<sup>4</sup>EdNOTE: TODO

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

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

### Example 9

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

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

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

### Example 10

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

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

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

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

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

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

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

## Other Argument Types

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

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

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

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

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

**Module 7:** b-type arguments are indistinguishable from i-type arguments within  $\text{\TeX}$ , but are treated very differently in OMDoc and by MMT. More interesting *within*  $\text{\TeX}$  are a-type arguments, which represent (associative) arguments of flexible arity, which are provided as comma-separated lists. This allows e.g. better representing the `\mult`-macro above:

### Example 11

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

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

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

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

### Example 12

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

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

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

5 6

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

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

## Precedences

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

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

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

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

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

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

For example:

**Module 8:**

### Example 13

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

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

## 8.1.2 Archives and Imports

### Namespaces

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

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

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

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

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

### Paths in Import-Statements

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

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

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

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

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

---

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



## Part II

# Documentation

# Chapter 9

## sTeX-Basics

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

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

**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

---

$\backslash$ sTeX	Both print this sTeX logo.
$\backslash$ stex	

---

---

$\backslash$ stex_debug:nn	$\backslash$ stex_debug:nn { $\langle log-prefix \rangle$ } { $\langle message \rangle$ }
----------------------------	---

---

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

---

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

---

---

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

---

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

---

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

---

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

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

---

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

---

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

---

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

---

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

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

---

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

---

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

# Chapter 10

## sTEX-MathHub

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

### 10.1 Macros and Environments

---

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

---

#### 10.1.1 Files, Paths, URIs

---

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

---

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

---

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

---

---

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

---

---

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

---

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

---

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

---

---

**`\g_stex_currentfile_seq`**

---

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

**Test 1**

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

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

**10.1.2 MathHub Archives**

---

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

---

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

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

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

---

**`\l_stex_current_repository_prop`**

---

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

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

## Test 2

```

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

```

```

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

```

## Chapter 11

# sTeX-References

Code related to links and cross-references

### 11.1 Macros and Environments

# Chapter 12

## sTeX-Modules

Code related to Modules

### 12.1 Macros and Environments

---

---

`\l_stex_current_module_str`

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

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

Additionally, it stores:

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

---

---

`\l_stex_all_modules_seq`

Stores full URIs for all modules currently in scope.



---

```
\g_stex_module_files_prop
\g_stex_modules_in_file_seq
```

---

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

---

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

---



---

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

---

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

---

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

---

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

---

```
\stex_add_constant_to_current_module:n
```

---

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

---

```
\stex_add_import_to_current_module:n
```

---

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

---

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

---

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

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

---

```
\stex_modules_current_namespace:
```

---

Computes the current namespace

### Test 3

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

```

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

```

### 12.1.1 The module-environment

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

---

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

---

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

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

### Test 4

```

\ExplSyntaxOn
\stex_set_current_repository:n {Foo/Bar}
\seq_pop_right:NN \g_stex_currentfile_seq \l_tmpa_tl
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{tests} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{Foo} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{Bar} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{source} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{Foo.tex} }
\begin{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 9:  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 10: 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 11:**  
**Module 12:**  
**Module 13:**    file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?STEXModuleTest1  
file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?STEXModuleTest2  
file://home/jazzpirate/work/Software/ext/sTeX/doc/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_in_smsmode:nn { foo } {
\input{tests/sometest.tex}
}
\ExplSyntaxOff

```

### 13.1.2 Imports and Inheritance

---

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

---

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

#### Test 8

```

\begin{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 14:      Meaning: >macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo?foo}<
                Meaning: >macro:->\protect \bar <
Module 15:      Meaning: >macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo?foo}<
Module 16:      Meaning: >macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo?foo}<

```

---

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

---

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

## Test 9

```

\begin{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 17:**

**Module 18:** Meaning:  $\text{\>macro:->\stex\_invoke\_symbol:n \{file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest1?foo\}}$

**Module 19:** Meaning:  $\text{\>undefined}$

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

All modules: <http://mathhub.info/sTeX?Metatheory>, file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest3, file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest2

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

## Test 10

```

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

```

Circular dependencies:

**Module 20:**  $\text{\>macro:->\stex\_invoke\_symbol:n \{http://mathhub.info/tests/Foo/Bar/circular1?Circular1?fooA\}}$

$\text{\>macro:->\stex\_invoke\_symbol:n \{http://mathhub.info/tests/Bar/Foo/circular2?Circular2?fooB\}}$

---

---

`\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\Nat}{x\geq0}`.

---

---

`\stex_symdecl_do:n`

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

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

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

### Test 11

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

```
Module 21:      Meaning: >macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?SymdeclTest?foo}<
Result: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?SymdeclTest?foo
Meaning: >macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?SymdeclTest?bardef}<
```

---

---

`\l_stex_all_symbols_seq`

Stores full URIs for all modules currently in scope.

---

---

`\stex_get_symbol:n`

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

---

---

`\notation`

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

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

---

---

`\stex_notation_do:nn`

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

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

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

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

### Test 12

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

Module 22:

---

---

`\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 23:  $a + b + c$

# 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 24:**  $\langle a^b_c \rangle$  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} _ {#2} }
$\foobar a{b,c,d,e,f}g$ and $\foobar[foo] a{b,c}g$ and $\foobar abc$

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

**Module 25:**  $\langle a \mid [b;c;d;e;f]^g \rangle$  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 26:

```
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`  
`\compemph`  
`\compemph@uri`  
`\defemph`  
`\defemph@uri`  
`\symrefemph`  
`\symrefemph@uri`

---

`\comp{<args>}`

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

The precise behavior is governed by `\@comp`, which takes as additional argument the URI of the current symbol. By default, `\@comp` adds the URI as a PDF tooltip and colors the highlighted part in blue.

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

---

`\STEXinvisible`

---

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

---

`\ellipses`

---

TODO

## Chapter 16

# TeX-Structural Features

Code related to structural features

### 16.1 Macros and Environments

#### 16.1.1 Structures

`mathstructure` TODO

# Chapter 17

## TeX-Statements

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

### 17.1 Macros and Environments

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



## Chapter 18

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

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

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

## Contents

## 18.1 Introduction

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

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

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

Example 1: A very explicit proof, marked up semantically

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

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

## 18.2 The User Interface

### 18.2.1 Package Options

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

### 18.2.2 Proofs and Proof steps

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

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

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

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

### 18.2.3 Justifications

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

#### 18.2.4 Proof Structure

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

### 18.2.5 Proof End Markers

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

$$\backslash\text{proofend}$$

$$\backslash\text{sProofEndSymbol}$$

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

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

### 18.2.6 Configuration of the Presentation

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

Environment	configuration macro	value
<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

$$\backslash\text{pstlabelstyle}$$

`\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  $\text{\LaTeX}$  `\@for...:=... \do{...}` macro; see Figure 2 for examples.

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

Figure 2: Configuration Proof Step Label Styles

## 18.3 Limitations

In this section we document known limitations. If you want to help alleviate them, please feel free to contact the package author. Some of them are currently discussed in the  $\text{\LaTeX}$  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  $\S\TeX$  collection, a version of  $\TeX$ / $\LaTeX$  that allows to markup  $\TeX$ / $\LaTeX$  documents semantically without leaving the document format, essentially turning  $\TeX$ / $\LaTeX$  into a document format for mathematical knowledge management (MKM).

This package supplies an infrastructure for writing OMDOC documents in  $\LaTeX$ . This includes a simple structure sharing mechanism for  $\S\TeX$  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  $\S\TeX$  sources, or after translation.

### 21.1 Introduction

$\S\TeX$  is a version of  $\TeX$ / $\LaTeX$  that allows to markup  $\TeX$ / $\LaTeX$  documents semantically without leaving the document format, essentially turning  $\TeX$ / $\LaTeX$  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  $\S\TeX$  sources, or after translation. Currently, trans-document referencing provided by this package can only be used in the  $\S\TeX$  collection.

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

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

## 21.2 The User Interface

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

### 21.2.1 Package and Class Options

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

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

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

### 21.2.2 Document Structure

<code>document</code>	The top-level <code>document</code> environment can be given key/value information by the
<code>\documentkeys</code>	<code>\documentkeys</code> macro in the preamble <sup>2</sup> . This can be used to give metadata about the
<code>id</code>	document. For the moment only the <code>id</code> key is used to give an identifier to the <code>omdoc</code>
<code>omgroup</code>	element resulting from the L <sup>A</sup> T <sub>E</sub> XML transformation.
	The structure of the document is given by the <code>omgroup</code> environment just like in OM-
	DOC. In the L <sup>A</sup> T <sub>E</sub> X route, the <code>omgroup</code> environment is flexibly mapped to sectioning com-
	mands, inducing the proper sectioning level from the nesting of <code>omgroup</code> environments.
	Correspondingly, the <code>omgroup</code> environment takes an optional key/value argument for
<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.

---

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

2022-02-13

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

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

[illegible]

Example 8: A generated test heading.

Part IV

# Implementation

## Chapter 25

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

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

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

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

### 25.2 Preliminaries

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

Package options:

```

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

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

```

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

## 25.3 Messages and logging

```

42 <@@=stex_log>

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

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

```

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

Redirecting messages:

```

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

```

## 25.4 Persistence

77 `<@=stex_persist>`

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

```

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

```

(End definition for `\c__stex_persist_sms_iow`.)

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

```

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

```

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

## 25.5 HTML Annotations

```

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

```

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

```

98 \rustex_add_Namespace:nn{stex}{http://kwarc.info/ns/sTeX}

```

`\if@latexml` Conditionals for L<sup>A</sup>T<sub>E</sub>X<sub>M</sub>L:

```

\latexml_if_p:
\latexml_if:TF
99 \ifcsname if@latexml\endcsname\else

```



```

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

```

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

`\l__stex_annotate_arg_tl` Used by annotation macros to ensure that the HTML output to annotate is not empty.  
`\c__stex_annotate_emptyarg_tl`

```

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

```

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

`\_stex_annotate_checkempty:n`

```

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

```

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

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

```

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

```

(End definition for `\l_stex_html_do_output_bool` and `\stex_if_do_html:`. These functions are documented on page ??.)

`\stex_suppress_html:n` Whether to (locally) produce HTML output

```

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

```

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

`\stex_annotate:env`

`\stex_annotate_invisible:n`

`\stex_annotate_invisible:nnn`

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

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

```

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

```

```

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

```

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

## 25.6 Languages

```

233 <@=stex_language>

```

`\c_stex_languages_prop`  
`\c_stex_language_abbrevs_prop`

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

```

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

```

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

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

```

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

```

## 25.7 Activating/Deactivating Macros

`\stex_deactivate_macro:Nn`

```

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

```

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

`\stex_reactivate_macro:N`

```

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

```

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

`\stex_do_aftergroup:nn`

```

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

```

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

```

300 </package>

```

## Chapter 26

# STEX -MathHub Implementation

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

### 26.1 Generic Path Handling

We treat paths as L<sup>A</sup>T<sub>E</sub>X3-sequences (of the individual path segments, i.e. separated by a /-character) unix-style; i.e. a path is absolute if the sequence starts with an empty entry.

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

```

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

```

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

`\stex_path_to_string:NN`  
`\stex_path_to_string:N`

```

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

```

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

`\c__stex_path_dot_str` . and .., respectively.  
`\c__stex_path_up_str`

```

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

```

(End definition for `\c__stex_path_dot_str` and `\c__stex_path_up_str`.)

`\stex_path_canonicalize:N` Canonicalizes the path provided; in particular, resolves . and .. path segments.

```

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

```

```

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

```

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

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

```

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

```

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

## 26.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

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

```

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

We determine the PWD

`\c_stex_pwd_seq`  
`\c_stex_pwd_str`

```

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

```



```

406 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
407 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
408 \stex_debug:nn {mathhub} {PWD:~\str_use:N\c_stex_pwd_str}

```

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

## 26.3 File Hooks and Tracking

```

409 <@=stex_files>

```

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

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

`\g_stex_files_stack` keeps track of file changes

```

410 \seq_gclear_new:N\g_stex_files_stack

```

(End definition for `\g_stex_files_stack`.)

`\c_stex_mainfile_seq`  
`\c_stex_mainfile_str`

```

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

```

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

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

```

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

```

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

## 26.4 MathHub Repositories

```

438 <@@=stex_mathhub>

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

```

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

```

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

```

\l_stex_mathhub_manifest_file_seq
475 \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:

```

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

```

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

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

```

511 \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:

```

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

```

```

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

```

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

`\stex_set_current_repository:n`

```

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

```

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

`\stex_require_repository:n`

```

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

```

```

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

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

573 %\prop_new:N \l_stex_current_repository_prop
574
575 \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
576 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
577   \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
578 } {
579   \__stex_mathhub_parse_manifest:n { main }
580   \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
581   \l_tmpa_str
582   \prop_set_eq:cN { c_stex_mathhub\_l_tmpa_str _manifest_prop }
583   \c_stex_mathhub_main_manifest_prop
584   \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
585   \stex_debug:nn{mathhub}{Current~repository:~
586     \prop_item:Nn \l_stex_current_repository_prop {id}
587   }
588 }

```

(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.

```

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

```

```

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

```

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

```

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

```

```

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

```

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

#### `\mhpath`

```

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

```

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

#### `\libinput`

```

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

```

```

706     }
707   }{}
708 }
709 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
710   / \l_tmpa_str / lib / #1.tex
711 }{
712   \bool_set_true:N \l_tmpa_bool
713   \tl_put_right:Nx \l_tmpa_tl {
714     \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
715       / \l_tmpa_str / lib / #1.tex}
716   }
717 }{}
718 \bool_if:NF \l_tmpa_bool {
719   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
720 }
721 \l_tmpa_tl
722 }

```

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

#### `\libusepackage`

```

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

```



```

756     }
757   }{}
758   \bool_if:NF \l_libusepackage_bool {
759     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
760   }
761   \l_tmpa_tl
762 }

```

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

```

763
764 \AddToHook{begindocument}{
765 \ltx@ifpackageloaded{graphicx}{
766   \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
767   \newcommand\mhgraphics[2][]{\%
768     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
769     \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}}
770   \newcommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
771 }{}
772 \ltx@ifpackageloaded{listings}{
773   \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
774   \newcommand\lstinputmhlisting[2][]{\%
775     \def\lst@mhrepos{}\setkeys{lst}{#1}%
776     \lstinputlisting[#1]{\mhp\lst@mhrepos{#2}}}
777   \newcommand\clstinputmhlisting[2][]{\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
778 }{}
779 }
780
781
782 \</package>

```

## Chapter 27

# STEX -References Implementation

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

### 27.1 Document URIs and URLs

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

```

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

```

```

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

```

## 27.2 Setting Reference Targets

```

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

```

```

918 }
919 \seq_gput_right:No \g__stex_refs_all_refs_seq \l_tmpa_str
920 \stex_if_smsmode:TF {
921   \stex_get_document_url:
922   \str_gset_eq:cN {sref_url_\l_tmpa_str _str}\l_stex_current_docurl_str
923   \str_gset_eq:cN {sref_\l_tmpa_str _type}\c__stex_refs_url_str
924 }{
925   \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter\unexpanded\expandafter{
926   \exp_args:Nx\label{sref_\l_tmpa_str}
927   \exp_args:NNNx\immediate\write\@auxout{\stexauxadddocref{\l_tmpa_str}}
928   \str_gset:cx {sref_\l_tmpa_str _type}\c__stex_refs_ref_str
929 }
930 }
931 \cs_new_protected:Npn \stexauxadddocref #1 {
932   \str_set:Nx \l_tmpa_str {#1}
933   \str_gset_eq:cN{sref_\l_tmpa_str _type}\c__stex_refs_ref_str
934   \seq_gput_right:Nx \g__stex_refs_all_refs_seq {\l_tmpa_str}
935 }
936 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
937   \stex_get_document_uri:
938   \stex_if_smsmode:TF {
939     \stex_get_document_url:
940     \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str
941     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
942   }
943   }{
944     \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter{\@currentlabel\iffalse}}
945     \exp_args:Nx\label{sref_sym_#1}
946   }
947   \exp_args:NNNx\immediate\write\@auxout{\stexauxadddocref{sym_#1}}
948   \str_gset:cx {sref_sym_#1_type}\c__stex_refs_ref_str
949 }
950 }

```

## 27.3 Using References

```

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

```

```

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

```

```

1023 \str_if_empty:NTF \l__stex_refs_indocument_str {
1024   \tl_set:Nn \l_tmpa_tl {
1025     \l__stex_refs_fallback_tl
1026   }
1027   \tl_if_exist:cT{sref_sym_\l_stex_get_symbol_uri_str_type}{
1028     \tl_set:Nn \l_tmpa_tl {
1029       % doc uri in \l_tmpb_str
1030       \str_set:Nx \l_tmpa_str {\use:c{sref_sym_\l_stex_get_symbol_uri_str_type}}
1031       \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1032         % reference
1033         \cs_if_exist:cTF{autoref}{
1034           \l__stex_refs_pre_tl\autoref{sref_sym_\l_stex_get_symbol_uri_str}\l__stex_refs_post_tl
1035         }{
1036           \l__stex_refs_pre_tl\ref{sref_sym_\l_stex_get_symbol_uri_str}\l__stex_refs_post_tl
1037         }
1038       }{
1039         % URL
1040         \if_bool:N \c__stex_refs_hyperref_bool {
1041           \exp_args:Nx \href{\use:c{sref_sym_url_\l_stex_get_symbol_uri_str_str}}{\l__stex_refs_ref_str}
1042         }{
1043           \l__stex_refs_fallback_tl
1044         }
1045       }
1046     }
1047   }
1048   \l_tmpa_tl
1049 }{
1050   % TODO
1051 }
1052 }
1053
1054 \cs_new_protected:Npn \srefsymuri #1 #2 {
1055   \hyperref[sref_sym_#1]{#2}
1056 }
1057
1058 </package>

```

## Chapter 28

# STEX -Modules Implementation

```
1059 <*package>
1060
1061 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1062
1063 <@@=stex_modules>
1064
1065     Warnings and error messages
1066 \msg_new:nnn{stex}{error/unknownmodule}{
1067     No~module~#1~found
1068 }
1069 \msg_new:nnn{stex}{error/syntax}{
1070     Syntax~error:~#1
1071 }
1072 \msg_new:nnn{stex}{error/siglanguage}{
1073     Module~#1~declares~signature~#2,~but~does~not~
1074     declare~its~language
1075 }
1076 \msg_new:nnn{stex}{error/conflictingmodules}{
1077     Conflicting~imports~for~module~#1
1078 }
1079
1080 \l_stex_current_module_str The current module:
1081 \str_new:N \l_stex_current_module_str
1082
1083 (End definition for \l_stex_current_module_str. This variable is documented on page 26.)
1084
1085 \l_stex_all_modules_seq Stores all available modules
1086 \seq_new:N \l_stex_all_modules_seq
1087
1088 (End definition for \l_stex_all_modules_seq. This variable is documented on page 26.)
1089
1090 \stex_if_in_module_p:
1091 \stex_if_in_module:TF
1092 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1093     \str_if_empty:NTF \l_stex_current_module_str
1094     \prg_return_false: \prg_return_true:
1095 }
```



(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`

```
1084 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1085   \prop_if_exist:cTF { c_stex_module_#1_prop }
1086   \prg_return_true: \prg_return_false:
1087 }
```

(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:

```
1088 \cs_new_protected:Nn \stex_add_to_current_module:n {
1089   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1090 }
1091 \cs_new_protected:Npn \STEXexport {
1092   \begingroup
1093   \newlinechar=-1\relax
1094   \endlinechar=-1\relax
1095   %\catcode'\ = 9\relax
1096   \expandafter\endgroup\STEXexport:n
1097 }
1098 \cs_new_protected:Nn \STEXexport:n {
1099   \ignorespaces #1
1100   \stex_add_to_current_module:n { \ignorespaces #1 }
1101   \stex_smsmode_set_codes:
1102 }
1103 \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`

```
1104 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1105   \str_set:Nx \l_tmpa_str { #1 }
1106   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1107 }
1108
1109 %\cs_new_protected:Nn \stex_add_field_to_current_module:n {
1110 % \str_set:Nx \l_tmpa_str { #1 }
1111 % \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _fields} { \l_tmpa_str }
1112 %}
```

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

`\stex_collect_imports:n`

```
1113 \cs_new_protected:Nn \stex_collect_imports:n {
1114   \seq_clear:N \l_stex_collect_imports_seq
1115   \__stex_modules_collect_imports:n {#1}
1116 }
1117 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1118   \seq_map_inline:cn {c_stex_module_#1_imports} {
1119     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1120       \__stex_modules_collect_imports:n { ##1 }
1121     }
1122 }
```

```

1122 }
1123 \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1124   \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1125 }
1126 }

```

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

`\stex_add_import_to_current_module:n`

```

1127 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1128   \str_set:Nx \l_tmpa_str { #1 }
1129   \exp_args:Nno
1130   \seq_if_in:cnF{c_stex_module\l_stex_current_module_str_imports}\l_tmpa_str{
1131     \seq_gput_right:co{c_stex_module\l_stex_current_module_str_imports}\l_tmpa_str
1132   }
1133 }

```

(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).

```

1134 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
1135   \str_set:Nx \l_tmpa_str { #1 }
1136   \seq_set_eq:NN \l_tmpa_seq #2
1137   % split off file extension
1138   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1139   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1140   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1141   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1142
1143   \bool_set_true:N \l_tmpa_bool
1144   \bool_while_do:Nn \l_tmpa_bool {
1145     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1146     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1147       {source} { \bool_set_false:N \l_tmpa_bool }
1148     }{}{
1149       \seq_if_empty:NT \l_tmpa_seq {
1150         \bool_set_false:N \l_tmpa_bool
1151       }
1152     }
1153   }
1154
1155   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
1156   \str_if_empty:NTF \l_stex_modules_subpath_str {
1157     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
1158   }{
1159     \str_set:Nx \l_stex_modules_ns_str {
1160       \l_tmpa_str/\l_stex_modules_subpath_str
1161     }
1162   }
1163 }

```

(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
1164 \str_new:N \l_stex_modules_ns_str
1165 \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.

```

1166 \cs_new_protected:Nn \stex_modules_current_namespace: {
1167   \str_clear:N \l_stex_modules_subpath_str
1168   \prop_if_exist:NTF \l_stex_current_repository_prop {
1169     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1170     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1171   }{
1172     % split off file extension
1173     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1174     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1175     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1176     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1177     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1178     \str_set:Nx \l_stex_modules_ns_str {
1179       file:/\stex_path_to_string:N \l_tmpa_seq
1180     }
1181   }
1182 }

```

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

## 28.1 The module environment

module arguments:

```

1183 \keys_define:nn { stex / module } {
1184   title      .tl_set:N      = \smodulename ,
1185   type       .str_set_x:N   = \smodulename ,
1186   id         .str_set_x:N   = \smodulename ,
1187   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1188   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1189   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1190   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1191   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1192   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1193   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1194 }
1195
1196 \cs_new_protected:Nn \__stex_modules_args:n {
1197   \str_clear:N \smodulename
1198   \str_clear:N \smodulename
1199   \str_clear:N \smodulename
1200   \str_clear:N \l_stex_module_ns_str
1201   \str_clear:N \l_stex_module_lang_str
1202   \str_clear:N \l_stex_module_sig_str
1203   \str_clear:N \l_stex_module_creators_str

```

```

1204 \str_clear:N \l_stex_module_contributors_str
1205 \str_clear:N \l_stex_module_meta_str
1206 \str_clear:N \l_stex_module_srccite_str
1207 \keys_set:nn { stex / module } { #1 }
1208 }
1209
1210 % module parameters here? In the body?
1211

```

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

```

1212 \cs_new_protected:Nn \stex_module_setup:nn {
1213   \str_set:Nx \l_stex_module_name_str { #2 }
1214   \__stex_modules_args:n { #1 }

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

1215   \stex_if_in_module:TF {
1216     % Nested module
1217     \prop_get:cnN {c_stex_module\_l_stex_current_module_str _prop}
1218       { ns } \l_stex_module_ns_str
1219     \str_set:Nx \l_stex_module_name_str {
1220       \prop_item:cn {c_stex_module\_l_stex_current_module_str _prop}
1221         { name } / \l_stex_module_name_str
1222     }
1223   }{
1224     % not nested:
1225     \str_if_empty:NT \l_stex_module_ns_str {
1226       \stex_modules_current_namespace:
1227       \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1228       \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1229         / { \l_stex_module_ns_str }
1230       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1231       \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1232         \str_set:Nx \l_stex_module_ns_str {
1233           \stex_path_to_string:N \l_tmpa_seq
1234         }
1235       }
1236     }
1237   }

```

Next, we determine the language of the module:

```

1238 \str_if_empty:NT \l_stex_module_lang_str {
1239   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1240   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1241   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1242   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1243   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1244     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1245       inferred~from~file~name}
1246     \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1247   }
1248 }
1249
1250 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {

```

```

1251 \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1252 \l_tmpa_str {
1253   \ltx@ifpackageloaded{babel}{
1254     \exp_args:Nx \selectlanguage { \l_tmpa_str }
1255   }{}
1256 } {
1257   \msg_error:nxx{stex}{error/unknownlanguage}{\l_tmpa_str}
1258 }
1259 }}

```

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

```

1260 \str_if_empty:NTF \l_stex_module_sig_str {
1261   \exp_args:Nnx \prop_gset_from_keyval:cn {
1262     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1263   } {
1264     name      = \l_stex_module_name_str ,
1265     ns        = \l_stex_module_ns_str ,
1266     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1267     lang      = \l_stex_module_lang_str ,
1268     sig       = \l_stex_module_sig_str ,
1269     meta      = \l_stex_module_meta_str
1270   }
1271   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1272   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _fields}
1273   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1274   \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1275   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1276 \str_if_empty:NT \l_stex_module_meta_str {
1277   \str_set:Nx \l_stex_module_meta_str {
1278     \c_stex_metatheory_ns_str ? Metatheory
1279   }
1280 }
1281 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1282   \bool_set_true:N \l_stex_in_meta_bool
1283   \exp_args:Nx \stex_add_to_current_module:n {
1284     \bool_set_true:N \l_stex_in_meta_bool
1285     \stex_activate_module:n {\l_stex_module_meta_str}
1286     \bool_set_false:N \l_stex_in_meta_bool
1287   }
1288   \stex_activate_module:n {\l_stex_module_meta_str}
1289   \bool_set_false:N \l_stex_in_meta_bool
1290 }
1291 }{
1292   \str_if_empty:NT \l_stex_module_lang_str {
1293     \msg_error:nxxx{stex}{error/siglanguage}{
1294       \l_stex_module_ns_str?\l_stex_module_name_str
1295     }{\l_stex_module_sig_str}
1296   }
1297
1298   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1299   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str

```

```

1300 \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1301 \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1302 \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1303 \str_set:Nx \l_tmpa_str {
1304   \stex_path_to_string:N \l_tmpa_seq /
1305   \l_tmpa_str . \l_stex_module_sig_str .tex
1306 }
1307 \IfFileExists \l_tmpa_str {
1308   \exp_args:No \stex_in_smsmode:nn { \l_tmpa_str } {
1309     \str_clear:N \l_stex_current_module_str
1310     \seq_clear:N \l_stex_all_modules_seq
1311     \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1312     \input { \l_tmpa_str }
1313   }
1314 }{
1315   \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1316 }
1317 \stex_if_smsmode:F {
1318   \stex_activate_module:n {
1319     \l_stex_module_ns_str ? \l_stex_module_name_str
1320   }
1321 }
1322 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1323 }
1324 }

```

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

```

```

1348 %      { \l_stex_module_ns_str ? \l_stex_module_name_str }
1349
1350
1351 \stex_if_smsmode:TF {
1352   \stex_smsmode_set_codes:
1353 } {
1354   \begin{stex_annotate_env} {theory} {
1355     \l_stex_module_ns_str ? \l_stex_module_name_str
1356   }
1357
1358   \stex_annotate_invisible:nnn{header}{} {
1359     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1360     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1361     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1362       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1363     }
1364     \str_if_empty:NF \smodulotype {
1365       \stex_annotate:nnn{type}{\smodulotype}{}
1366     }
1367   }
1368 }
1369 \int_set:Nn \l_stex_module_group_depth_int {\currentgrouplevel}
1370 % TODO: Inherit metatheory for nested modules?
1371 }
1372 \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}

1373 \cs_new_protected:Nn \_stex_modules_end_module: {
1374 % \str_set:Nx \l_tmpa_str {
1375 %   c_stex_module_
1376 %   \prop_item:Nn \l_stex_current_module_prop { ns } ?
1377 %   \prop_item:Nn \l_stex_current_module_prop { name }
1378 %   _prop
1379 % }
1380 %^^A \prop_new:c { \l_tmpa_str }
1381 % \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
1382 \stex_debug:nn{modules}{Closing module~\prop_item:cn {c_stex_module_\l_stex_current_module_
1383 }

```

(End definition for `\_stex_modules_end_module:.`)

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

```

1384 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1385 \NewDocumentEnvironment { smodule } { 0 } { m } {
1386   \stex_module_setup:nn{#1}{#2}
1387   \par
1388   \stex_if_smsmode:F{
1389     \tl_clear:N \l_tmpa_tl
1390     \clist_map_inline:Nn \smodulotype {
1391       \tl_if_exist:cT {\_stex_modules_smodule_##1_start:}{
1392         \tl_set:Nn \l_tmpa_tl {\use:c{\_stex_modules_smodule_##1_start:}}
1393       }

```

```

1394     }
1395     \tl_if_empty:NTF \l_tmpa_tl {
1396         \__stex_modules_smodule_start:
1397     }{
1398         \l_tmpa_tl
1399     }
1400 }
1401 \__stex_modules_begin_module:
1402 \stex_ref_new_doc_target:n \smoduleid
1403 } {
1404     \__stex_modules_end_module:
1405     \stex_if_smsmode:TF {
1406         % \exp_args:Nx \stex_add_to_sms:n {
1407         %     \prop_gset_from_keyval:cn {
1408         %         c_stex_module_
1409         %         \prop_item:Nn \l_stex_current_module_prop { ns } ?
1410         %         \prop_item:Nn \l_stex_current_module_prop { name }
1411         %         _prop
1412         %     } {
1413         %         name      = \prop_item:cn { \l_tmpa_str } { name } ,
1414         %         ns       = \prop_item:cn { \l_tmpa_str } { ns } ,
1415         %         file     = \prop_item:cn { \l_tmpa_str } { file } ,
1416         %         lang     = \prop_item:cn { \l_tmpa_str } { lang } ,
1417         %         sig      = \prop_item:cn { \l_tmpa_str } { sig } ,
1418         %         meta     = \prop_item:cn { \l_tmpa_str } { meta }
1419         %     }
1420         % }
1421     }{
1422         \end{stex_annotate_env}
1423         \clist_set:Nn \l_tmpa_clist \smoduletype
1424         \tl_clear:N \l_tmpa_tl
1425         \clist_map_inline:Nn \l_tmpa_clist {
1426             \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1427                 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1428             }
1429         }
1430         \tl_if_empty:NTF \l_tmpa_tl {
1431             \__stex_modules_smodule_end:
1432         }{
1433             \l_tmpa_tl
1434         }
1435     }
1436 }
1437
1438 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1439 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1440
1441 \newcommand\stexpatchmodule[3] [] {
1442     \str_set:Nx \l_tmpa_str{ #1 }
1443     \str_if_empty:NTF \l_tmpa_str {
1444         \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1445         \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1446     }{
1447         \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }

```



```

1448     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1449   }
1450 }
1451

```

## 28.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n
1452 \NewDocumentCommand \STEXModule { m } {
1453   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1454   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1455   \tl_set:Nn \l_tmpa_tl {
1456     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1457   }
1458   \seq_map_inline:Nn \l_stex_all_modules_seq {
1459     \str_set:Nn \l_tmpb_str { ##1 }
1460     \str_if_eq:eeT { \l_tmpa_str } {
1461       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1462     } {
1463       \seq_map_break:n {
1464         \tl_set:Nn \l_tmpa_tl {
1465           \stex_invoke_module:n { ##1 }
1466         }
1467       }
1468     }
1469   }
1470   \l_tmpa_tl
1471 }
1472
1473 \cs_new_protected:Nn \stex_invoke_module:n {
1474   \stex_debug:nn{modules}{Invoking~module~#1}
1475   \peek_charcode_remove:NTF ! {
1476     \__stex_modules_invoke_uri:nN { #1 }
1477   } {
1478     \peek_charcode_remove:NTF ? {
1479       \__stex_modules_invoke_symbol:nn { #1 }
1480     } {
1481       \msg_error:nnx{stex}{error/syntax}{
1482         ?~or~!~expected~after~
1483         \c_backslash_str STEXModule{#1}
1484       }
1485     }
1486   }
1487 }
1488
1489 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1490   \str_set:Nn #2 { #1 }
1491 }
1492
1493 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1494   \stex_invoke_symbol:n{#1?#2}
1495 }

```

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

`\stex_activate_module:n`

```

1496 \bool_new:N \l_stex_in_meta_bool
1497 \bool_set_false:N \l_stex_in_meta_bool
1498 \cs_new_protected:Nn \stex_activate_module:n {
1499   \stex_debug:nn{modules}{Activating~module~#1}
1500   \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1501     \msg_error:nnn{stex}{error/conflictingmodules}{ #1 }
1502   }
1503   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1504     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1505     \use:c{ c_stex_module_#1_code }
1506   }
1507 }

```

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

```

1508 </package>

```

## Chapter 29

# STEX -Module Inheritance Implementation

```
1509 <*package>
1510
1511 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1512
```

### 29.1 SMS Mode

```
1513 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1514 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1515 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1516 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1517
1518 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1519   \makeatletter
1520   \makeatother
1521   \ExplSyntaxOn
1522   \ExplSyntaxOff
1523   \rustexBREAK
1524 }
1525
1526 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1527   \symdef
1528   \importmodule
1529   \notation
1530   \symdecl
1531   \STEXexport
1532   \inlineass
1533   \inlinedef
1534   \inlineex
1535 }
1536
```

```

1537 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1538   \tl_to_str:n {
1539     smodule,
1540     copymodule,
1541     interpretmodule
1542   %   sdefinition,
1543   %   sexample,
1544   %   sassertion,
1545   %   sparagraph
1546 }
1547 }

```

(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`

```

1548 \bool_new:N \g__stex_smsmode_bool
1549 \bool_set_false:N \g__stex_smsmode_bool
1550 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1551   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1552 }

```

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

`\_stex_smsmode_if_catcodes_p:`

Checks whether the SMS mode category code scheme is active.

`\_stex_smsmode_if_catcodes:TF`

```

1553 \bool_new:N \g__stex_smsmode_catcode_bool
1554 \bool_set_false:N \g__stex_smsmode_catcode_bool
1555 \prg_new_conditional:Nnn \_stex_smsmode_if_catcodes: { p, T, F, TF } {
1556   \bool_if:NTF \g__stex_smsmode_catcode_bool
1557   \prg_return_true: \prg_return_false:
1558 }

```

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

`\stex_smsmode_set_codes:`

```

1559 \cs_new_protected:Nn \stex_smsmode_set_codes: {
1560   \stex_if_smsmode:T {
1561     \_stex_smsmode_if_catcodes:F {
1562       \bool_gset_true:N \g__stex_smsmode_catcode_bool
1563       \exp_after:wN \char_gset_active_eq:NN
1564       \c_backslash_str \_stex_smsmode_cs:
1565       \tex_global:D \char_set_catcode_active:N \
1566       \tex_global:D \char_set_catcode_other:N $
1567       \tex_global:D \char_set_catcode_other:N ^
1568       \tex_global:D \char_set_catcode_other:N _
1569       \tex_global:D \char_set_catcode_other:N &
1570       \tex_global:D \char_set_catcode_other:N ##
1571     }
1572   }
1573 } \iffalse $ \fi % to make syntax highlighting work again

```

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

`\__stex_smsmode_unset_codes:` Sets category code scheme back from the one used in SMS mode.

```

1574 \cs_new_protected:Nn \__stex_smsmode_unset_codes: {
1575   \__stex_smsmode_if_catcodes:T {
1576     \bool_gset_false:N \g__stex_smsmode_catcode_bool
1577     \exp_after:wN \tex_global:D \exp_after:wN
1578     \char_set_catcode_escape:N \c_backslash_str
1579     \tex_global:D \char_set_catcode_math_toggle:N $
1580     \tex_global:D \char_set_catcode_math_superscript:N ^
1581     \tex_global:D \char_set_catcode_math_subscript:N _
1582     \tex_global:D \char_set_catcode_alignment:N &
1583     \tex_global:D \char_set_catcode_parameter:N ##
1584   }
1585 } \iffalse $ \fi % to make syntax highlighting work again

```

(End definition for `\__stex_smsmode_unset_codes:`.)

`\stex_in_smsmode:nn`

```

1586 \cs_new_protected:Nn \stex_in_smsmode:nn {
1587   \vbox_set:Nn \l_tmpa_box {
1588     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1589     \bool_gset_true:N \g__stex_smsmode_bool
1590     \stex_smsmode_set_codes:
1591     #2
1592     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1593     \stex_if_smsmode:F {
1594       \__stex_smsmode_unset_codes:
1595     }
1596   }
1597   \box_clear:N \l_tmpa_box
1598 }

```

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

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

```

1599 \cs_new_protected:Nn \__stex_smsmode_cs: {
1600   \str_clear:N \l_tmpa_str
1601   \peek_analysis_map_inline:n {
1602     % #1: token (one expansion)
1603     % #2: charcode
1604     % #3 catcode
1605     \token_if_eq_charcode:NNTF ##3 B {
1606       % token is a letter
1607       \exp_args:NNo \str_put_right:Nn \l_tmpa_str { ##1 }
1608     } {
1609       \str_if_empty:NTF \l_tmpa_str {
1610         % we don't allow (or need) single non-letter CSs
1611         % for now
1612         \peek_analysis_map_break:
1613       }{
1614         \str_if_eq:onTF \l_tmpa_str { begin } {
1615           \peek_analysis_map_break:n {
1616             \exp_after:wN \__stex_smsmode_checkbegin:n ##1
1617           }

```

```

1618     } {
1619         \str_if_eq:onTF \l_tmpa_str { end } {
1620             \peek_analysis_map_break:n {
1621                 \exp_after:wN \__stex_smsmode_checkend:n ##1
1622             }
1623         } {
1624             \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
1625             \exp_args:NNNo \exp_args:NNo \tl_if_in:NnTF
1626             \g_stex_smsmode_allowedmacros_tl
1627             { \use:c{\l_tmpa_str} } {
1628                 \stex_debug:nn{modules}{Executing~1:~\l_tmpa_str}
1629                 \peek_analysis_map_break:n {
1630                     \exp_after:wN \l_tmpa_tl ##1
1631                 }
1632             } {
1633                 \exp_args:NNNo \exp_args:NNo \tl_if_in:NnTF
1634                 \g_stex_smsmode_allowedmacros_escape_tl
1635                 { \use:c{\l_tmpa_str} } {
1636                     \__stex_smsmode_unset_codes:
1637                     \stex_debug:nn{modules}{Executing~2:~\l_tmpa_str}
1638                     % TODO \__stex_smsmode_rescan_cs:
1639                     \int_compare:nNnTF {##2} = {92} {
1640                         \peek_analysis_map_break:n {
1641                             \__stex_smsmode_unset_codes:
1642                             \__stex_smsmode_rescan_cs:
1643                         }
1644                     } {
1645                         \peek_analysis_map_break:n {
1646                             \exp_after:wN \l_tmpa_tl ##1
1647                         }
1648                     }
1649                 } {
1650                     \int_compare:nNnTF {##2} = {92} {
1651                         \peek_analysis_map_break:n { \__stex_smsmode_cs: }
1652                     } {
1653                         \peek_analysis_map_break:n { \exp_after:wN\relax ##1 }
1654                     }
1655                 }
1656             }
1657         }
1658     }
1659 }
1660 }
1661 }
1662 }

```

(End definition for \\_\_stex\_smsmode\_cs:.)

\\_\_stex\_smsmode\_rescan\_cs: If the last token gobbled by \stex\_smsmode\_cs: happened to be a \, we need to rescan the cs name and reinsert it into the input stream:

```

1663 \cs_new_protected:Nn \__stex_smsmode_rescan_cs: {
1664     \str_clear:N \l_tmppb_str
1665     \peek_analysis_map_inline:n {
1666         \token_if_eq_charcode:NNTF ##3 B {

```

```

1667         % token is a letter
1668         \exp_args:NNo \str_put_right:Nn \l_tmpb_str { ##1 }
1669     } {
1670         \peek_analysis_map_break:n {
1671             \exp_after:wN \use:c \exp_after:wN {
1672                 \exp_after:wN \l_tmpa_str\exp_after:wN
1673             } \use:c { \l_tmpb_str \exp_after:wN } ##1
1674         }
1675     }
1676 }
1677 }

```

(End definition for `\__stex_smsmode_rescan_cs:.`)

`\__stex_smsmode_checkbegin:n` called on `\begin`; checks whether the environment being opened is allowed in SMS mode.

```

1678 \cs_new_protected:Nn \__stex_smsmode_checkbegin:n {
1679     \str_set:Nn \l_tmpa_str { #1 }
1680     \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1681         \__stex_smsmode_unset_codes:
1682         \begin{#1}
1683     }
1684 }

```

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

`\__stex_smsmode_checkend:n` called on `\end`; checks whether the environment being opened is allowed in SMS mode.

```

1685 \cs_new_protected:Nn \__stex_smsmode_checkend:n {
1686     \str_set:Nn \l_tmpa_str { #1 }
1687     \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1688         \end{#1}
1689     }
1690 }

```

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

## 29.2 Inheritance

```

1691 <@@=stex_importmodule>

```

`\stex_import_module_uri:nn`

```

1692 \cs_new_protected:Nn \stex_import_module_uri:nn {
1693     \str_set:Nx \l_stex_import_archive_str { #1 }
1694     \str_set:Nn \l_stex_import_path_str { #2 }
1695
1696     \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1697     \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1698     \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1699
1700     \stex_modules_current_namespace:
1701     \bool_lazy_all:nTF {
1702         {\str_if_empty_p:N \l_stex_import_archive_str}
1703         {\str_if_empty_p:N \l_stex_import_path_str}
1704         {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1705     }{

```

```

1706     \str_set_eq:NN \l_stex_import_path_str \l_stex_modules_subpath_str
1707     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1708   }{
1709     \str_if_empty:NT \l_stex_import_archive_str {
1710       \prop_if_exist:NT \l_stex_current_repository_prop {
1711         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1712       }
1713     }
1714     \str_if_empty:NTF \l_stex_import_archive_str {
1715       \str_if_empty:NF \l_stex_import_path_str {
1716         \str_set:Nx \l_stex_import_ns_str {
1717           \l_stex_module_ns_str / \l_stex_import_path_str
1718         }
1719       }
1720     }{
1721       \stex_require_repository:n \l_stex_import_archive_str
1722       \prop_get:cnN { c_stex_mathhub\_l_stex_import_archive_str_manifest_prop } { ns }
1723       \l_stex_import_ns_str
1724       \str_if_empty:NF \l_stex_import_path_str {
1725         \str_set:Nx \l_stex_import_ns_str {
1726           \l_stex_import_ns_str / \l_stex_import_path_str
1727         }
1728       }
1729     }
1730   }
1731 }

```

(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 1732 \str_new:N \l_stex_import_name_str
\l_stex_import_path_str    1733 \str_new:N \l_stex_import_archive_str
\l_stex_import_ns_str      1734 \str_new:N \l_stex_import_path_str
                           1735 \str_new:N \l_stex_import_ns_str

```

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

```

\stex_import_require_module:nnnnn {<ns>} {<archive-ID>} {<path>} {<name>}}
1736 \cs_new_protected:Nn \stex_import_require_module:nnnnn {
1737   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1738
1739     % archive
1740     \str_set:Nx \l_tmpa_str { #2 }
1741     \str_if_empty:NTF \l_tmpa_str {
1742       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1743     } {
1744       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1745       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1746       \seq_put_right:Nn \l_tmpa_seq { source }
1747     }
1748
1749     % path
1750     \str_set:Nx \l_tmpb_str { #3 }
1751     \str_if_empty:NTF \l_tmpb_str {

```



```

1752 \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1753
1754 \ltx@ifpackageloaded{babel} {
1755   \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1756     { \language } \l_tmpb_str {
1757     \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1758   }
1759 } {
1760   \str_clear:N \l_tmpb_str
1761 }
1762
1763 \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1764 \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1765   \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1766 }{
1767   \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1768   \IfFileExists{ \l_tmpa_str.tex }{
1769     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1770   }{
1771     % try english as default
1772     \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1773     \IfFileExists{ \l_tmpa_str.en.tex }{
1774       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1775     }{
1776       \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1777     }
1778   }
1779 }
1780
1781 } {
1782   \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1783   \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1784
1785   \ltx@ifpackageloaded{babel} {
1786     \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1787       { \language } \l_tmpb_str {
1788       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1789     }
1790   } {
1791     \str_clear:N \l_tmpb_str
1792   }
1793
1794   \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1795
1796   \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1797   \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1798     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1799   }{
1800     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1801     \IfFileExists{ \l_tmpa_str/#4.tex }{
1802       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1803     }{
1804       % try english as default
1805       \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}

```

```

1806 \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1807   \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1808 }{
1809   \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1810   \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1811     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1812   }{
1813     \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1814     \IfFileExists{ \l_tmpa_str.tex }{
1815       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1816     }{
1817       % try english as default
1818       \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1819       \IfFileExists{ \l_tmpa_str.en.tex }{
1820         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1821       }{
1822         \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1823       }
1824     }
1825   }
1826 }
1827 }
1828 }
1829 }
1830
1831 \exp_args:No \stex_in_smsmode:nn { \g__stex_importmodule_file_str } {
1832   \seq_clear:N \l_stex_all_modules_seq
1833   \str_clear:N \l_stex_current_module_str
1834   \str_set:Nx \l_tmpb_str { #2 }
1835   \str_if_empty:NF \l_tmpb_str {
1836     \stex_set_current_repository:n { #2 }
1837   }
1838   \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1839   \input { \g__stex_importmodule_file_str }
1840 }
1841
1842 \stex_if_module_exists:nF { #1 ? #4 } {
1843   \msg_error:nnx{stex}{error/unknownmodule}{
1844     #1?#4~(in~file~\g__stex_importmodule_file_str)
1845   }
1846 }
1847 }
1848 \stex_activate_module:n { #1 ? #4 }
1849 }

```

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

## `\importmodule`

```

1850 \NewDocumentCommand \importmodule { 0{} m } {
1851   \stex_import_module_uri:nn { #1 } { #2 }
1852   \stex_debug:nn{modules}{Importing~module:~
1853     \l_stex_import_ns_str ? \l_stex_import_name_str
1854   }
1855   \stex_if_smsmode:F {

```

```

1856 \stex_import_require_module:nnnn
1857 { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1858 { \l_stex_import_path_str } { \l_stex_import_name_str }
1859 \stex_annotate_invisible:nnn
1860 {import} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1861 }
1862 \exp_args:Nx \stex_add_to_current_module:n {
1863 \stex_import_require_module:nnnn
1864 { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1865 { \l_stex_import_path_str } { \l_stex_import_name_str }
1866 }
1867 \exp_args:Nx \stex_add_import_to_current_module:n {
1868 \l_stex_import_ns_str ? \l_stex_import_name_str
1869 }
1870 \stex_smsmode_set_codes:
1871 }
1872 \stex_deactivate_macro:Nn \importmodule {module-environments}

```

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

#### `\usemodule`

```

1873 \NewDocumentCommand \usemodule { 0{} m } {
1874 \stex_if_smsmode:F {
1875 \stex_import_module_uri:nn { #1 } { #2 }
1876 \stex_import_require_module:nnnn
1877 { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1878 { \l_stex_import_path_str } { \l_stex_import_name_str }
1879 \stex_annotate_invisible:nnn
1880 {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1881 }
1882 \stex_smsmode_set_codes:
1883 }

```

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

```

1884 \endpackage

```

## Chapter 30

# STEX -Symbols Implementation

```
1885 <*package>
1886
1887 %%%%%%%%%% symbols.dtx %%%%%%%%%%
1888
```

Warnings and error messages

```
1889
```

### 30.1 Symbol Declarations

```
1890 <@@=stex_symdecl>
```

`\l_stex_all_symbols_seq` Stores all available symbols

```
1891 \seq_new:N \l_stex_all_symbols_seq
```

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

`\STEXsymbol`

```
1892 \NewDocumentCommand \STEXsymbol { m } {
1893   \stex_get_symbol:n { #1 }
1894   \exp_args:No
1895   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1896 }
```

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

symdecl arguments:

```
1897 \keys_define:nn { stex / symdecl } {
1898   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1899   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1900   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1901   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
1902   align     .str_set:N    = \l_stex_symdecl_align_str , % TODO(?)
1903   gfc       .str_set:N    = \l_stex_symdecl_gfc_str , % TODO(?)
1904   specializes .str_set:N  = \l_stex_symdecl_specializes_str , % TODO(?)
1905   def       .tl_set:N    = \l_stex_symdecl_definiens_tl
1906 }
```

```

1907
1908 \bool_new:N \l_stex_symdecl_make_macro_bool
1909
1910 \cs_new_protected:Nn \__stex_symdecl_args:n {
1911   \str_clear:N \l_stex_symdecl_name_str
1912   \str_clear:N \l_stex_symdecl_args_str
1913   \bool_set_false:N \l_stex_symdecl_local_bool
1914   \tl_clear:N \l_stex_symdecl_type_tl
1915   \tl_clear:N \l_stex_symdecl_definiens_tl
1916
1917   \keys_set:nn { stex / symdecl } { #1 }
1918 }

```

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

```

1919
1920 \NewDocumentCommand \symdecl { s O{} m } {
1921   \__stex_symdecl_args:n { #2 }
1922   \IfBooleanTF #1 {
1923     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1924   } {
1925     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1926   }
1927   \stex_symdecl_do:n { #3 }
1928   \stex_smsmode_set_codes:
1929 }
1930 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

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

```

1931 \cs_new_protected:Nn \stex_symdecl_do:n {
1932   \stex_if_in_module:F {
1933     % TODO throw error? some default namespace?
1934   }
1935
1936   \str_if_empty:NT \l_stex_symdecl_name_str {
1937     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1938   }
1939
1940   \prop_if_exist:cT { l_stex_symdecl_
1941     \l_stex_current_module_str ?
1942     \l_stex_symdecl_name_str
1943     _prop
1944   }{
1945     % TODO throw error (beware of circular dependencies)
1946   }
1947
1948   \prop_clear:N \l_tmpa_prop
1949   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
1950   \seq_clear:N \l_tmpa_seq
1951   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1952   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1953

```

```

1954 \exp_args:No \stex_add_constant_to_current_module:n {
1955   \l_stex_symdecl_name_str
1956 }
1957
1958 % arity/args
1959 \int_zero:N \l_tmpb_int
1960
1961 \bool_set_true:N \l_tmpa_bool
1962 \str_map_inline:Nn \l_stex_symdecl_args_str {
1963   \token_case_meaning:NnF ##1 {
1964     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1965     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1966     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1967     {\tl_to_str:n a} {
1968       \bool_set_false:N \l_tmpa_bool
1969       \int_incr:N \l_tmpb_int
1970     }
1971     {\tl_to_str:n B} {
1972       \bool_set_false:N \l_tmpa_bool
1973       \int_incr:N \l_tmpb_int
1974     }
1975   }{
1976     \msg_set:nnn{stex}{error/wrongargs}{
1977       args~value~in~symbol~declaration~for~
1978       \l_stex_current_module_str ?
1979       \l_stex_symdecl_name_str ~
1980       needs~to~be~
1981       i,~a,~b~or~B,~but~##1~given
1982     }
1983     \msg_error:nn{stex}{error/wrongargs}
1984   }
1985 }
1986 \bool_if:NTF \l_tmpa_bool {
1987   % possibly numeric
1988   \str_if_empty:NTF \l_stex_symdecl_args_str {
1989     \prop_put:Nnn \l_tmpa_prop { args } {}
1990     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1991   }{
1992     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1993     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1994     \str_clear:N \l_tmpa_str
1995     \int_step_inline:nn \l_tmpa_int {
1996       \str_put_right:Nn \l_tmpa_str i
1997     }
1998     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1999   }
2000 } {
2001   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2002   \prop_put:Nnx \l_tmpa_prop { arity }
2003     { \str_count:N \l_stex_symdecl_args_str }
2004 }
2005 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
2006
2007

```

```

2008 % semantic macro
2009
2010 \bool_if:NT \l_stex_symdecl_make_macro_bool {
2011   \exp_args:Nx \stex_do_aftergroup:n {
2012     \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2013       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2014     }}
2015   }
2016
2017   \bool_if:NF \l_stex_symdecl_local_bool {
2018     \exp_args:Nx \stex_add_to_current_module:n {
2019       \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2020         \l_stex_current_module_str ? \l_stex_symdecl_name_str
2021       } }
2022     }
2023   }
2024 }
2025
2026 % add to all symbols
2027
2028 \bool_if:NF \l_stex_symdecl_local_bool {
2029   \exp_args:Nx \stex_add_to_current_module:n {
2030     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2031       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2032     }
2033   }
2034 %   \exp_args:Nx \stex_add_field_to_current_module:n {
2035 %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2036 %   }
2037 }
2038
2039 \stex_debug:nn{symbols}{New~symbol:~
2040   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2041   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2042   Args:~\prop_item:Nn \l_tmpa_prop { args }
2043 }
2044
2045 % circular dependencies require this:
2046
2047 \prop_if_exist:cF {
2048   l_stex_symdecl_
2049   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2050   _prop
2051 } {
2052   \prop_set_eq:cN {
2053     l_stex_symdecl_
2054     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2055     _prop
2056   } \l_tmpa_prop
2057 }
2058
2059 \seq_clear:c {
2060   l_stex_symdecl_
2061   \l_stex_current_module_str ? \l_stex_symdecl_name_str

```

```

2062   _notations
2063 }
2064
2065 \bool_if:NF \l_stex_symdecl_local_bool {
2066   \exp_args:Nx
2067   \stex_add_to_current_module:n {
2068     \seq_clear:c {
2069       l_stex_symdecl_
2070       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2071       _notations
2072     }
2073     \prop_set_from_keyval:cn {
2074       l_stex_symdecl_
2075       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2076       _prop
2077     } {
2078       name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2079       module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2080       type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2081       args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2082       arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2083       assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
2084     }
2085   }
2086 }
2087
2088 \stex_if_smsmode:TF {
2089   \bool_if:NF \l_stex_symdecl_local_bool {
2090     % \exp_args:Nx \stex_add_to_sms:n {
2091     %   \prop_set_from_keyval:cn {
2092     %     l_stex_symdecl_
2093     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2094     %     _prop
2095     %   } {
2096     %     name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2097     %     module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2098     %     local     = \prop_item:Nn \l_tmpa_prop { local }     ,
2099     %     type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2100     %     args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2101     %     arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2102     %     assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
2103     %   }
2104     %   \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2105     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2106     %   }
2107     % }
2108   }
2109 }{
2110   \exp_args:Nx \stex_do_aftergroup:n {
2111     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2112       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2113     }
2114   }
2115   \stex_if_do_html:T {

```



```

2116 \stex_annotate_invisible:nnn {symdecl} {
2117   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2118 } {
2119   \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}{}}{ $\l_st
2120   \stex_annotate_invisible:nnn{args}{}{
2121     \prop_item:Nn \l_tmpa_prop { args }
2122   }
2123   \stex_annotate_invisible:nnn{macroname}{#1}{}
2124   \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2125     \stex_annotate_invisible:nnn{definiens}{}
2126     { $\l_stex_symdecl_definiens_tl$ }
2127   }
2128 }
2129 }
2130 }
2131 }

```

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

`\stex_get_symbol:n`

```

2132 \str_new:N \l_stex_get_symbol_uri_str
2133
2134 \cs_new_protected:Nn \stex_get_symbol:n {
2135   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2136     \__stex_symdecl_get_symbol_from_cs:n { #1 }
2137   }{
2138     % argument is a string
2139     % is it a command name?
2140     \cs_if_exist:cTF { #1 }{
2141       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2142       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2143       \str_if_empty:NNTF \l_tmpa_str {
2144         \exp_args:Nx \cs_if_eq:NNTF {
2145           \tl_head:N \l_tmpa_tl
2146         } \stex_invoke_symbol:n {
2147           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
2148         }{
2149           \__stex_symdecl_get_symbol_from_string:n { #1 }
2150         }
2151       } {
2152         \__stex_symdecl_get_symbol_from_string:n { #1 }
2153       }
2154     }{
2155       % argument is not a command name
2156       \__stex_symdecl_get_symbol_from_string:n { #1 }
2157       % \l_stex_all_symbols_seq
2158     }
2159   }
2160 }
2161
2162 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2163   \str_set:Nn \l_tmpa_str { #1 }
2164   \bool_set_false:N \l_tmpa_bool
2165   \stex_if_in_module:T {

```

```

2166 \exp_args:Nno \seq_if_in:cnT {c_stex_module_\l_stex_current_module_str _constants} { \l_
2167 \bool_set_true:N \l_tmpa_bool
2168 \str_set:Nx \l_stex_get_symbol_uri_str {
2169 \l_stex_current_module_str ? #1
2170 }
2171 }
2172 }
2173 \bool_if:NF \l_tmpa_bool {
2174 \tl_set:Nn \l_tmpa_tl {
2175 \msg_set:nnn{stex}{error/unknownsymbol}{
2176 No~symbol~#1~found!
2177 }
2178 \msg_error:nn{stex}{error/unknownsymbol}
2179 }
2180 \str_set:Nn \l_tmpa_str { #1 }
2181 \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2182 \seq_map_inline:Nn \l_stex_all_symbols_seq {
2183 \str_set:Nn \l_tmpb_str { ##1 }
2184 \str_if_eq:eeT { \l_tmpa_str } {
2185 \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
2186 } {
2187 \seq_map_break:n {
2188 \tl_set:Nn \l_tmpa_tl {
2189 \str_set:Nn \l_stex_get_symbol_uri_str {
2190 ##1
2191 }
2192 }
2193 }
2194 }
2195 }
2196 \l_tmpa_tl
2197 }
2198 }
2199
2200 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
2201 \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2202 { \tl_tail:N \l_tmpa_tl }
2203 \tl_if_single:NTF \l_tmpa_tl {
2204 \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2205 \exp_after:wN \str_set:Nn \exp_after:wN
2206 \l_stex_get_symbol_uri_str \l_tmpa_tl
2207 }{
2208 % TODO
2209 % tail is not a single group
2210 }
2211 }{
2212 % TODO
2213 % tail is not a single group
2214 }
2215 }

```

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

## 30.2 Notations

```

2216 <@@=stex_notation>
      notation arguments:
2217 \keys_define:nn { stex / notation } {
2218   lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2219   variant   .tl_set_x:N = \l__stex_notation_variant_str ,
2220   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2221   op        .tl_set:N   = \l__stex_notation_op_tl ,
2222   primary   .bool_set:N = \l__stex_notation_primary_bool ,
2223   primary   .default:n  = {true} ,
2224   unknown   .code:n     = \str_set:Nx
2225               \l__stex_notation_variant_str \l_keys_key_str
2226 }
2227
2228 \cs_new_protected:Nn \stex_notation_args:n {
2229   \str_clear:N \l__stex_notation_lang_str
2230   \str_clear:N \l__stex_notation_variant_str
2231   \str_clear:N \l__stex_notation_prec_str
2232   \tl_clear:N \l__stex_notation_op_tl
2233   \bool_set_false:N \l__stex_notation_primary_bool
2234
2235   \keys_set:nn { stex / notation } { #1 }
2236 }

```

**\notation**

```

2237 \NewDocumentCommand \notation { 0{} m } {
2238   \stex_notation_args:n { #1 }
2239   \tl_clear:N \l_stex_symdecl_definiens_tl
2240   \stex_get_symbol:n { #2 }
2241   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
2242 }
2243 \stex_deactivate_macro:Nn \notation {module-environments}

```

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

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

```

2244 \cs_new_protected:Nn \stex_notation_do:nn {
2245   \let\l_stex_current_symbol_str\relax
2246   \prop_set_eq:Nc \l_tmpa_prop {
2247     l_stex_symdecl_ #1 _prop
2248   }
2249
2250   \prop_clear:N \l_tmpb_prop
2251   \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
2252   \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
2253   \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
2254
2255   % precedences
2256   \seq_clear:N \l_tmpb_seq
2257   \exp_args:NNno
2258   \str_if_empty:NTF \l__stex_notation_prec_str {
2259     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2260     \int_compare:nNnTF \l_tmpa_str = 0 {

```

```

2261     \exp_args:NNx
2262     \prop_put:Nno \l_tmpb_prop { opprec }
2263     { \neginfprec }
2264   }{
2265     \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
2266   }
2267 } {
2268   \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2269     \exp_args:NNx
2270     \prop_put:Nno \l_tmpb_prop { opprec }
2271     { \neginfprec }
2272     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2273     \int_step_inline:nn { \l_tmpa_str } {
2274       \exp_args:NNx
2275       \seq_put_right:Nn \l_tmpb_seq { \infprec }
2276     }
2277   }{
2278     \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2279     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2280       \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
2281       \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2282         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2283         \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2284         \seq_map_inline:Nn \l_tmpa_seq {
2285           \seq_put_right:Nn \l_tmpb_seq { ##1 }
2286         }
2287       }
2288       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2289     }{
2290       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2291       \int_compare:nNnTF \l_tmpa_str = 0 {
2292         \exp_args:NNx
2293         \prop_put:Nno \l_tmpb_prop { opprec }
2294         { \infprec }
2295       }{
2296         \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
2297       }
2298     }
2299   }
2300 }
2301
2302 \seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
2303 \int_step_inline:nn { \l_tmpa_str } {
2304   \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
2305     \exp_args:NNx
2306     \seq_put_right:Nn \l_tmpb_seq {
2307       \prop_item:Nn \l_tmpb_prop { opprec }
2308     }
2309   }
2310 }
2311
2312 \prop_put:Nno \l_tmpb_prop { argprecs } \l_tmpb_seq
2313 \tl_clear:N \l_tmpa_tl
2314

```

```

2315 \int_compare:nNnTF \l_tmpa_str = 0 {
2316   \exp_args:NNe
2317   \cs_set:Npn \l__stex_notation_macrocode_cs {
2318     \_stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2319     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2320     { \prop_item:Nn \l_tmpb_prop { opprec } }
2321     { \exp_not:n { #2 } }
2322   }
2323   \__stex_notation_final:
2324 }{
2325   \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
2326   \str_if_in:NnTF \l_tmpb_str b {
2327     \exp_args:Nne \use:nn
2328     {
2329       \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2330       \cs_set:Npn \l_tmpa_str } { {
2331         \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2332         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2333         { \prop_item:Nn \l_tmpb_prop { opprec } }
2334         { \exp_not:n { #2 } }
2335       } }
2336   }{
2337     \str_if_in:NnTF \l_tmpb_str B {
2338       \exp_args:Nne \use:nn
2339       {
2340         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2341         \cs_set:Npn \l_tmpa_str } { {
2342           \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2343           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2344           { \prop_item:Nn \l_tmpb_prop { opprec } }
2345           { \exp_not:n { #2 } }
2346         } }
2347     }{
2348       \exp_args:Nne \use:nn
2349       {
2350         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2351         \cs_set:Npn \l_tmpa_str } { {
2352           \_stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2353           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2354           { \prop_item:Nn \l_tmpb_prop { opprec } }
2355           { \exp_not:n { #2 } }
2356         } }
2357     }
2358   }
2359
2360   \int_zero:N \l_tmpa_int
2361   \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2362   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2363   \__stex_notation_arguments:
2364 }
2365 }

```

(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

```

2366 \cs_new_protected:Nn \_stex_notation_arguments: {
2367   \int_incr:N \l_tmpa_int
2368   \str_if_empty:NTF \l_tmpa_str {
2369     \_stex_notation_final:
2370   }{
2371     \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2372     \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2373     \str_if_eq:VnTF \l_tmpb_str a {
2374       \_stex_notation_argument_assoc:n
2375     }{
2376       \str_if_eq:VnTF \l_tmpb_str B {
2377         \_stex_notation_argument_assoc:n
2378       }{
2379         \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2380         \tl_put_right:Nx \l_tmpa_tl {
2381           { \_stex_term_math_arg:nnn
2382             { \int_use:N \l_tmpa_int }
2383             { \l_tmpb_str }
2384             { ####\int_use:N \l_tmpa_int }
2385           }
2386         }
2387         \_stex_notation_arguments:
2388       }
2389     }
2390   }
2391 }

```

(End definition for `\_stex_notation_arguments:.`)

`\_stex_notation_argument_assoc:n`

```

2392 \cs_new_protected:Nn \_stex_notation_argument_assoc:n {
2393   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2394   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
2395   \tl_put_right:Nx \l_tmpa_tl {
2396     { \_stex_term_math_assoc_arg:nnnn
2397       { \int_use:N \l_tmpa_int }
2398       { \l_tmpb_str }
2399       \exp_args:No \exp_not:n
2400       {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
2401       { ####\int_use:N \l_tmpa_int }
2402     }
2403   }
2404   \_stex_notation_arguments:
2405 }

```

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

`\_stex_notation_final:` Called after processing all notation arguments

```

2406 \cs_new_protected:Nn \_stex_notation_final: {
2407   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
2408   \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
2409   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2410   \exp_args:Nne \use:nn

```

```

2411 {
2412 \cs_generate_from_arg_count:cNnn {
2413   stex_notation_ \l_tmpa_str \c_hash_str
2414   \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2415   _cs
2416 }
2417 \cs_set:Npn \l_tmpb_str } { {
2418   \exp_after:wN \exp_after:wN \exp_after:wN
2419   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2420   { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2421 } }
2422
2423 \tl_if_empty:NF \l__stex_notation_op_tl {
2424   \cs_set:cpx {
2425     stex_op_notation_ \l_tmpa_str \c_hash_str
2426     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2427     _cs
2428   } {
2429     \_stex_term_oms:nnn {
2430       \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2431       \l__stex_notation_lang_str
2432     }{
2433       \l_tmpa_str
2434     }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2435   }
2436 }
2437
2438 \exp_args:Ne
2439 \stex_add_to_current_module:n {
2440   \cs_generate_from_arg_count:cNnn {
2441     stex_notation_ \l_tmpa_str \c_hash_str
2442     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2443     _cs
2444   } \cs_set:Npn {\l_tmpb_str} {
2445     \exp_after:wN \exp_after:wN \exp_after:wN
2446     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2447     { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2448   }
2449   \tl_if_empty:NF \l__stex_notation_op_tl {
2450     \cs_set:cpn {
2451       stex_op_notation_ \l_tmpa_str \c_hash_str
2452       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2453       _cs
2454     } {
2455       \_stex_term_oms:nnn {
2456         \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2457         \l__stex_notation_lang_str
2458       }{
2459         \l_tmpa_str
2460       }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2461     }
2462   }
2463 }
2464

```

```

2465 \seq_put_right:cx {
2466   l_stex_symdecl_
2467   \prop_item:Nn \l_tmpb_prop { symbol }
2468   _notations
2469 } {
2470   \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2471 }
2472
2473 \stex_debug:nn{symbols}{
2474   Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2475   ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
2476   Operator~precedence:~
2477   \prop_item:Nn \l_tmpb_prop { opprec }^^J
2478   Argument~precedences:~
2479   \seq_use:Nn \l_tmpa_seq {,~}^^J
2480   Notation: \cs_meaning:c {
2481     stex_notation_ \l_tmpa_str \c_hash_str
2482     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2483     _cs
2484   }
2485 }
2486
2487 \prop_set_eq:cN {
2488   l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2489   \c_hash_str \l__stex_notation_lang_str _prop
2490 } \l_tmpb_prop
2491
2492 \exp_args:Ne
2493 \stex_add_to_current_module:n {
2494   \seq_put_right:cn {
2495     l_stex_symdecl_
2496     \prop_item:Nn \l_tmpb_prop { symbol }
2497     _notations
2498   } {
2499     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2500   }
2501   \prop_set_from_keyval:cn {
2502     l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2503     \c_hash_str \l__stex_notation_lang_str _prop
2504   } {
2505     symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2506     language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2507     variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,
2508     opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2509     argprecs    = \prop_item:Nn \l_tmpb_prop { argprecs }    ,
2510   }
2511 }
2512
2513 \stex_if_smsmode:TF {
2514   \stex_smsmode_set_codes:
2515   % \exp_args:Nx \stex_add_to_sms:n {
2516   %   \prop_set_from_keyval:cn {
2517   %     l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2518   %     \c_hash_str \l__stex_notation_lang_str _prop

```



```

2519 %      } {
2520 %          symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2521 %          language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2522 %          variant      = \prop_item:Nn \l_tmpb_prop { variant }      ,
2523 %          opprec       = \prop_item:Nn \l_tmpb_prop { opprec }       ,
2524 %          argprec      = \prop_item:Nn \l_tmpb_prop { argprec }      ,
2525 %      }
2526 %  }
2527 }{
2528
2529 % HTML annotations
2530 \stex_if_do_html:T {
2531     \stex_annotate_invisible:nnn { notation }
2532     { \prop_item:Nn \l_tmpb_prop { symbol } } {
2533         \stex_annotate_invisible:nnn { notationfragment }
2534         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
2535         \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2536         \stex_annotate_invisible:nnn { precedence }
2537         { \prop_item:Nn \l_tmpb_prop { opprec };
2538           \seq_use:Nn \l_tmpa_seq { x }
2539         }{}
2540
2541         \int_zero:N \l_tmpa_int
2542         \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2543         \tl_clear:N \l_tmpa_tl
2544         \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
2545             \int_incr:N \l_tmpa_int
2546             \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2547             \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2548             \str_if_eq:VnTF \l_tmpb_str a {
2549                 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2550                     \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2551                     \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2552                 } }
2553             }{
2554                 \str_if_eq:VnTF \l_tmpb_str B {
2555                     \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2556                         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2557                         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2558                     } }
2559                 }{
2560                     \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2561                         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2562                     } }
2563                 }
2564             }
2565         }
2566         \stex_annotate_invisible:nnn { notationcomp }{}{
2567             \str_set:Nx \l_stex_current_symbol_str {\prop_item:Nn \l_tmpb_prop { symbol }}
2568             $ \exp_args:Nno \use:nn { \use:c {
2569                 stex_notation_ \l_stex_current_symbol_str
2570                 \c_hash_str \l__stex_notation_variant_str
2571                 \c_hash_str \l__stex_notation_lang_str _cs
2572             } } { \l_tmpa_tl } $

```

```

2573     }
2574   }
2575 }
2576 }
2577 }

```

(End definition for `\_stex_notation_final:`.)

`\setnotation`

```

2578 \keys_define:nn { stex / setnotation } {
2579   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2580   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2581   unknown .code:n = \str_set:Nx
2582     \l__stex_notation_variant_str \l_keys_key_str
2583 }
2584
2585 \cs_new_protected:Nn \_stex_setnotation_args:n {
2586   \str_clear:N \l__stex_notation_lang_str
2587   \str_clear:N \l__stex_notation_variant_str
2588   \keys_set:nn { stex / setnotation } { #1 }
2589 }
2590
2591 \NewDocumentCommand \setnotation {m m} {
2592   \stex_get_symbol:n { #1 }
2593   \_stex_setnotation_args:n { #2 }
2594   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl\_l_stex_get_symbol_uri_str _notations }
2595     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2596     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notation
2597       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2598     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notation
2599       { \c_hash_str }
2600     \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notations
2601       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2602     \exp_args:Nx \stex_add_to_current_module:n {
2603       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notati
2604         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2605       \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notation
2606         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2607       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notati
2608         { \c_hash_str }
2609     }
2610     \stex_debug:nn {notations}{
2611       Setting~default~notation~
2612       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }~for~
2613       \l_stex_get_symbol_uri_str \
2614       \expandafter\meaning\csname
2615         l_stex_symdecl\_l_stex_get_symbol_uri_str _notations\endcsname
2616     }
2617   }{
2618     % todo throw error
2619   }
2620 }
2621

```

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

**\symdef**

```
2622 \keys_define:nn { stex / symdef } {
2623   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2624   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2625   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2626   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
2627   def       .tl_set:N    = \l_stex_symdecl_definiens_tl ,
2628   op        .tl_set:N    = \l__stex_notation_op_tl ,
2629   lang      .str_set_x:N = \l__stex_notation_lang_str ,
2630   variant   .str_set_x:N = \l__stex_notation_variant_str ,
2631   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2632   unknown   .code:n      = \str_set:Nx
2633             \l__stex_notation_variant_str \l_keys_key_str
2634 }
2635
2636 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2637   \str_clear:N \l_stex_symdecl_name_str
2638   \str_clear:N \l_stex_symdecl_args_str
2639   \bool_set_false:N \l_stex_symdecl_local_bool
2640   \tl_clear:N \l_stex_symdecl_type_tl
2641   \tl_clear:N \l_stex_symdecl_definiens_tl
2642   \str_clear:N \l__stex_notation_lang_str
2643   \str_clear:N \l__stex_notation_variant_str
2644   \str_clear:N \l__stex_notation_prec_str
2645   \tl_clear:N \l__stex_notation_op_tl
2646
2647   \keys_set:nn { stex / symdef } { #1 }
2648 }
2649
2650 \NewDocumentCommand \symdef { 0{} m } {
2651   \__stex_notation_symdef_args:n { #1 }
2652   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2653   \stex_symdecl_do:n { #2 }
2654   \exp_args:Nx \stex_notation_do:nn {
2655     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2656   }
2657 }
2658 \stex_deactivate_macro:Nn \symdef {module~environments}
2659
2659 (End definition for \symdef. This function is documented on page 37.)
2659 \</package>
```

## Chapter 31

# STEX -Terms Implementation

```
2660 <*package>
2661
2662 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2663
2664 <@@=stex_terms>
2665
2666   Warnings and error messages
2667 \msg_new:nnn{stex}{error/nonotation}{
2668   Symbol~#1~invoked,~but~has~no~notation~#2!
2669 }
2670 \msg_new:nnn{stex}{error/notationarg}{
2671   Error~in~parsing~notation~#1
2672 }
2673 \msg_new:nnn{stex}{error/noop}{
2674   Symbol~#1~has~no~operator~notation~for~notation~#2
2675 }
```

### 31.1 Symbol Invocations

Arguments:

```
2675 \keys_define:nn { stex / terms } {
2676   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2677   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2678   unknown .code:n = \str_set:Nx
2679     \l__stex_terms_variant_str \l_keys_key_str
2680 }
2681
2682 \cs_new_protected:Nn \__stex_terms_args:n {
2683   \str_clear:N \l__stex_terms_lang_str
2684   \str_clear:N \l__stex_terms_variant_str
2685   \str_clear:N \l__stex_terms_prec_str
2686   \tl_clear:N \l__stex_terms_op_tl
2687
2688   \keys_set:nn { stex / terms } { #1 }
```

2689 }

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

```
2690 \cs_new_protected:Nn \stex_invoke_symbol:n {
2691   \if_mode_math:
2692     \exp_after:wN \__stex_terms_invoke_math:n
2693   \else:
2694     \exp_after:wN \__stex_terms_invoke_text:n
2695   \fi: { #1 }
2696 }
```

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

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

```
2697 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2698   \peek_charcode_remove:NTF ! {
2699     \peek_charcode:NTF [ {
2700       \__stex_terms_invoke_op:nw { #1 }
2701     }{
2702       \peek_charcode_remove:NTF ! {
2703         \peek_charcode:NTF [ {
2704           \__stex_terms_invoke_op_custom:nw
2705         }{
2706           % TODO throw error
2707         }
2708       }{
2709         \__stex_terms_invoke_op:nw { #1 } []
2710       }
2711     }
2712   }{
2713     \peek_charcode_remove:NTF * {
2714       \__stex_terms_invoke_text:n { #1 }
2715     }{
2716       \peek_charcode:NTF [ {
2717         \__stex_terms_invoke_math:nw { #1 }
2718       }{
2719         \__stex_terms_invoke_math:nw { #1 } []
2720       }
2721     }
2722   }
2723 }
```

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

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

```
2724 \cs_new_protected:Npn \__stex_terms_invoke_op_custom:nw #1 [#2] {
2725   \stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
2726     \stex_highlight_term:nn{#1}{#2}
2727   }
2728 }
```

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

\\_stex\_terms\_invoke\_op:nw

```

2729 \cs_new_protected:Npn \_stex_terms_invoke_op:nw #1 [#2] {
2730   \_stex_terms_args:n { #2 }
2731   \cs_if_exist:cTF {
2732     stex_op_notation_ #1 \c_hash_str
2733     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2734   }{
2735     \csname stex_op_notation_ #1 \c_hash_str
2736       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2737     \endcsname
2738   }{
2739     \msg_error:nnxx{stex}{error/noop}{#1}{\l__stex_terms_variant_str \c_hash_str \l__stex_t
2740   }
2741 }

```

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

\\_stex\_terms\_invoke\_math:nw

```

2742 \cs_new_protected:Npn \_stex_terms_invoke_math:nw #1 [#2] {
2743   \_stex_terms_args:n { #2 }
2744   \seq_if_empty:cTF {
2745     l_stex_symdecl_ #1 _notations
2746   } {
2747     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
2748   } {
2749     \seq_if_in:cxTF {
2750       l_stex_symdecl_ #1 _notations
2751     }
2752     { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
2753       \str_set:Nn \l_stex_current_symbol_str { #1 }
2754       \use:c{
2755         stex_notation_ #1 \c_hash_str
2756         \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2757         _cs
2758       }
2759     }{
2760       \str_if_empty:NTF \l__stex_terms_variant_str {
2761         \str_if_empty:NTF \l__stex_terms_lang_str {
2762           \seq_get_left:cN {
2763             l_stex_symdecl_ #1 _notations
2764           } \l_tmpa_str
2765           \str_set:Nn \l_stex_current_symbol_str { #1 }
2766           \use:c{
2767             stex_notation_ #1 \c_hash_str \l_tmpa_str
2768             _cs
2769           }
2770         }{
2771           \msg_error:nnxx{stex}{error/nonotation}{#1}{
2772             ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2773           }
2774         }
2775       }{
2776         \msg_error:nnxx{stex}{error/nonotation}{#1}{
2777           ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str

```

```

2778     }
2779   }
2780 }
2781 }
2782 }

```

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

`\_stex_terms_invoke_text:n`

```

2783 \cs_new_protected:Nn \_stex_terms_invoke_text:n {
2784   \peek_charcode_remove:NTF ! {
2785     \stex_term_custom:nn { #1 } { }
2786   }{
2787     \prop_set_eq:Nc \l_tmpa_prop {
2788       l_stex_symdecl_ #1 _prop
2789     }
2790     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2791     \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2792   }
2793 }

```

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

## 31.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
2794 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2795 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2796 \int_new:N \l__stex_terms_downprec
2797 \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
2798 \tl_set:Nn \l__stex_terms_left_bracket_str (
2799 \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

```

2800 \cs_new_protected:Nn \_stex_terms_maybe_brackets:nn {
2801   \bool_if:NTF \l__stex_terms_brackets_done_bool {
2802     \bool_set_false:N \l__stex_terms_brackets_done_bool
2803     #2
2804   } {
2805     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
2806       \bool_if:NTF \l_stex_inarray_bool { #2 }{
2807         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
2808         \dobrackets { #2 }
2809       }

```

```

2810     }{ #2 }
2811   }
2812 }

```

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

### `\dobrackets`

```

2813 \bool_new:N \l__stex_terms_brackets_done_bool
2814 %\RequirePackage{scalerel}
2815 \cs_new_protected:Npn \dobrackets #1 {
2816   %\ThisStyle{\if D\m@switch
2817   %   \exp_args:Nnx \use:nn
2818   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
2819   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
2820   % \else
2821   \exp_args:Nnx \use:nn
2822   {
2823     \bool_set_true:N \l__stex_terms_brackets_done_bool
2824     \int_set:Nn \l__stex_terms_downprec \infprec
2825     \l__stex_terms_left_bracket_str
2826     #1
2827   }
2828   {
2829     \bool_set_false:N \l__stex_terms_brackets_done_bool
2830     \l__stex_terms_right_bracket_str
2831     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2832   }
2833   %\fi}
2834 }

```

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

### `\withbrackets`

```

2835 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2836   \exp_args:Nnx \use:nn
2837   {
2838     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
2839     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
2840     #3
2841   }
2842   {
2843     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
2844     {\l__stex_terms_left_bracket_str}
2845     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
2846     {\l__stex_terms_right_bracket_str}
2847   }
2848 }

```

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

### `\STEXinvisible`

```

2849 \cs_new_protected:Npn \STEXinvisible #1 {
2850   \stex_annotate_invisible:n { #1 }
2851 }

```



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

OMDoc terms:

`\_stex_term_math_oms:nnnn`

```

2852 \cs_new_protected:Nn \_stex_term_oms:nnn {
2853   \stex_annotate:nnn{ OMID }{ #2 }{
2854     \stex_highlight_term:nn { #1 } { #3 }
2855   }
2856 }
2857
2858 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2859   \__stex_terms_maybe_brackets:nn { #3 }{
2860     \stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2861   }
2862 }

```

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

`\_stex_term_math_oma:nnnn`

```

2863 \cs_new_protected:Nn \_stex_term_oma:nnn {
2864   \stex_annotate:nnn{ OMA }{ #2 }{
2865     \stex_highlight_term:nn { #1 } { #3 }
2866   }
2867 }
2868
2869 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2870   \__stex_terms_maybe_brackets:nn { #3 }{
2871     \stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2872   }
2873 }

```

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

`\_stex_term_math_omb:nnnn`

```

2874 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2875   \stex_annotate:nnn{ OMBIND }{ #2 }{
2876     \stex_highlight_term:nn { #1 } { #3 }
2877   }
2878 }
2879
2880 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2881   \__stex_terms_maybe_brackets:nn { #3 }{
2882     \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2883   }
2884 }

```

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

`\_stex_term_math_arg:nnn`

```

2885 \cs_new_protected:Nn \_stex_term_arg:nn {
2886   \stex_unhighlight_term:n {
2887     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2888   }
2889 }

```

```

2890 \cs_new_protected:Nn \stex_term_math_arg:nnn {
2891   \exp_args:Nnx \use:nn
2892   { \int_set:Nn \l__stex_terms_downprec { #2 }
2893     \stex_term_arg:nn { #1 }{ #3 }
2894   }
2895   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2896 }

```

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

`\stex_term_math_assoc_arg:nnnn`

```

2897 \cs_new_protected:Nn \stex_term_math_assoc_arg:nnnn {
2898   \clist_set:Nn \l_tmpa_clist{ #4 }
2899   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
2900     \tl_set:Nn \l_tmpa_tl { #4 }
2901   }{
2902     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
2903     \clist_reverse:N \l_tmpa_clist
2904     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
2905
2906     \clist_map_inline:Nn \l_tmpa_clist {
2907       \exp_args:NNNo \exp_args:Nno \tl_set:No \l_tmpa_tl {
2908         \exp_args:Nno
2909         \l_tmpa_cs { ##1 } \l_tmpa_tl
2910       }
2911     }
2912
2913   }
2914   \exp_args:Nnno
2915   \stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2916 }

```

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

`\stex_term_custom:nn`

```

2917 \cs_new_protected:Nn \stex_term_custom:nn {
2918   \str_set:Nn \l__stex_terms_custom_uri { #1 }
2919   \str_set:Nn \l_tmpa_str { #2 }
2920   \tl_clear:N \l_tmpa_tl
2921   \int_zero:N \l_tmpa_int
2922   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2923   \__stex_terms_custom_loop:
2924 }

```

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

`\__stex_terms_custom_loop:`

```

2925 \cs_new_protected:Nn \__stex_terms_custom_loop: {
2926   \bool_set_false:N \l_tmpa_bool
2927   \bool_while_do:nn {
2928     \str_if_eq_p:ee X {
2929       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2930     }
2931   }{
2932     \int_incr:N \l_tmpa_int

```

```

2933 }
2934
2935 \peek_charcode:NTF [ {
2936   % notation/text component
2937   \__stex_terms_custom_component:w
2938 } {
2939   \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2940     % all arguments read => finish
2941     \__stex_terms_custom_final:
2942   } {
2943     % arguments missing
2944     \peek_charcode_remove:NTF * {
2945       % invisible, specific argument position or both
2946       \peek_charcode:NTF [ {
2947         % visible specific argument position
2948         \__stex_terms_custom_arg:wn
2949       } {
2950         % invisible
2951         \peek_charcode_remove:NTF * {
2952           % invisible specific argument position
2953           \__stex_terms_custom_arg_inv:wn
2954         } {
2955           % invisible next argument
2956           \__stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2957         }
2958       }
2959     } {
2960       % next normal argument
2961       \__stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
2962     }
2963   }
2964 }
2965 }

```

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

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

```

2966 \cs_new_protected:Npn \__stex_terms_custom_arg_inv:wn [ #1 ] #2 {
2967   \bool_set_true:N \l_tmpa_bool
2968   \__stex_terms_custom_arg:wn [ #1 ] { #2 }
2969 }

```

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

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

```

2970 \cs_new_protected:Npn \__stex_terms_custom_arg:wn [ #1 ] #2 {
2971   \str_set:Nx \l_tmpb_str {
2972     \str_item:Nn \l_tmpa_str { #1 }
2973   }
2974   \str_case:VnTF \l_tmpb_str {
2975     { X } {
2976       \msg_error:nnx{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2977     }
2978     { i } { \__stex_terms_custom_set_X:n { #1 } }
2979     { b } { \__stex_terms_custom_set_X:n { #1 } }

```

```

2980 { a } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
2981 { B } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
2982 }{}{
2983 \msg_error:nnx{stex}{error/notationarg}{\l\_stex_terms_custom_uri}
2984 }
2985
2986 \bool_if:nTF \l_tmpa_bool {
2987   \tl_put_right:Nx \l_tmpa_tl {
2988     \stex_annotate_invisible:n {
2989       \stex_term_arg:nn { \int_eval:n { #1 } }
2990       \exp_not:n { { #2 } }
2991     }
2992   }
2993 } {
2994   \tl_put_right:Nx \l_tmpa_tl {
2995     \stex_term_arg:nn { \int_eval:n { #1 } }
2996     \exp_not:n { { #2 } }
2997   }
2998 }
2999
3000 \_stex_terms_custom_loop:
3001 }

```

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

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

```

3002 \cs_new_protected:Nn \_stex_terms_custom_set_X:n {
3003   \str_set:Nx \l_tmpa_str {
3004     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
3005     X
3006     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
3007   }
3008 }

```

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

\\_stex\_terms\_custom\_component:

```

3009 \cs_new_protected:Npn \_stex_terms_custom_component:w [ #1 ] {
3010   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
3011   \_stex_terms_custom_loop:
3012 }

```

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

\\_stex\_terms\_custom\_final:

```

3013 \cs_new_protected:Nn \_stex_terms_custom_final: {
3014   \int_compare:nNnTF \l_tmpb_int = 0 {
3015     \exp_args:Nnno \stex_term_oms:nnn
3016   }{
3017     \str_if_in:NnTF \l_tmpa_str {b} {
3018       \exp_args:Nnno \stex_term_ombind:nnn
3019     } {
3020       \exp_args:Nnno \stex_term_oma:nnn
3021     }
3022   }

```

```

3023 { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
3024 }

```

(End definition for `\_stex_terms_custom_final:`.)

`\symref`

`\symname`

```

3025 \NewDocumentCommand \symref { m m }{
3026   \let\compemph_uri_prev:\compemph@uri
3027   \let\compemph@uri\symrefemph@uri
3028   \STEXsymbol{#1}! [#2]
3029   \let\compemph@uri\compemph_uri_prev:
3030 }
3031
3032 \keys_define:nn { stex / symname } {
3033   post      .str_set_x:N    = \l_stex_symname_post_str
3034 }
3035
3036 \cs_new_protected:Nn \stex_symname_args:n {
3037   \str_clear:N \l_stex_symname_post_str
3038   \keys_set:nn { stex / symname } { #1 }
3039 }
3040
3041 \NewDocumentCommand \symname { O{} m }{
3042   \stex_symname_args:n { #1 }
3043   \stex_get_symbol:n { #2 }
3044   \str_set:Nx \l_tmpa_str {
3045     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3046   }
3047   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {-}
3048
3049   \let\compemph_uri_prev:\compemph@uri
3050   \let\compemph@uri\symrefemph@uri
3051   \exp_args:NNx \use:nn
3052   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }! [
3053     \l_tmpa_str \l_stex_symname_post_str
3054   ] }
3055   \let\compemph@uri\compemph_uri_prev:
3056 }

```

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

### 31.3 Notation Components

```

3057 <@@=stex_notationcomps>

```

`\stex_highlight_term:nn`

```

3058
3059 \str_new:N \l_stex_current_symbol_str
3060 \cs_new_protected:Nn \stex_highlight_term:nn {
3061   \exp_args:Nnx
3062   \use:nn {
3063     \str_set:Nx \l_stex_current_symbol_str { #1 }
3064     #2
3065   } {

```

```

3066 \str_set:Nx \exp_not:N \l_stex_current_symbol_str
3067 { \l_stex_current_symbol_str }
3068 }
3069 }
3070
3071 \cs_new_protected:Nn \stex_unhighlight_term:n {
3072 % \latexml_if:TF {
3073 % #1
3074 % } {
3075 % \rustex_if:TF {
3076 % #1
3077 % } {
3078 #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
3079 % }
3080 % }
3081 }

```

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

```

\comp
\compemph@uri 3082 \cs_new_protected:Npn \comp #1 {
\compemph 3083 \str_if_empty:NF \l_stex_current_symbol_str {
\defemph 3084 \rustex_if:TF {
\defemph@uri 3085 \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
\symrefemph 3086 }{
\symrefemph@uri 3087 \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3088 }
3089 }
3090 }
3091
3092 \cs_new_protected:Npn \compemph@uri #1 #2 {
3093 \compemph{ #1 }
3094 }
3095
3096
3097 \cs_new_protected:Npn \compemph #1 {
3098 #1
3099 }
3100
3101 \cs_new_protected:Npn \defemph@uri #1 #2 {
3102 \defemph{#1}
3103 }
3104
3105 \cs_new_protected:Npn \defemph #1 {
3106 \textbf{#1}
3107 }
3108
3109 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3110 \symrefemph{#1}
3111 }
3112
3113 \cs_new_protected:Npn \symrefemph #1 {
3114 \textbf{#1}
3115 }

```

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

## `\ellipses`

```
3116 \NewDocumentCommand \ellipses {} { \ldots }
```

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

```

\parray
\prmatrix 3117 \bool_new:N \l_stex_inarray_bool
\parrayline 3118 \bool_set_false:N \l_stex_inarray_bool
\parraylineh 3119 \NewDocumentCommand \parray { m m } {
\parraycell 3120 \begin{group}
3121 \bool_set_true:N \l_stex_inarray_bool
3122 \begin{array}{#1}
3123 #2
3124 \end{array}
3125 \end{group}
3126 }
3127
3128 \NewDocumentCommand \prmatrix { m } {
3129 \begin{group}
3130 \bool_set_true:N \l_stex_inarray_bool
3131 \begin{matrix}
3132 #1
3133 \end{matrix}
3134 \end{group}
3135 }
3136
3137 \def \maybepline {
3138 \bool_if:NT \l_stex_inarray_bool {\hline}
3139 }
3140
3141 \def \parrayline #1 #2 {
3142 #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3143 }
3144
3145 \def \pmrow #1 { \parrayline{}{ #1 } }
3146
3147 \def \parraylineh #1 #2 {
3148 #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3149 }
3150
3151 \def \parraycell #1 {
3152 #1 \bool_if:NT \l_stex_inarray_bool {\&}
3153 }

```

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

```
3154 \end{package}
```

## Chapter 32

# STEX -Structural Features Implementation

```
3155 <*package>
3156
3157 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3158
3159 <@@=stex_features>
3160
3161   Warnings and error messages
3162   \msg_new:nnn{stex}{error/copymodule/notallowed}{
3163     Symbol~#1~can~not~be~assigned~in~copymodule~#2
3164   }
3165   \msg_new:nnn{stex}{error/interpretmodule/noddefinens}{
3166     Symbol~#1~not~assigned~in~interpretmodule~#2
3167   }
3168
```

### 32.1 Imports with modification

```
3167 \cs_new_protected:Nn \stex_get_symbol_in_copymodule:n {
3168   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3169     \__stex_features_get_symbol_from_cs:n { #1 }
3170   }{
3171     % argument is a string
3172     % is it a command name?
3173     \cs_if_exist:cTF { #1 }{
3174       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3175       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3176       \str_if_empty:NTF \l_tmpa_str {
3177         \exp_args:Nx \cs_if_eq:NNTF {
3178           \tl_head:N \l_tmpa_tl
3179         } \stex_invoke_symbol:n {
3180           \exp_args:No \__stex_features_get_symbol_from_cs:n { \use:c { #1 } }
3181         }{
3182           \__stex_features_get_symbol_from_string:n { #1 }
3183         }
3184       }
3185     }
3186   }
3187
```



```

3183     }
3184   } {
3185     \__stex_features_get_symbol_from_string:n { #1 }
3186   }
3187   ){
3188     % argument is not a command name
3189     \__stex_features_get_symbol_from_string:n { #1 }
3190     % \l_stex_all_symbols_seq
3191   }
3192 }
3193 }
3194
3195 \cs_new_protected:Nn \__stex_features_get_symbol_from_string:n {
3196   \str_set:Nn \l_tmpa_str { #1 }
3197   \bool_set_false:N \l_tmpa_bool
3198   \bool_if:NF \l_tmpa_bool {
3199     \tl_set:Nn \l_tmpa_tl {
3200       \msg_set:nnn{stex}{error/unknownsymbol}{
3201         No~symbol~#1~found!
3202       }
3203       \msg_error:nn{stex}{error/unknownsymbol}
3204     }
3205     \str_set:Nn \l_tmpa_str { #1 }
3206     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3207     \seq_map_inline:Nn \l__stex_features_copymodule_fields_seq {
3208       \str_set:Nn \l_tmpb_str { ##1 }
3209       \str_if_eq:eeT { \l_tmpa_str } {
3210         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3211       } {
3212         \seq_map_break:n {
3213           \tl_set:Nn \l_tmpa_tl {
3214             \str_set:Nn \l_stex_get_symbol_uri_str {
3215               ##1
3216             }
3217             \__stex_features_get_symbol_check:
3218           }
3219         }
3220       }
3221     }
3222     \l_tmpa_tl
3223   }
3224 }
3225
3226 \cs_new_protected:Nn \__stex_features_get_symbol_from_cs:n {
3227   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3228   { \tl_tail:N \l_tmpa_tl }
3229   \tl_if_single:NTF \l_tmpa_tl {
3230     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3231       \exp_after:wN \str_set:Nn \exp_after:wN
3232       \l_stex_get_symbol_uri_str \l_tmpa_tl
3233       \__stex_features_get_symbol_check:
3234     }{
3235       % TODO
3236       % tail is not a single group

```

```

3237     }
3238 }{
3239     % TODO
3240     % tail is not a single group
3241 }
3242 }
3243
3244 \cs_new_protected:Nn \__stex_features_get_symbol_check: {
3245     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq {?} \l_stex_get_symbol_uri_str
3246     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 3 {
3247         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
3248         \str_set:Nx \l_tmpa_str {\seq_use:Nn \l_tmpa_seq ?}
3249         \seq_if_in:Nof \l__stex_features_copymodule_modules_seq \l_tmpa_str {
3250             \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3251                 \l_stex_current_copymodule_name_str\Allowed:~\seq_use:Nn \l__stex_features_copymodule_modules_seq \l_tmpa_str
3252             }
3253         }
3254     }{
3255         \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3256             \l_stex_current_copymodule_name_str~(inexplicably)
3257         }
3258     }
3259 }
3260
3261 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
3262     \stex_import_module_uri:nn { #1 } { #2 }
3263     \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3264     \stex_import_require_module:nnnn
3265     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3266     { \l_stex_import_path_str } { \l_stex_import_name_str }
3267     \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3268     \seq_set_eq:NN \l__stex_features_copymodule_modules_seq \l_stex_collect_imports_seq
3269     \seq_clear:N \l__stex_features_copymodule_fields_seq
3270     \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3271         \seq_map_inline:cn {c_stex_module_###1_constants}{
3272             \exp_args:NNx \seq_put_right:Nn \l__stex_features_copymodule_fields_seq {
3273                 ###1 ? ####1
3274             }
3275         }
3276     }
3277     \seq_clear:N \l_tmpa_seq
3278     \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3279         name      = \l_stex_current_copymodule_name_str ,
3280         module     = \l_stex_current_module_str ,
3281         from       = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3282         includes   = \l_tmpa_seq ,
3283         fields     = \l_tmpa_seq
3284     }
3285     \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3286         as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3287     \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_features_copymodule_modules_seq
3288     \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_features_copymodule_fields_seq {,~}
3289     \stex_if_smsmode:TF {
3290         \stex_smsmode_set_codes:

```

```

3291 } {
3292   \begin{stex_annotate_env} {#4} {
3293     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3294   }
3295   \stex_annotate_invisible:nnn{from}{\l_stex_import_ns_str ?\l_stex_import_name_str}{ }
3296 }
3297 \bool_set_eq:NN \l__stex_features_oldhtml_bool \l_stex_html_do_output_bool
3298 \bool_set_false:N \l_stex_html_do_output_bool
3299 }
3300 \cs_new_protected:Nn \stex_copymodule_end:n {
3301   \def \l_tmpa_cs ##1 ##2 {#1}
3302   \bool_set_eq:NN \l_stex_html_do_output_bool \l__stex_features_oldhtml_bool
3303   \tl_clear:N \l_tmpa_tl
3304   \prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3305   \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3306     \seq_map_inline:cn {c_stex_module_##1_constants}{\stex_annotate:nnn{assignment} {##1?###
3307       \l_tmpa_cs{##1}{####1}
3308       \str_if_exist:cTF {l__stex_features_copymodule_##1?####1_name_str} {
3309         \tl_put_right:Nx \l_tmpa_tl {
3310           \prop_set_from_keyval:cn {
3311             l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_#
3312           }{
3313             \exp_after:wN \prop_to_keyval:N \csname
3314               l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule
3315             \endcsname
3316           }
3317           \seq_clear:c {
3318             l_stex_symdecl_
3319             \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_name_s
3320             _notations
3321           }
3322         }
3323         \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_features_copymodule_##1?####1_nam
3324         \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{l__stex_features_
3325         \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3326           \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?###
3327           \tl_put_right:Nx \l_tmpa_tl {
3328             \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3329               \stex_invoke_symbol:n {
3330                 \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_na
3331               }
3332             }
3333           }
3334         }
3335       }{
3336         \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1_prop}
3337         \prop_put:Nnx \l_tmpa_prop { name }{\l_stex_current_copymodule_name_str / ####1 }
3338         \prop_put:Nnx \l_tmpa_prop { module }{\l_stex_current_module_str }
3339         \tl_put_right:Nx \l_tmpa_tl {
3340           \prop_set_from_keyval:cn {
3341             l_stex_symdecl_\l_stex_current_module_str ? \l_stex_current_copymodule_name_str
3342           }{
3343             \prop_to_keyval:N \l_tmpa_prop
3344           }

```

```

3345         \seq_clear:c {
3346             l_stex_symdecl_
3347             \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / #####1
3348             _notations
3349         }
3350     }
3351     \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str / #####1}
3352     \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3353         \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}
3354         \tl_put_right:Nx \l_tmpa_tl {
3355             \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3356                 \stex_invoke_symbol:n {
3357                     \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / #####1
3358                 }
3359             }
3360         }
3361     }
3362 }
3363 \tl_if_exist:cT {l__stex_features_copymodule_##1?####1_def_tl}{
3364     \stex_annotate_invisible:nnn{definiens}{\use:c{l__stex_features_copymodule_##1?####1_def_tl}}
3365 }
3366 % todo notations
3367 }}
3368 }
3369 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3370 \tl_put_left:Nx \l_tmpa_tl {
3371     \prop_set_from_keyval:cn {
3372         l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _pro
3373     }{
3374         \prop_to_keyval:N \l_stex_current_copymodule_prop
3375     }
3376 }
3377 \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl
3378 \stex_debug:nn{copymodule}{result:\meaning \l_tmpa_tl}
3379 \exp_args:Nx \stex_do_aftergroup:n {
3380     \exp_args:No \exp_not:n \l_tmpa_tl
3381 }
3382 \stex_if_smsmode:F {
3383     \end{stex_annotate_env}
3384 }
3385 }
3386
3387 \NewDocumentEnvironment {copymodule} { 0{} m m}{
3388     \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ structure }
3389     \stex_deactivate_macro:Nn \symdecl {module~environments}
3390     \stex_deactivate_macro:Nn \symdef {module~environments}
3391     \stex_deactivate_macro:Nn \notation {module~environments}
3392     \stex_reactivate_macro:N \assign
3393     \stex_reactivate_macro:N \renamedec1
3394     \stex_reactivate_macro:N \donotcopy
3395 }{
3396     \stex_copymodule_end:n {}
3397 }
3398

```

```

3399 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
3400   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ realization }
3401   \stex_deactivate_macro:Nn \symdecl {module~environments}
3402   \stex_deactivate_macro:Nn \symdef {module~environments}
3403   \stex_deactivate_macro:Nn \notation {module~environments}
3404   \stex_reactivate_macro:N \assign
3405   \stex_reactivate_macro:N \renamedekl
3406   \stex_reactivate_macro:N \donotcopy
3407 }{
3408   \stex_copymodule_end:n {
3409     \tl_if_exist:cF {
3410       l__stex_features_copymodule_##1?##2_def_tl
3411     }{
3412       \msg_error:nnxx{stex}{error/interpretmodule/nodedefiniens}{
3413         ##1?##2
3414       }{\l_stex_current_copymodule_name_str}
3415     }
3416   }
3417 }
3418
3419 \NewDocumentCommand \donotcopy { 0{} m}{
3420   \stex_import_module_uri:nn { #1 } { #2 }
3421   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3422   \seq_map_inline:Nn \l_stex_collect_imports_seq {
3423     \seq_remove_all:Nn \l_stex_features_copymodule_modules_seq { ##1 }
3424     \seq_map_inline:cn {c_stex_module_##1_constants}{
3425       \seq_remove_all:Nn \l_stex_features_copymodule_fields_seq { ##1 ? #####1 }
3426       \bool_lazy_any_p:nT {
3427         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_name_str}}
3428         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_macroname_str}}
3429         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_def_tl}}
3430       }{
3431         % TODO throw error
3432       }
3433     }
3434   }
3435
3436   \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
3437   \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_ns_str ?\l_stex_import_name_str }
3438   \prop_put:Nnx \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
3439 }
3440
3441 \NewDocumentCommand \assign { m m }{
3442   \stex_get_symbol_in_copymodule:n {#1}
3443   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
3444   \tl_set:cn {l__stex_features_copymodule_\l_stex_get_symbol_uri_str_def_tl}{#2}
3445 }
3446
3447 \keys_define:nn { stex / renamedekl } {
3448   name .str_set_x:N = \l_stex_renamedekl_name_str
3449 }
3450 \cs_new_protected:Nn \__stex_features_renamedekl_args:n {
3451   \str_clear:N \l_stex_renamedekl_name_str
3452

```

```

3453 \keys_set:nn { stex / renamedec1 } { #1 }
3454 }
3455
3456 \NewDocumentCommand \renamedec1 { 0{} m m}{
3457   \__stex_features_renamedec1_args:n { #1 }
3458   \stex_get_symbol_in_copymodule:n {#2}
3459   \stex_debug:nn{renamedec1}{renaming-{\l_stex_get_symbol_uri_str}-to~#3}
3460   \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
3461   \str_if_empty:NTF \l_stex_renamedec1_name_str {
3462     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3463       \l_stex_get_symbol_uri_str
3464     } }
3465   } {
3466     \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex_r
3467     \stex_debug:nn{renamedec1}{@~\l_stex_current_module_str ? \l_stex_renamedec1_name_str}
3468     \prop_set_eq:cc {l_stex_symdecl_
3469       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3470     _prop
3471     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
3472     \seq_set_eq:cc {l_stex_symdecl_
3473       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3474     _notations
3475     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
3476     \prop_put:cnx {l_stex_symdecl_
3477       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3478     _prop
3479     }{ name }{ \l_stex_renamedec1_name_str }
3480     \prop_put:cnx {l_stex_symdecl_
3481       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3482     _prop
3483     }{ module }{ \l_stex_current_module_str }
3484     \exp_args:NNx \seq_put_left:Nn \l__stex_features_copymodule_fields_seq {
3485       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3486     }
3487     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3488       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3489     } }
3490   }
3491 }
3492 %\NewDocumentCommand \notation_in_copymodules: { 0{} m } {
3493 % \_stex_notation_args:n { #1 }
3494 % \tl_clear:N \l_stex_symdecl_definiens_tl
3495 % \stex_get_symbol_in_copymodule:n { #2 }
3496 % \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
3497 % % todo
3498 %}
3499 \stex_deactivate_macro:Nn \assign {copymodules}
3500 \stex_deactivate_macro:Nn \renamedec1 {copymodules}
3501 \stex_deactivate_macro:Nn \donotcopy {copymodules}
3502
3503
3504 \seq_new:N \l_stex_implicit_morphisms_seq
3505 \NewDocumentCommand \implicitmorphism { 0{} m m}{
3506   \stex_import_module_uri:nn { #1 } { #2 }

```

```

3507 \stex_debug:nn{implicits}{
3508   Implicit~morphism:~
3509   \l_stex_module_ns_str ? \l__stex_features_name_str
3510 }
3511 \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
3512   \l_stex_module_ns_str ? \l__stex_features_name_str
3513 }{
3514   \msg_error:nnn{stex}{error/conflictingmodules}{
3515     \l_stex_module_ns_str ? \l__stex_features_name_str
3516   }
3517 }
3518
3519 % TODO
3520
3521
3522
3523 \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
3524   \l_stex_module_ns_str ? \l__stex_features_name_str
3525 }
3526 }
3527

```

## 32.2 The feature environment

structural@feature

```

3528
3529 \NewDocumentEnvironment{structural@feature}{ m m m }{
3530   \stex_if_in_module:F {
3531     \msg_set:nnn{stex}{error/nomodule}{
3532       Structural~Feature~has~to~occur~in~a~module:\\
3533       Feature~#2~of~type~#1\\
3534       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
3535     }
3536     \msg_error:nn{stex}{error/nomodule}
3537   }
3538
3539   \str_set:Nx \l_stex_module_name_str {
3540     \prop_item:Nn \l_stex_current_module_prop
3541     { name } / #2 - feature
3542   }
3543
3544   \str_set:Nx \l_stex_module_ns_str {
3545     \prop_item:Nn \l_stex_current_module_prop
3546     { ns }
3547   }
3548
3549
3550   \str_clear:N \l_tmpa_str
3551   \seq_clear:N \l_tmpa_seq
3552   \tl_clear:N \l_tmpa_tl
3553   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
3554     origname = #2,
3555     name      = \l_stex_module_name_str ,
3556     ns        = \l_stex_module_ns_str ,

```

```

3557 imports = \exp_not:o { \l_tmpa_seq } ,
3558 constants = \exp_not:o { \l_tmpa_seq } ,
3559 content = \exp_not:o { \l_tmpa_tl } ,
3560 file = \exp_not:o { \g_stex_currentfile_seq } ,
3561 lang = \l_stex_module_lang_str ,
3562 sig = \l_tmpa_str ,
3563 meta = \l_tmpa_str ,
3564 feature = #1 ,
3565 }
3566
3567 \stex_if_smsmode:TF {
3568   \stex_smsmode_set_codes:
3569 } {
3570   \begin{stex_annotate_env}{ feature:#1 }{}
3571   \stex_annotate_invisible:nnn{header}{}{ #3 }
3572 }
3573 }{
3574   \str_set:Nx \l_tmpa_str {
3575     c_stex_feature_
3576     \prop_item:Nn \l_stex_current_module_prop { ns } ?
3577     \prop_item:Nn \l_stex_current_module_prop { name }
3578     _prop
3579   }
3580   \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
3581   \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
3582   \stex_if_smsmode:TF {
3583     \exp_args:Nx \stex_add_to_sms:n {
3584       \prop_gset_from_keyval:cn {
3585         c_stex_feature_
3586         \prop_item:Nn \l_stex_current_module_prop { ns } ?
3587         \prop_item:Nn \l_stex_current_module_prop { name }
3588         _prop
3589       } {
3590         origname = #2,
3591         name = \prop_item:cn { \l_tmpa_str } { name } ,
3592         ns = \prop_item:cn { \l_tmpa_str } { ns } ,
3593         imports = \prop_item:cn { \l_tmpa_str } { imports } ,
3594         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
3595         content = \prop_item:cn { \l_tmpa_str } { content } ,
3596         file = \prop_item:cn { \l_tmpa_str } { file } ,
3597         lang = \prop_item:cn { \l_tmpa_str } { lang } ,
3598         sig = \prop_item:cn { \l_tmpa_str } { sig } ,
3599         meta = \prop_item:cn { \l_tmpa_str } { meta } ,
3600         feature = \prop_item:cn { \l_tmpa_str } { feature }
3601       }
3602     }
3603   } {
3604     \end{stex_annotate_env}
3605   }
3606 }
3607

```



## 32.3 Features

structure

```

3608
3609 \prop_new:N \l_stex_all_structures_prop
3610
3611 \keys_define:nn { stex / features / structure } {
3612   name          .str_set_x:N = \l__stex_features_structure_name_str ,
3613 }
3614
3615 \cs_new_protected:Nn \__stex_features_structure_args:n {
3616   \str_clear:N \l__stex_features_structure_name_str
3617   \keys_set:nn { stex / features / structure } { #1 }
3618 }
3619
3620 %\stex_new_feature:nnnn { structure } { 0{} m } {
3621   % \__stex_features_structure_args:n { ##1 }
3622   % \str_if_empty:NT \l__stex_features_structure_name_str {
3623     % \str_set:Nx \l__stex_features_structure_name_str { ##2 }
3624   % }
3625 %} {
3626 %
3627 %}
3628
3629 \NewDocumentEnvironment{mathstructure}{0{} m }{
3630   \__stex_features_structure_args:n { #1 }
3631   \str_if_empty:NT \l__stex_features_structure_name_str {
3632     \str_set:Nx \l__stex_features_structure_name_str { #2 }
3633   }
3634   \exp_args:Nnnx
3635   \begin{structural@feature}{ structure }
3636     { \l__stex_features_structure_name_str }{}
3637     \seq_clear:N \l_tmpa_seq
3638     \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
3639
3640   }{
3641     \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
3642     \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
3643     \str_set:Nx \l_tmpa_str {
3644       \prop_item:Nn \l_stex_current_module_prop { ns } ?
3645       \prop_item:Nn \l_stex_current_module_prop { name }
3646     }
3647     \seq_map_inline:Nn \l_tmpa_seq {
3648       \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
3649     }
3650     \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
3651     \exp_args:Nnx
3652     \AddToHookNext { env / mathstructure / after }{
3653       \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}}{
3654         \stex_term_math_oms:nnnn { \l_tmpa_str }{}{0}{}
3655       }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
3656     \STEXexport {
3657       \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3658       { \prop_item:Nn \l_stex_current_module_prop { origname } }

```

```

3659         {\l_tmpa_str}
3660         \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3661         {#2}{\l_tmpa_str}
3662 %       \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3663 %       \prop_item:Nn \l_stex_current_module_prop { origname },
3664 %       \l_tmpa_str
3665 %     }
3666 %     \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3667 %     #2,\l_tmpa_str
3668 %   }
3669 %   \tl_set:cx { #2 } {
3670 %     \stex_invoke_structure:n { \l_tmpa_str }
3671 %   }
3672 }
3673
3674 \end{structural@feature}
3675 % \g_stex_last_feature_prop
3676 }

```

\instantiate

```

3677 \seq_new:N \l__stex_features_structure_field_seq
3678 \str_new:N \l__stex_features_structure_field_str
3679 \str_new:N \l__stex_features_structure_def_tl
3680 \prop_new:N \l__stex_features_structure_prop
3681 \NewDocumentCommand \instantiate { m O{} m }{
3682   \stex_smsmode_set_codes:
3683   \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
3684   \prop_set_eq:Nc \l__stex_features_structure_prop {
3685     c_stex_feature_\l_tmpa_str _prop
3686   }
3687   \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
3688   \seq_map_inline:Nn \l__stex_features_structure_field_seq {
3689     \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
3690     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3691       \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
3692       \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
3693       {!} \l_tmpa_tl
3694       \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3695         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
3696         \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3697         \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3698       }{
3699         \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
3700         \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3701         \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
3702         \l_tmpa_tl
3703         \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3704           \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
3705           \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3706         }{
3707           \tl_clear:N \l_tmpb_tl
3708         }
3709       }
3710     }{

```

```

3711 \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
3712 \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3713   \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
3714   \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
3715   \tl_clear:N \l_tmpa_tl
3716 }{
3717   % TODO throw error
3718 }
3719 }
3720 % \l_tmpa_str: name
3721 % \l_tmpa_tl: definiens
3722 % \l_tmpb_tl: notation
3723 \tl_if_empty:NT \l__stex_features_structure_field_str {
3724   % TODO throw error
3725 }
3726 \str_clear:N \l_tmpb_str
3727
3728 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3729 \seq_map_inline:Nn \l_tmpa_seq {
3730   \seq_set_split:Nnn \l_tmpb_seq ? { ###1 }
3731   \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
3732   \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
3733     \seq_map_break:n {
3734       \str_set:Nn \l_tmpb_str { ###1 }
3735     }
3736   }
3737 }
3738 \prop_get:cnN { l_stex_symdecl_ \l_tmpb_str _prop } {args}
3739 \l_tmpb_str
3740
3741 \tl_if_empty:NTF \l_tmpb_tl {
3742   \tl_if_empty:NF \l_tmpa_tl {
3743     \exp_args:Nx \use:n {
3744       \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fe
3745     }
3746   }
3747 }{
3748   \tl_if_empty:NTF \l_tmpa_tl {
3749     \exp_args:Nx \use:n {
3750       \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\
3751     }
3752   }
3753 }{
3754   \exp_args:Nx \use:n {
3755     \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
3756     \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
3757   }
3758 }
3759 }
3760 % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
3761 % \prop_item:Nn \l_stex_current_module_prop {name} ?
3762 % #3/\l__stex_features_structure_field_str
3763 % \par
3764 % \expandafter\present\csname

```

```

3765 %      l_stex_symdecl_
3766 %      \prop_item:Nn \l_stex_current_module_prop {ns} ?
3767 %      \prop_item:Nn \l_stex_current_module_prop {name} ?
3768 %      #3/\l__stex_features_structure_field_str
3769 %      _prop
3770 %      \endcsname
3771 }
3772
3773 \tl_clear:N \l__stex_features_structure_def_tl
3774
3775 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3776 \seq_map_inline:Nn \l_tmpa_seq {
3777   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3778   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3779   \exp_args:Nx \use:n {
3780     \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
3781
3782     }
3783   }
3784
3785   \prop_if_exist:cF {
3786     l_stex_symdecl_
3787     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3788     \prop_item:Nn \l_stex_current_module_prop {name} ?
3789     #3/\l_tmpa_str
3790     _prop
3791   }{
3792     \prop_get:cnN { l_stex_symdecl_ ##1 _prop } {args}
3793     \l_tmpb_str
3794     \exp_args:Nx \use:n {
3795       \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
3796     }
3797   }
3798 }
3799
3800 \symdecl*[type={\STEXsymbol{module-type}}{
3801   \_stex_term_math_oms:nnnn {
3802     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3803     \prop_item:Nn \l__stex_features_structure_prop {name}
3804     }{}{0}{}
3805   }]}{#3}
3806
3807 % TODO: -> sms file
3808
3809 \tl_set:cx{ #3 }{
3810   \stex_invoke_structure:nnn {
3811     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3812     \prop_item:Nn \l_stex_current_module_prop {name} ? #3
3813   } {
3814     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3815     \prop_item:Nn \l__stex_features_structure_prop {name}
3816   }
3817 }
3818

```

3819 }

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

\stex\_invoke\_structure:nnn

```

3820 % #1: URI of the instance
3821 % #2: URI of the instantiated module
3822 \cs_new_protected:Nn \stex_invoke_structure:nnn {
3823   \tl_if_empty:nTF{ #3 }{
3824     \prop_set_eq:Nc \l__stex_features_structure_prop {
3825       c_stex_feature_ #2 _prop
3826     }
3827     \tl_clear:N \l_tmpa_tl
3828     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3829     \seq_map_inline:Nn \l_tmpa_seq {
3830       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3831       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3832       \cs_if_exist:cT {
3833         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
3834       }{
3835         \tl_if_empty:NF \l_tmpa_tl {
3836           \tl_put_right:Nn \l_tmpa_tl {,}
3837         }
3838         \tl_put_right:Nx \l_tmpa_tl {
3839           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
3840         }
3841       }
3842     }
3843     \exp_args:No \mathstrut \l_tmpa_tl
3844   }{
3845     \stex_invoke_symbol:n{#1/#3}
3846   }
3847 }
```

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

3848 </package>

## Chapter 33

# STEX -Statements Implementation

```
3849 <*package>
3850
3851 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3852
3853 \protected\def\ignorespacesandpars{
3854   \begingroup\catcode13=10\relax
3855   \@ifnextchar\par{
3856     \endgroup\expandafter\ignorespacesandpars\@gobble
3857   }{
3858     \endgroup
3859   }
3860 }
3861
3862 <@@=stex_statements>
3863
3864   Warnings and error messages
```

\titleemph

```
3864 \def\titleemph#1{\textbf{#1}}
```

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

### 33.1 Definitions

definiendum

```
3865 \keys_define:nn {stex / definiendum }{
3866   post      .tl_set:N      = \l__stex_statements_definiendum_post_tl,
3867   root      .str_set_x:N   = \l__stex_statements_definiendum_root_str,
3868   gfa       .str_set_x:N   = \l__stex_statements_definiendum_gfa_str
3869 }
3870 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
3871   \str_clear:N \l__stex_statements_definiendum_root_str
3872   \tl_clear:N \l__stex_statements_definiendum_post_tl
3873   \str_clear:N \l__stex_statements_definiendum_gfa_str
```

```

3874 \keys_set:nn { stex / definiendum } { #1 }
3875 }
3876 \NewDocumentCommand \definiendum { 0{} m m } {
3877   \__stex_statements_definiendum_args:n { #1 }
3878   \stex_get_symbol:n { #2 }
3879   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3880   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
3881     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
3882       \tl_set:Nn \l_tmpa_tl { #3 }
3883     } {
3884       \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
3885       \tl_set:Nn \l_tmpa_tl {
3886         \l__stex_statements_definiendum_root_str\l__stex_statements_definiendum_post_tl
3887       }
3888     }
3889   } {
3890     \tl_set:Nn \l_tmpa_tl { #3 }
3891   }
3892
3893   % TODO root
3894   \rustex_if:TF {
3895     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
3896   } {
3897     \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
3898   }
3899 }
3900 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

## definame

```

3901
3902 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3903
3904 \NewDocumentCommand \definame { 0{} m } {
3905   \__stex_statements_definiendum_args:n { #1 }
3906   % TODO: root
3907   \stex_get_symbol:n { #2 }
3908   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3909   \str_set:Nx \l_tmpa_str {
3910     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3911   }
3912   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3913   \rustex_if:TF {
3914     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3915       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3916     }
3917   } {
3918     \defemph@uri {
3919       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3920     } { \l_stex_get_symbol_uri_str }
3921   }
3922 }
3923 \stex_deactivate_macro:Nn \definame {definition~environments}

```

```

3924
3925 \NewDocumentCommand \Definame { 0{ } m } {
3926   \__stex_statements_definiendum_args:n { #1 }
3927   \stex_get_symbol:n { #2 }
3928   \str_set:Nx \l_tmpa_str {
3929     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3930   }
3931   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3932   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3933   \rustex_if:TF {
3934     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3935       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3936     }
3937   } {
3938     \defemph@uri {
3939       \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
3940     } { \l_stex_get_symbol_uri_str }
3941   }
3942 }
3943 \stex_deactivate_macro:Nn \Definame {definition-environments}
3944
3945 \NewDocumentCommand \Symname { 0{ } m }{
3946   \stex_symname_args:n { #1 }
3947   \stex_get_symbol:n { #2 }
3948   \str_set:Nx \l_tmpa_str {
3949     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3950   }
3951   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3952   \let\compemph_uri_prev:\compemph@uri
3953   \let\compemph@uri\symrefemph@uri
3954   \exp_args:NNx \use:nn
3955   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }! [
3956     \exp_after:wN \stex_capitalize:n \l_tmpa_str
3957     \l_stex_symname_post_str
3958   ] }
3959   \let\compemph@uri\compemph_uri_prev:
3960 }

```

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

## sdefinition

```

3961
3962 \keys_define:nn {stex / sdefinition }{
3963   type      .str_set_x:N = \sdefinitiontype,
3964   id        .str_set_x:N = \sdefinitionid,
3965   name      .str_set_x:N = \sdefinitionname,
3966   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
3967   title     .tl_set:N     = \sdefinitiontitle
3968 }
3969 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
3970   \str_clear:N \sdefinitiontype
3971   \str_clear:N \sdefinitionid
3972   \str_clear:N \sdefinitionname
3973   \clist_clear:N \l__stex_statements_sdefinition_for_clist

```



```

3974 \tl_clear:N \sdefinitiontitle
3975 \keys_set:nn { stex / sdefinition }{ #1 }
3976 }
3977
3978 \NewDocumentEnvironment{sdefinition}{O{}}{
3979   \__stex_statements_sdefinition_args:n{ #1 }
3980   \stex_reactivate_macro:N \definiendum
3981   \stex_reactivate_macro:N \definame
3982   \stex_reactivate_macro:N \Definame
3983   \stex_smsmode_set_codes:
3984   \stex_if_smsmode:TF {
3985     \stex_smsmode_set_codes:
3986   }{
3987     \seq_clear:N \l_tmpa_seq
3988     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
3989       \str_if_eq:nnF{ ##1 }{ }{
3990         \stex_get_symbol:n { ##1 }
3991         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
3992           \l_stex_get_symbol_uri_str
3993         }
3994       }
3995     }
3996     \exp_args:Nnnx
3997     \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {,}}
3998     \str_if_empty:NF \sdefinitiontype {
3999       \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{}
4000     }
4001     \clist_set:No \l_tmpa_clist \sdefinitiontype
4002     \tl_clear:N \l_tmpa_tl
4003     \clist_map_inline:Nn \l_tmpa_clist {
4004       \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
4005         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
4006       }
4007     }
4008     \tl_if_empty:NTF \l_tmpa_tl {
4009       \__stex_statements_sdefinition_start:
4010     }{
4011       \l_tmpa_tl
4012     }
4013   }
4014   \stex_ref_new_doc_target:n \sdefinitionid
4015 }{
4016   \str_if_empty:NF \sdefinitionname { \symdecl*{\sdefinitionname} }
4017   \stex_if_smsmode:F {
4018     \clist_set:No \l_tmpa_clist \sdefinitiontype
4019     \tl_clear:N \l_tmpa_tl
4020     \clist_map_inline:Nn \l_tmpa_clist {
4021       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
4022         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
4023       }
4024     }
4025     \tl_if_empty:NTF \l_tmpa_tl {
4026       \__stex_statements_sdefinition_end:
4027     }{

```

```

4028     \l_tmpa_tl
4029   }
4030   \end{stex_annotate_env}
4031 }
4032 }

```

`\stexpatchdefinition`

```

4033 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
4034   \par\noindent\titllemph{Definition\tl_if_empty:NF \sdefinitiontitle {
4035     ~(\sdefinitiontitle)
4036   }~}
4037 }
4038 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\par\medskip}
4039
4040 \newcommand\stexpatchdefinition[3][] {
4041   \str_set:Nx \l_tmpa_str{ #1 }
4042   \str_if_empty:NTF \l_tmpa_str {
4043     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
4044     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
4045   }{
4046     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
4047     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
4048   }
4049 }

```

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

`\inlinedef` inline:

```

4050 \keys_define:nn {stex / inlinedef }{
4051   type      .str_set_x:N = \sdefinitiontype,
4052   id        .str_set_x:N = \sdefinitionid,
4053   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4054   name      .str_set_x:N = \sdefinitionname
4055 }
4056 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
4057   \str_clear:N \sdefinitiontype
4058   \str_clear:N \sdefinitionid
4059   \str_clear:N \sdefinitionname
4060   \clist_clear:N \l__stex_statements_sdefinition_for_clist
4061   \keys_set:nn { stex / inlinedef }{ #1 }
4062 }
4063 \NewDocumentCommand \inlinedef { 0{} m } {
4064   \begingroup
4065   \__stex_statements_inlinedef_args:n{ #1 }
4066   \stex_ref_new_doc_target:n \sdefinitionid
4067   \stex_reactivate_macro:N \definiendum
4068   \stex_reactivate_macro:N \definame
4069   \stex_if_smsmode:TF{
4070     \stex_smsmode_set_codes:
4071     \str_if_empty:NF \sdefinitionname { \symdecl*{\sdefinitionname} }
4072   }{
4073     \seq_clear:N \l_tmpa_seq
4074     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4075       \str_if_eq:nnF{ ##1 }{ }{

```

```

4076         \stex_get_symbol:n { ##1 }
4077         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4078             \l_stex_get_symbol_uri_str
4079         }
4080     }
4081 }
4082 \exp_args:Nnx
4083 \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
4084     \str_if_empty:NF \sdefinitiontype {
4085         \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{}
4086     }
4087     #2
4088     \str_if_empty:NF \sdefinitionname { \symdecl*{\sdefinitionname} }
4089 }
4090 }
4091 \endgroup
4092 }

```

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

## 33.2 Assertions

**sassertion**

```

4093
4094 \keys_define:nn {stex / sassertion }{
4095     type      .str_set_x:N = \sassertiontype,
4096     id        .str_set_x:N = \sassertionid,
4097     title     .tl_set:N    = \sassertiontitle ,
4098     for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4099     name      .str_set_x:N = \sassertionname
4100 }
4101 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
4102     \str_clear:N \sassertiontype
4103     \str_clear:N \sassertionid
4104     \str_clear:N \sassertionname
4105     \clist_clear:N \l__stex_statements_sassertion_for_clist
4106     \tl_clear:N \sassertiontitle
4107     \keys_set:nn { stex / sassertion }{ #1 }
4108 }
4109
4110 %\tl_new:N \g__stex_statements_aftergroup_tl
4111
4112 \NewDocumentEnvironment{sassertion}{0{}}{
4113     \__stex_statements_sassertion_args:n{ #1 }
4114     \stex_smsmode_set_codes:
4115     \stex_if_smsmode:TF {
4116         \stex_smsmode_set_codes:
4117     } {
4118         \seq_clear:N \l_tmpa_seq
4119         \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4120             \str_if_eq:nnF{ ##1 }{}{
4121                 \stex_get_symbol:n { ##1 }
4122                 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {

```

```

4123         \l_stex_get_symbol_uri_str
4124     }
4125 }
4126 }
4127 \exp_args:Nnnx
4128 \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}
4129 \str_if_empty:NF \sassertiontype {
4130     \stex_annotate_invisible:nnn{type}{\sassertiontype}{ }
4131 }
4132 \clist_set:Nn \l_tmpa_clist \sassertiontype
4133 \tl_clear:N \l_tmpa_tl
4134 \clist_map_inline:Nn \l_tmpa_clist {
4135     \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
4136         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
4137     }
4138 }
4139 \tl_if_empty:NTF \l_tmpa_tl {
4140     \__stex_statements_sassertion_start:
4141 }{
4142     \l_tmpa_tl
4143 }
4144 }
4145 \stex_ref_new_doc_target:n \sassertionid
4146 }{
4147     \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
4148     \stex_if_smsmode:F {
4149         \clist_set:Nn \l_tmpa_clist \sassertiontype
4150         \tl_clear:N \l_tmpa_tl
4151         \clist_map_inline:Nn \l_tmpa_clist {
4152             \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
4153                 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
4154             }
4155         }
4156         \tl_if_empty:NTF \l_tmpa_tl {
4157             \__stex_statements_sassertion_end:
4158         }{
4159             \l_tmpa_tl
4160         }
4161     }
4162 }
4163 }

```

\stexpatchassertion

```

4164
4165 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
4166     \par\noindent\titllemph{Assertion~\tl_if_empty:NF \sassertiontitle {
4167         (\sassertiontitle)
4168     }~}
4169 }
4170 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
4171
4172 \newcommand\stexpatchassertion[3] [] {
4173     \str_set:Nx \l_tmpa_str{ #1 }
4174     \str_if_empty:NTF \l_tmpa_str {

```

```

4175     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
4176     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
4177   }{
4178     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
4179     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
4180   }
4181 }

```

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

`\inlineass` inline:

```

4182 \keys_define:nn {stex / inlineass }{
4183   type      .str_set_x:N = \sassertiontype,
4184   id        .str_set_x:N = \sassertionid,
4185   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4186   name      .str_set_x:N = \sassertionname
4187 }
4188 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
4189   \str_clear:N \sassertiontype
4190   \str_clear:N \sassertionid
4191   \str_clear:N \sassertionname
4192   \clist_clear:N \l__stex_statements_sassertion_for_clist
4193   \keys_set:nn { stex / inlineass }{ #1 }
4194 }
4195 \NewDocumentCommand \inlineass { 0{} m } {
4196   \begin{group}
4197     \__stex_statements_inlineass_args:n{ #1 }
4198     \stex_ref_new_doc_target:n \sassertionid
4199     \stex_if_smsmode:TF{
4200       \stex_smsmode_set_codes:
4201       \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
4202     }{
4203       \seq_clear:N \l_tmpa_seq
4204       \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4205         \str_if_eq:nnF{ ##1 }{ }{
4206           \stex_get_symbol:n { ##1 }
4207           \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4208             \l_stex_get_symbol_uri_str
4209           }
4210         }
4211       }
4212       \exp_args:Nnx
4213       \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{
4214         \str_if_empty:NF \sassertiontype {
4215           \stex_annotate_invisible:nnn{type}{\sassertiontype}{ }
4216         }
4217         #2
4218         \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
4219       }
4220     }
4221   \end{group}
4222 }

```

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

## 33.3 Examples

sexample

```

4223
4224 \keys_define:nn {stex / sexample }{
4225   type      .str_set_x:N = \exampletype,
4226   id        .str_set_x:N = \sexampleid,
4227   title     .tl_set:N     = \sexamplename,
4228   for       .clist_set:N  = \l__stex_statements_sexample_for_clist,
4229 }
4230 \cs_new_protected:Nn \l__stex_statements_sexample_args:n {
4231   \str_clear:N \sexampletype
4232   \str_clear:N \sexampleid
4233   \tl_clear:N \sexamplename
4234   \clist_clear:N \l__stex_statements_sexample_for_clist
4235   \keys_set:nn { stex / sexample }{ #1 }
4236 }
4237
4238 \NewDocumentEnvironment{sexample}{0{}}{
4239   \l__stex_statements_sexample_args:n{ #1 }
4240   \stex_smsmode_set_codes:
4241   \stex_if_smsmode:TF {
4242     \stex_smsmode_set_codes:
4243   } {
4244     \seq_clear:N \l_tmpa_seq
4245     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4246       \str_if_eq:nnF{ ##1 }{{}{
4247         \stex_get_symbol:n { ##1 }
4248         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4249           \l_stex_get_symbol_uri_str
4250         }
4251       }
4252     }
4253     \exp_args:Nnnx
4254     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
4255     \str_if_empty:NF \sexampletype {
4256       \stex_annotate_invisible:nnn{type}{\sexampletype}{
4257     }
4258     \clist_set:Nn \l_tmpa_clist \sexampletype
4259     \tl_clear:N \l_tmpa_tl
4260     \clist_map_inline:Nn \l_tmpa_clist {
4261       \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
4262         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
4263       }
4264     }
4265     \tl_if_empty:NTF \l_tmpa_tl {
4266       \l__stex_statements_sexample_start:
4267     }{
4268       \l_tmpa_tl
4269     }
4270   }
4271   \stex_ref_new_doc_target:n \sexampleid
4272 }{
4273   \str_if_empty:NF \sexamplename { \symdecl*{\sexamplename} }

```

```

4274 \stex_if_smsmode:F {
4275   \clist_set:No \l_tmpa_clist \sexamplotype
4276   \tl_clear:N \l_tmpa_tl
4277   \clist_map_inline:Nn \l_tmpa_clist {
4278     \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
4279       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
4280     }
4281   }
4282   \tl_if_empty:NTF \l_tmpa_tl {
4283     \__stex_statements_sexample_end:
4284   }{
4285     \l_tmpa_tl
4286   }
4287   \end{stex_annotate_env}
4288 }
4289 }

```

\stexpatchexample

```

4290
4291 \cs_new_protected:Nn \__stex_statements_sexample_start: {
4292   \par\noindent\titllemph{Example~\tl_if_empty:NF \sexampltitle {
4293     (\sexampltitle)
4294   }~}
4295 }
4296 \cs_new_protected:Nn \__stex_statements_sexample_end: {\par\medskip}
4297
4298 \newcommand\stexpatchexample[3]{} {
4299   \str_set:Nx \l_tmpa_str{ #1 }
4300   \str_if_empty:NTF \l_tmpa_str {
4301     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
4302     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
4303   }{
4304     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
4305     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
4306   }
4307 }

```

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

\inlineex inline:

```

4308 \keys_define:nn {stex / inlineex }{
4309   type      .str_set_x:N = \sexamplotype,
4310   id        .str_set_x:N = \sexampleid,
4311   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
4312   name      .str_set_x:N = \sexamplename
4313 }
4314 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
4315   \str_clear:N \sexamplotype
4316   \str_clear:N \sexampleid
4317   \str_clear:N \sexamplename
4318   \clist_clear:N \l__stex_statements_sexample_for_clist
4319   \keys_set:nn { stex / inlineex }{ #1 }
4320 }
4321 \NewDocumentCommand \inlineex { 0{} m } {

```

```

4322 \begingroup
4323 \_stex_statements_inlineex_args:n{ #1 }
4324 \stex_ref_new_doc_target:n \sexampleid
4325 \stex_if_smsmode:TF{
4326   \stex_smsmode_set_codes:
4327   \str_if_empty:NF \sexamplename { \symdecl*{\examplename} }
4328 }{
4329   \seq_clear:N \l_tmpa_seq
4330   \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4331     \str_if_eq:nnF{ ##1 }{}{
4332       \stex_get_symbol:n { ##1 }
4333       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4334         \l_stex_get_symbol_uri_str
4335       }
4336     }
4337   }
4338   \exp_args:Nnx
4339   \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpa_seq {},,}{
4340     \str_if_empty:NF \sexamplotype {
4341       \stex_annotate_invisible:nnn{type}{\sexamplotype}{
4342     }
4343     #2
4344     \str_if_empty:NF \sexamplename { \symdecl*{\sexamplename} }
4345   }
4346 }
4347 \endgroup
4348 }

```

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

## 33.4 Logical Paragraphs

sparagraph

```

4349 \keys_define:nn { stex / sparagraph } {
4350   id      .str_set_x:N = \sparagraphid ,
4351   title   .tl_set:N    = \l_stex_sparagraph_title_tl ,
4352   type    .str_set_x:N = \sparagraphtype ,
4353   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
4354   from    .tl_set:N    = \sparagraphfrom ,
4355   to      .tl_set:N    = \sparagraphto ,
4356   start   .tl_set:N    = \l_stex_sparagraph_start_tl ,
4357   name    .str_set:N    = \sparagraphname
4358 }
4359
4360 \cs_new_protected:Nn \stex_sparagraph_args:n {
4361   \tl_clear:N \l_stex_sparagraph_title_tl
4362   \tl_clear:N \sparagraphfrom
4363   \tl_clear:N \sparagraphto
4364   \tl_clear:N \l_stex_sparagraph_start_tl
4365   \str_clear:N \sparagraphid
4366   \str_clear:N \sparagraphtype
4367   \clist_clear:N \l__stex_statements_sparagraph_for_clist
4368   \str_clear:N \sparagraphname

```



```

4369 \keys_set:nn { stex / sparagraph }{ #1 }
4370 }
4371 \newif\if@in@omtext\@in@omtextfalse
4372
4373 \NewDocumentEnvironment {sparagraph} { 0{} } {
4374 \stex_sparagraph_args:n { #1 }
4375 \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4376 \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
4377 }{
4378 \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
4379 }
4380 \@in@omtexttrue
4381 \stex_if_smsmode:TF {
4382 \stex_smsmode_set_codes:
4383 } {
4384 \seq_clear:N \l_tmpa_seq
4385 \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
4386 \str_if_eq:nnF{ ##1 }{ }{
4387 \stex_get_symbol:n { ##1 }
4388 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4389 \l_stex_get_symbol_uri_str
4390 }
4391 }
4392 }
4393 \exp_args:Nnnx
4394 \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
4395 \str_if_empty:NF \sparagraphtype {
4396 \stex_annotate_invisible:nnn{type}{\sparagraphtype}{ }
4397 }
4398 \str_if_empty:NF \sparagraphfrom {
4399 \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{ }
4400 }
4401 \str_if_empty:NF \sparagraphto {
4402 \stex_annotate_invisible:nnn{to}{\sparagraphto}{ }
4403 }
4404 }
4405 \clist_set:N \l_tmpa_clist \sparagraphtype
4406 \tl_clear:N \l_tmpa_tl
4407 \clist_map_inline:Nn \sparagraphtype {
4408 \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
4409 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
4410 }
4411 }
4412 \tl_if_empty:NTF \l_tmpa_tl {
4413 \__stex_statements_sparagraph_start:
4414 }{
4415 \l_tmpa_tl
4416 }
4417 \stex_ref_new_doc_target:n \sparagraphid
4418 \ignorespacesandpars
4419 }{
4420 \clist_set:N \l_tmpa_clist \sparagraphtype
4421 \tl_clear:N \l_tmpa_tl
4422 \clist_map_inline:Nn \l_tmpa_clist {

```

```

4423 \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
4424 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
4425 }
4426 }
4427 \str_if_empty:NF \sparagraphname { \symdecl*{\sparagraphname} }
4428 \tl_if_empty:NTF \l_tmpa_tl {
4429 \__stex_statements_sparagraph_end:
4430 }{
4431 \l_tmpa_tl
4432 }
4433 \stex_if_smsmode:F {
4434 \end{stex_annotate_env}
4435 }
4436 }

```

# \stexpatchparagraph

```

4437
4438 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
4439 \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4440 \tl_if_empty:NF \l_stex_sparagraph_title_tl {
4441 \titleemph{\l_stex_sparagraph_title_tl}:~
4442 }
4443 }{
4444 \titleemph{\l_stex_sparagraph_start_tl}~
4445 }
4446 }
4447 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
4448
4449 \newcommand\stexpatchparagraph[3] [] {
4450 \str_set:Nx \l_tmpa_str{ #1 }
4451 \str_if_empty:NTF \l_tmpa_str {
4452 \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
4453 \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
4454 }{
4455 \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
4456 \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
4457 }
4458 }
4459
4460 \keys_define:nn { stex / inlinepara} {
4461 id .str_set_x:N = \sparagraphid ,
4462 type .str_set_x:N = \sparagraphtype ,
4463 for .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
4464 from .tl_set:N = \sparagraphfrom ,
4465 to .tl_set:N = \sparagraphto ,
4466 name .str_set:N = \sparagraphname
4467 }
4468 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
4469 \tl_clear:N \sparagraphfrom
4470 \tl_clear:N \sparagraphto
4471 \str_clear:N \sparagraphid
4472 \str_clear:N \sparagraphtype
4473 \clist_clear:N \l__stex_statements_sparagraph_for_clist
4474 \str_clear:N \sparagraphname

```

```

4475 \keys_set:nn { stex / inlinepara }{ #1 }
4476 }
4477 \NewDocumentCommand \inlinepara { 0{} m } {
4478   \beginngroup
4479   \__stex_statements_inlinepara_args:n{ #1 }
4480   \stex_ref_new_doc_target:n \sparagraphid
4481   \stex_if_smsmode:TF{
4482     \stex_smsmode_set_codes:
4483     \str_if_empty:NF \sparagraphname { \symdecl*{\sparagraphname} }
4484   }{
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:Nnx
4495     \stex_annotate:nnn{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       #2
4506       \str_if_empty:NF \sparagraphname { \symdecl*{\sparagraphname} }
4507     }
4508   }
4509   \endgroup
4510 }
4511

```

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

**symboldoc**

```

4512 \NewDocumentEnvironment{symboldoc}{ m }{
4513   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
4514   \seq_clear:N \l_tmpb_seq
4515   \seq_map_inline:Nn \l_tmpa_seq {
4516     \str_if_eq:nnF{ ##1 }{}{
4517       \stex_get_symbol:n { ##1 }
4518       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
4519         \l_stex_get_symbol_uri_str
4520       }
4521     }
4522   }
4523   \par
4524   \exp_args:Nnx

```

```

4525 \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
4526 }{
4527 \end{stex_annotate_env}
4528 }

4529 \end{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>

```
4530 \package
4531 \@@=stex_sproof
4532
4533 %%%%%%%%%%%%%%%%% sproof.dtx %%%%%%%%%%%%%%%%%
4534
```

### 34.2 Proofs

We first define some keys for the proof environment.

```
4535 \keys_define:nn { stex / spf } {
4536   id          .str_set:N = \l__stex_sproof_spf_id_str,
4537   display     .tl_set:N  = \l__stex_sproof_spf_display_tl,
4538   for         .tl_set:N  = \l__stex_sproof_spf_for_tl ,
4539   from        .tl_set:N  = \l__stex_sproof_spf_from_tl ,
4540   proofend    .tl_set:N  = \l__stex_sproof_spf_proofend_tl,
4541   type        .tl_set:N  = \l__stex_sproof_spf_type_tl,
4542   title       .tl_set:N  = \l__stex_sproof_spf_title_tl,
4543   continues   .tl_set:N  = \l__stex_sproof_spf_continues_tl,
4544   functions   .tl_set:N  = \l__stex_sproof_spf_functions_tl,
4545   method      .tl_set:N  = \l__stex_sproof_spf_method_tl
4546 }
4547 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
4548   \str_clear:N \l__stex_sproof_spf_id_str
4549   \tl_clear:N \l__stex_sproof_spf_display_tl
4550   \tl_clear:N \l__stex_sproof_spf_for_tl
4551   \tl_clear:N \l__stex_sproof_spf_from_tl
4552   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
4553   \tl_clear:N \l__stex_sproof_spf_type_tl
4554   \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

```

4555 \tl_clear:N \l__stex_sproof_spf_continues_tl
4556 \tl_clear:N \l__stex_sproof_spf_functions_tl
4557 \tl_clear:N \l__stex_sproof_spf_method_tl
4558 \keys_set:nn { stex / spf }{ #1 }
4559 }

```

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

```

4560 \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.

```

4561 \newcount\count_ten
4562 \newenvironment{pst@with@label}[1]{
4563   \edef\pst@label{#1}
4564   \advance\count_ten by 1\relax
4565   \count_ten=1
4566 }{
4567   \advance\count_ten by -1\relax
4568 }

```

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

```

4569 \def\the@pst@label{
4570   \pst@make@label\pst@label{\number\count_ten}\l__stex_sproof_pstlabel_postfix_tl
4571 }

```

*(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.

```

4572 \keys_define:nn { stex / pstlabel }{
4573   prefix      .tl_set:N   = \l__stex_sproof_pstlabel_prefix_tl,
4574   delimiter   .tl_set:N   = \l__stex_sproof_pstlabel_delimiter_tl,
4575   postfix     .tl_set:N   = \l__stex_sproof_pstlabel_postfix_tl
4576 }
4577 \cs_new_protected:Nn \__stex_sproof_pstlabel_args:n {

```

---

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

```

4578 \tl_set:Nn \l__stex_sproof_pstlabel_prefix_tl {P}
4579 \tl_set:Nn \l__stex_sproof_pstlabel_delimiter_tl {.}
4580 \tl_clear:N \l__stex_sproof_pstlabel_postfix_tl
4581 }
4582 \__stex_sproof_pstlabel_args:n {}
4583 \newcommand\setpstlabelstyle[1]{
4584   \__stex_sproof_pstlabel_args:n {#1}
4585 }
4586 \newcommand\setpstlabelstyledefault{%
4587   \__stex_sproof_pstlabel_args:n{prefix=P,delimiter=.,postfix={}}
4588 }

```

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

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

```

4589 \ExplSyntaxOff
4590 \def\pst@make@label@long#1#2{\@for\@I:=#1\do{\expandafter\expandafter\expandafter\@I\csname
4591 \def\pst@make@label@angles#1#2{\ensuremath{\@for\@I:=#1\do{\rangle}}#2}
4592 \def\pst@make@label@short#1#2{#2}
4593 \def\pst@make@label@empty#1#2{}
4594 \ExplSyntaxOn
4595 \def\pstlabelstyle#1{%
4596   \def\pst@make@label{\use:c{pst@make@label@#1}}%
4597 }%
4598 \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.

```

4599 \def\next@pst@label{%
4600   \global\advance\count\count10 by 1%
4601 }%

```

(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

```

4602 \def\sproof@box{
4603   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
4604 }
4605 \def\spf@proofend{\sproof@box}
4606 \def\sproofend{
4607   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
4608     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
4609   }
4610 }
4611 \def\sProofEndSymbol#1{\def\sproof@box{#1}}

```

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

**spf@\*@kw**

```

4612 \def\spf@proofsketch@kw{Proof Sketch}
4613 \def\spf@proof@kw{Proof}
4614 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

4615 \AddToHook{begindocument}{
4616   \ltx@ifpackageloaded{babel}{
4617     \makeatletter
4618     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4619     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4620       \input{sproof-ngerman.ldf}
4621     }
4622     \clist_if_in:NnT \l_tmpa_clist {finnish}{
4623       \input{sproof-finnish.ldf}
4624     }
4625     \clist_if_in:NnT \l_tmpa_clist {french}{
4626       \input{sproof-french.ldf}
4627     }
4628     \clist_if_in:NnT \l_tmpa_clist {russian}{
4629       \input{sproof-russian.ldf}
4630     }
4631     \makeatother
4632   }{}
4633 }

```

`spfsketch`

```

4634 \newcommand\spfsketch[2][]{
4635   \__stex_sproof_spf_args:n{#1}
4636   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4637     \titleemph{
4638       \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {
4639         \spf@proofsketch@kw
4640       }{
4641         \l__stex_sproof_spf_type_tl
4642       }
4643     }:
4644   }
4645   {~#2}
4646   %\sref@label@id{this \ifx\spf@type\@empty\spf@proofsketch@kw\else\spf@type\fi}
4647   \sproofend
4648 }

```

(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>

```

4649 \newenvironment{spfeq}[2][]{
4650   \__stex_sproof_spf_args:n{#1}
4651   %\sref@target
4652   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4653     \titleemph{
4654       \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {
4655         \spf@proof@kw
4656       }{

```

<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



```

4657         \l__stex_sproof_spf_type_tl
4658     }
4659     }:
4660 }
4661 {-#2}
4662 \begin{displaymath}\begin{array}{rcll}
4663 }{
4664 \end{array}\end{displaymath}
4665 }

```

(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.

```

4666 \newenvironment{spf@proof}[2] []{
4667   \l__stex_sproof_spf_args:n{#1}
4668   %\sref@target
4669   \count_ten=10
4670   \par\noindent
4671   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4672     \titleemph{
4673       \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {
4674         \spf@proof@kw
4675       }{
4676         \l__stex_sproof_spf_type_tl
4677       }
4678     }:
4679   }
4680   {-#2}
4681   %\sref@label@id{this \ifx\spf@type\empty\spf@proof@kw\else\spf@type\fi}
4682   \def\pst@label{}
4683   \newcount\pst@count% initialize the labeling mechanism
4684   \begin{description}\begin{pst@with@label}{\l__stex_sproof_pstlabel_prefix_tl}
4685   }{
4686     \end{pst@with@label}\end{description}
4687   }
4688   \newenvironment{sproof}[2] []{\begin{spf@proof}[#1]{#2}}{\sproofend\end{spf@proof}}
4689   \newenvironment{sProof}[2] []{\begin{spf@proof}[#1]{#2}}{\end{spf@proof}}

```

**\spfidea**

```

4690 \newcommand\spfidea[2] []{
4691   \l__stex_sproof_spf_args:n{#1}
4692   \titleemph{
4693     \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {Proof~Idea}{
4694       \l__stex_sproof_spf_type_tl
4695     }:
4696   }-#2
4697   \sproofend
4698 }

```

(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
4699 \newenvironment{spfstep}[1][]{
4700   \_stex_sproof_spf_args:n{#1}
4701   \@in@omtexttrue
4702   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4703     \item[\the@pst@label]
4704   }
4705   \tl_if_empty:NF \l__stex_sproof_spf_title_tl {
4706     {(\titleemph{\l__stex_sproof_spf_title_tl})\enspace}
4707   }
4708   %\sref@label@id{\pst@label}
4709   \ignorespacesandpars
4710 }{
4711   \next@pst@label\ignorespacesandpars
4712 }

```

**sproofcomment**

```

4713 \newenvironment{sproofcomment}[1][]{
4714   \_stex_sproof_spf_args:n{#1}
4715   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4716     \item[\the@pst@label]
4717   }
4718 }{
4719   \next@pst@label
4720 }

```

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.

```

4721 \newenvironment{subproof}[2][]{
4722   \_stex_sproof_spf_args:n{#1}
4723   \def\@test{#2}
4724   \ifx\@test\empty\else
4725     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4726       \item[\the@pst@label]
4727     }{#2}
4728   \fi
4729   \begin{pst@with@label}{\pst@label,\number\count_ten}
4730 }{
4731   \end{pst@with@label}\next@pst@label
4732 }

```

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

```

4733 \newenvironment{spfcases}[2][]{
4734   \def\@test{#1}
4735   \ifx\@test\empty
4736     \begin{subproof}[method=by-cases]{#2}

```

---

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

```

4737 \else
4738   \begin{subproof}[#1,method=by-cases]{#2}
4739 \fi
4740 }{
4741   \end{subproof}
4742 }

```

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

```

4743 \newenvironment{spfcase}[2] [] {
4744   \__stex_sproof_spf_args:n{#1}
4745   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4746     \item[\the@pst@label]
4747   }
4748   \def\@test{#2}
4749   \ifx\@test\@empty
4750   \else
4751     {\titleemph{#2}:~}
4752   \fi
4753   \begin{pst@with@label}{\pst@label,\number\count_ten}
4754 }{
4755   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4756     \sproofend
4757   }
4758   \end{pst@with@label}
4759   \next@pst@label
4760 }

```

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

```

4761 \newcommand\spfcasesketch[3] [] {
4762   \__stex_sproof_spf_args:n{#1}
4763   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4764     \item[\the@pst@label]
4765   }
4766   \def\@test{#2}
4767   \ifx\@test\@empty
4768   \else
4769     {\titleemph{#2}:~}
4770   \fi#3
4771   \next@pst@label
4772 }%

```

### 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.

```

4773 \keys_define:nn { stex / just }{
4774   id      .str_set:x:N = \l__stex_sproof_just_id_str,
4775   method  .tl_set:N    = \l__stex_sproof_just_method_tl,
4776   premises .tl_set:N    = \l__stex_sproof_just_premises_tl,
4777   args    .tl_set:N    = \l__stex_sproof_just_args_tl
4778 }

```

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`

`4779 \newenvironment{justification}[1] [] {}{}`

`\premise`

`4780 \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.

`4781 \newcommand\justarg[2] [] {#2}`

`4782 \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

```
4783 <*package>
4784
4785 %%%%%%%%%% others.dtx %%%%%%%%%%
4786
4787 <@@=stex_others>
      Warnings and error messages
4788 % None

\MSC Math subject classifier

4789 \NewDocumentCommand \MSC {m} {
4790   % TODO
4791 }

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

4792 \@ifpackageloaded{tikzinput}{
4793   \RequirePackage{stex-tikzinput}
4794 }{}
4795 </package>
```

## Chapter 36

# STEX -Metatheory Implementation

```
4796 \*package>
4797 \@@=stex_modules>
4798
4799 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
4800
4801 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
4802 \begingroup
4803 \stex_module_setup:nn{
4804   ns=\c_stex_metatheory_ns_str,
4805   meta=NONE
4806 }{Metatheory}
4807 \stex_reactivate_macro:N \symdecl
4808 \stex_reactivate_macro:N \notation
4809 \stex_reactivate_macro:N \symdef
4810 \ExplSyntaxOff
4811 \csname stex_suppress_html:n\endcsname{
4812   % is-a (a:A, a \in A, a is an A, etc.)
4813   \symdecl[args=ai]{isa}
4814   \notation[typed]{isa}{#1 \comp{:} #2}{#1 \comp, #2}
4815   \notation[in]{isa}{#1 \comp\in #2}{#1 \comp, #2}
4816   \notation[pred]{isa}{#2\comp(#1 \comp)}{#1 \comp, #2}
4817
4818   % bind (\forall, \Pi, \lambda etc.)
4819   \symdecl[args=Bi]{bind}
4820   \notation[forall]{bind}{\comp\forall #1.\;#2}{#1 \comp, #2}
4821   \notation[\Pi]{bind}{\comp\prod_{#1}#2}{#1 \comp, #2}
4822   \notation[deffun]{bind}{\comp( #1 \comp)\; \to\;}{#1 \comp, #2}
4823
4824   % dummy variable
4825   \symdecl{dummyvar}
4826   \notation[underscore]{dummyvar}{\comp\_}
4827   \notation[dot]{dummyvar}{\comp\cdot}
4828   \notation[dash]{dummyvar}{\comp{\rm --}}
4829
4830   %fromto (function space, Hom-set, implication etc.)
```

```

4831 \symdecl[args=ai]{fromto}
4832 \notation[xarrow]{fromto}{#1 \comp\to #2}{#1 \comp\times #2}
4833 \notation[arrow]{fromto}{#1 \comp\to #2}{#1 \comp\to #2}
4834
4835 % mapto (lambda etc.)
4836 %\symdecl[args=Bi]{mapto}
4837 %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
4838 %\notation[lambda]{mapto}{\comp\lambda #1 \comp. \; #2}{#1 \comp, #2}
4839 %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp. \; #2}{#1 \comp, #2}
4840
4841 % function/operator application
4842 \symdecl[args=ia]{apply}
4843 \notation[prec=0;0x\infpres,parens]{apply}{#1 \comp( #2 \comp)}{#1 \comp, #2}
4844 \notation[prec=0;0x\infpres,lambda]{apply}{#1 \; #2 }{#1 \; #2}
4845
4846 % ‘‘type’’ of all collections (sets, classes, types, kinds)
4847 \symdecl{collection}
4848 \notation[U]{collection}{\comp{\mathcal{U}}}
4849 \notation[set]{collection}{\comp{\textsf{Set}}}
4850
4851 % sequences
4852 \symdecl[args=1]{seqtype}
4853 \notation[kleene]{seqtype}{#1^{\comp\ast}}
4854
4855 \symdef[args=2,li,prec=nobrackets]{sequence-index}{#1}_{#2}
4856 \notation[ui,prec=nobrackets]{sequence-index}{#1}^{\sim #2}
4857
4858 %\symdef[args=3,li]{sequence-from-to}{#1}_{#2}\comp{\,\ellipses,}#1_{#3}
4859 %\notation[ui]{sequence-from-to}{#1}^{\sim #2}\comp{\,\ellipses,}#1^{\sim #3}
4860 % ^ superceded by \aseqfromto and \livar/\uivar
4861
4862 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{\,\ellipses}}{#1\comp,#2}
4863 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{\,\ellipses,}#2}{#1\comp,#2}
4864 \symdef[args=aui,prec=nobrackets]{aseqfromtovia}{#1\comp{\,\ellipses,}#2\comp{\,\ellipses,}#3}
4865
4866 % letin (‘‘let’’, local definitions, variable substitution)
4867 \symdecl[args=bii]{letin}
4868 \notation[let]{letin}{\comp{\rm let}}{\;#1\comp{=}#2\; \comp{\rm in}}{\;#3}
4869 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
4870 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
4871
4872 % structures
4873 \symdecl*[args=1]{module-type}
4874 \notation{module-type}{\mathtt{MOD} #1}
4875 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
4876 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{#1 \comp, #2}
4877
4878 }
4879 \ExplSyntaxOn
4880 \stex_add_to_current_module:n{
4881   \let\nappa\apply
4882   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
4883   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
4884   \def\livar{\csname sequence-index\endcsname[li]}

```

```

4885 \def\uivar{\csname sequence-index\endcsname[ui]}
4886 \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
4887 \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
4888 \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
4889 }
4890 \__stex_modules_end_module:
4891 \endgroup
4892 \</package>

```



## Chapter 37

# Tikzinput Implementation

```
4893 <*package>
4894
4895 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
4896
4897 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
4898 \RequirePackage{l3keys2e}
4899
4900 \keys_define:nn { tikzinput } {
4901   image .bool_set:N = \c_tikzinput_image_bool,
4902   image .default:n = false ,
4903   unknown .code:n = {}
4904 }
4905
4906 \ProcessKeysOptions { tikzinput }
4907
4908 \bool_if:NTF \c_tikzinput_image_bool {
4909   \RequirePackage{graphicx}
4910
4911   \providecommand\usetikzlibrary[]{}
4912   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
4913 }{
4914   \RequirePackage{tikz}
4915   \RequirePackage{standalone}
4916
4917   \newcommand \tikzinput [2] [] {
4918     \setkeys{Gin}{#1}
4919     \ifx \Gin@ewidth \Gin@exclamation
4920       \ifx \Gin@eheight \Gin@exclamation
4921         \input { #2 }
4922       \else
4923         \resizebox{!}{ \Gin@eheight }{
4924           \input { #2 }
4925         }
4926       \fi
4927     \else
4928       \ifx \Gin@eheight \Gin@exclamation
4929         \resizebox{ \Gin@ewidth }{!}{
4930           \input { #2 }
4931         }
4932       \else
4933         \input { #2 }
4934       \fi
4935     \fi
4936   }
4937 }
```

```

4931     }
4932     \else
4933         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
4934             \input { #2 }
4935         }
4936     \fi
4937 \fi
4938 }
4939 }
4940
4941 \newcommand \ctikzinput [2] [] {
4942     \begin{center}
4943         \tikzinput [ #1 ] { #2 }
4944     \end{center}
4945 }
4946
4947 \@ifpackageloaded{stex}{
4948     \RequirePackage{stex-tikzinput}
4949 }{}
4950
4951 </package>
4952 <*stex>
4953 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
4954 \RequirePackage{stex}
4955 \RequirePackage{tikzinput}
4956
4957 \newcommand\mhtikzinput [2] [] {%
4958     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
4959     \stex_in_repository:nn\Gin@mhrepos{
4960         \tikzinput [ #1 ] {\mhp{##1}{#2}}
4961     }
4962 }
4963 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [ #1 ] { #2 }\end{center}}
4964 </stex>

```

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

## 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.

```
4965 \*cls)
4966 \@@=document_structure)
4967 \ProvidesExplClass{document-structure}{2022/02/10}{3.0}{Modular Document Structure Class}
4968 \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`

```
4969 \keys_define:nn{ document-structure / pkg }{
4970   class      .str_set_x:N = \c_document_structure_class_str,
4971   minimal    .bool_set:N = \c_document_structure_minimal_bool,
4972   report     .code:n      = {
4973     \ClassWarning{document-structure}{the option 'report' is deprecated, use 'class=report',
4974     \str_set:Nn \c_document_structure_class_str {report}
4975   },
4976   book       .code:n      = {
4977     \ClassWarning{document-structure}{the option 'book' is deprecated, use 'class=book', ins
4978     \str_set:Nn \c_document_structure_class_str {book}
4979   },
4980   bookpart   .code:n      = {
4981     \ClassWarning{document-structure}{the option 'bookpart' is deprecated, use 'class=book,t
4982     \str_set:Nn \c_document_structure_class_str {book}
4983     \str_set:Nn \c_document_structure_topsect_str {chapter}
4984   },
```

```

4985 docopt      .str_set_x:N = \c_document_structure_docopt_str,
4986 unknown     .code:n      = {
4987   \PassOptionsToPackage{ \CurrentOption }{ document-structure }
4988 }
4989 }
4990 \ProcessKeysOptions{ document-structure / pkg }
4991 \str_if_empty:NT \c_document_structure_class_str {
4992   \str_set:Nn \c_document_structure_class_str {article}
4993 }
4994 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
4995   {\c_document_structure_class_str}
4996

```

### 38.3 Beefing up the document environment

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

```

4997 \RequirePackage{document-structure}
4998 \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>

```

4999 \keys_define:nn { document-structure / document }{
5000   id .str_set_x:N = \c_document_structure_document_id_str
5001 }
5002 \let\__document_structure_orig_document=\document
5003 \renewcommand{\document}[1][]{
5004   \keys_set:nn{ document-structure / document }{ #1 }
5005   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
5006   \__document_structure_orig_document
5007 }

```

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

```

5008 }
5009 \</cls>

```

### 38.4 Implementation: document-structure Package

```

5010 \<*package>
5011 \ProvidesExplPackage{document-structure}{2022/02/10}{3.0}{Modular Document Structure}
5012 \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

```

5013
5014 \keys_define:nn{ document-structure / pkg }{
5015   class      .str_set_x:N = \c_document_structure_class_str,
5016   topsect    .str_set_x:N = \c_document_structure_topsect_str,
5017   % showignores .bool_set:N = \c_document_structure_showignores_bool,
5018 }
5019 \ProcessKeysOptions{ document-structure / pkg }
5020 \str_if_empty:NT \c_document_structure_class_str {
5021   \str_set:Nn \c_document_structure_class_str {article}
5022 }
5023 \str_if_empty:NT \c_document_structure_topsect_str {
5024   \str_set:Nn \c_document_structure_topsect_str {section}
5025 }

```

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

```

5026 \RequirePackage{xspace}
5027 \RequirePackage{comment}
5028 \AddToHook{begindocument}{
5029   \ltx@ifpackageloaded{babel}{
5030     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5031     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5032       \makeatletter\input{omdoc-ngerman.ldf}\makeatother
5033     }
5034   }{}
5035 }

```

`\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.

```

5036 \int_new:N \l_document_structure_section_level_int
5037 \str_case:VnF \c_document_structure_topsect_str {
5038   {part}}{
5039     \int_set:Nn \l_document_structure_section_level_int {0}
5040   }
5041   {chapter}}{
5042     \int_set:Nn \l_document_structure_section_level_int {1}
5043   }
5044 }{
5045   \str_case:VnF \c_document_structure_class_str {
5046     {book}}{
5047       \int_set:Nn \l_document_structure_section_level_int {0}
5048     }
5049     {report}}{
5050       \int_set:Nn \l_document_structure_section_level_int {0}
5051     }
5052   }{
5053     \int_set:Nn \l_document_structure_section_level_int {2}
5054   }
5055 }

```

## 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

```
5056 \def\current@section@level{document}%
5057 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
5058 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
5059 \cs_new_protected:Npn \skipomgroup {
5060   \ifcase\l_document_structure_section_level_int
5061   \or\stepcounter{part}
5062   \or\stepcounter{chapter}
5063   \or\stepcounter{section}
5064   \or\stepcounter{subsection}
5065   \or\stepcounter{subsubsection}
5066   \or\stepcounter{paragraph}
5067   \or\stepcounter{subparagraph}
5068   \fi
5069 }
```

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

`blindomgroup`

```
5070 \newcommand\at@begin@blindomgroup[1]{%
5071 \newenvironment{blindomgroup}
5072 {
5073   \int_incr:N\l_document_structure_section_level_int
5074   \at@begin@blindomgroup\l_document_structure_section_level_int
5075 }{}}
```

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

```
5076 \newcommand\omgroup@nonum[2]{
5077   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
5078   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
5079 }
```

*(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.

```
5080 \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.

```

5081 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
5082   \@nameuse{#1}{#2}
5083 }{
5084   \cs_if_exist:NTF\rdfmata@sectioning{
5085     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
5086   }{
5087     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
5088   }
5089 }
5090 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
5091 }

```

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

omgroup

```

5092 \keys_define:nn { document-structure / omgroup }{
5093   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
5094   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
5095   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
5096   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
5097   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
5098   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
5099   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
5100   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
5101   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
5102   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
5103 }
5104 \cs_new_protected:Nn \__document_structure_omgroup_args:n {
5105   \str_clear:N \l__document_structure_omgroup_id_str
5106   \str_clear:N \l__document_structure_omgroup_date_str
5107   \clist_clear:N \l__document_structure_omgroup_creators_clist
5108   \clist_clear:N \l__document_structure_omgroup_contributors_clist
5109   \tl_clear:N \l__document_structure_omgroup_srccite_tl
5110   \tl_clear:N \l__document_structure_omgroup_type_tl
5111   \tl_clear:N \l__document_structure_omgroup_short_tl
5112   \tl_clear:N \l__document_structure_omgroup_display_tl
5113   \tl_clear:N \l__document_structure_omgroup_intro_tl
5114   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
5115   \keys_set:nn { document-structure / omgroup } { #1 }
5116 }

```

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.

```

5117 \newif\if@mainmatter\@mainmattertrue
5118 \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.

```

5119 \keys_define:nn { document-structure / sectioning }{
5120   name .str_set_x:N = \l__document_structure_sect_name_str ,
5121   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
5122   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
5123   num .bool_set:N = \l__document_structure_sect_num_bool ,
5124 }

```

```

5125 \cs_new_protected:Nn \__document_structure_sect_args:n {
5126   \str_clear:N \l__document_structure_sect_name_str
5127   \str_clear:N \l__document_structure_sect_ref_str
5128   \bool_set_false:N \l__document_structure_sect_clear_bool
5129   \bool_set_false:N \l__document_structure_sect_num_bool
5130   \keys_set:nn { document-structure / sectioning } { #1 }
5131 }
5132 \newcommand\omdoc@sectioning[3][]{
5133   \__document_structure_sect_args:n {#1}
5134   \let\omdoc@sect@name\l__document_structure_sect_name_str
5135   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
5136   \if@mainmatter% numbering not overridden by frontmatter, etc.
5137     \bool_if:NTF \l__document_structure_sect_num_bool {
5138       \omgroup@num{#2}{#3}
5139     }{
5140       \omgroup@nonum{#2}{#3}
5141     }
5142     \def\current@section@level{\omdoc@sect@name}
5143   \else
5144     \omgroup@nonum{#2}{#3}
5145   \fi
5146 }% 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.

```

5147 \newcommand\omgroup@redefine@addtocontents[1]{%
5148   %\edef\__document_structureimport{#1}%
5149   %\@for\@I:=\__document_structureimport\do{%
5150     %\edef\@path{\csname module@\@I @path\endcsname}%
5151     %\@ifundefined{tf@toc}\relax%
5152     % {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
5153   %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
5154   %\def\addcontentsline##1##2##3{%
5155     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
5156   %\else% hyperref.sty not loaded
5157   %\def\addcontentsline##1##2##3{%
5158     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
5159   %\fi
5160 }% hyperref.sty loaded?

```

now the `omgroup` environment itself. This takes care of the table of contents via the helper macro above and then selects the appropriate sectioning command from `article.cls`. It also registers the current level of `omgroups` in the `\omgroup@level` counter.

```

5161 \int_new:N \l_document_structure_omgroup_level_int
5162 \newenvironment{omgroup}[2][]{% keys, title
5163 {
5164   \__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.

```

5165 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
5166   \omgroup@redefine@addtocontents{
5167     %\@ifundefined{module@id}\used@modules%
5168     %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}

```



```

5169     }
5170 }

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

5171 \int_incr:N \l_document_structure_omgroup_level_int
5172 \int_incr:N \l_document_structure_section_level_int
5173 \ifcase\l_document_structure_section_level_int
5174   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
5175   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
5176   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
5177   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
5178   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
5179   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
5180   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
5181 \fi
5182 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
5183 \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
5184 }% for customization
5185 {}

```

and finally, we localize the sections

```

5186 \newcommand\omdoc@part@kw{Part}
5187 \newcommand\omdoc@chapter@kw{Chapter}
5188 \newcommand\omdoc@section@kw{Section}
5189 \newcommand\omdoc@subsection@kw{Subsection}
5190 \newcommand\omdoc@subsubsection@kw{Subsubsection}
5191 \newcommand\omdoc@paragraph@kw{paragraph}
5192 \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`

```

5193 \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).

```

5194 \cs_if_exist:NTF\frontmatter{
5195   \let\__document_structure_orig_frontmatter\frontmatter
5196   \let\frontmatter\relax
5197 }{
5198   \tl_set:Nn\__document_structure_orig_frontmatter{
5199     \clearpage
5200     \@mainmatterfalse
5201     \pagenumbering{roman}
5202   }
5203 }

```

```

5204 \cs_if_exist:NTF\backmatter{
5205   \let\__document_structure_orig_backmatter\backmatter
5206   \let\backmatter\relax
5207 }{
5208   \tl_set:Nn\__document_structure_orig_backmatter{
5209     \clearpage
5210     \@mainmatterfalse
5211     \pagenumbering{roman}
5212   }
5213 }

```

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

`frontmatter` we use the `\orig@frontmatter` macro defined above and `\mainmatter` if it exists, otherwise we define it.

```

5214 \newenvironment{frontmatter}{
5215   \__document_structure_orig_frontmatter
5216 }{
5217   \cs_if_exist:NTF\mainmatter{
5218     \mainmatter
5219   }{
5220     \clearpage
5221     \@mainmattertrue
5222     \pagenumbering{arabic}
5223   }
5224 }

```

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

```

5225 \newenvironment{backmatter}{
5226   \__document_structure_orig_backmatter
5227 }{
5228   \cs_if_exist:NTF\mainmatter{
5229     \mainmatter
5230   }{
5231     \clearpage
5232     \@mainmattertrue
5233     \pagenumbering{arabic}
5234   }
5235 }

```

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

5236 \@mainmattertrue\pagenumbering{arabic}

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

```

5237 \def \c__document_structure_document_str{document}
5238 \newcommand\afterprematurestop{}
5239 \def\prematurestop@endomgroup{
5240   \unless\ifx\@currentvir\c__document_structure_document_str
5241     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
5242     \expandafter\prematurestop@endomgroup
5243   \fi
5244 }

```

```

5245 \providecommand\prematurestop{
5246   \message{Stopping~sTeX~processing~prematurely}
5247   \prematurestop@endomgroup
5248   \afterprematurestop
5249   \end{document}
5250 }

```

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

## 38.8 Global Variables

**\setSGvar** set a global variable

```

5251 \RequirePackage{etoolbox}
5252 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

**\useSGvar** use a global variable

```

5253 \newrobustcmd\useSGvar[1]{%
5254   \@ifundefined{sTeX@Gvar@#1}
5255   {\PackageError{document-structure}
5256     {The sTeX Global variable #1 is undefined}
5257     {set it with \protect\setSGvar}}
5258   \@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.

```

5259 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
5260   \@ifundefined{sTeX@Gvar@#1}
5261   {\PackageError{document-structure}
5262     {The sTeX Global variable #1 is undefined}
5263     {set it with \protect\setSGvar}}
5264   {\expandafter\ifx\cename sTeX@Gvar@#1\endcsname\@test #3\fi}}

```

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

## Chapter 39

# NotesSlides – Implementation

### 39.1 Class and Package Options

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

```
5265 \*cls)
5266 \@@=notesslides)
5267 \ProvidesExplClass{notesslides}{2022/02/10}{3.0}{notesslides Class}
5268 \RequirePackage{l3keys2e,expl-keystr-compatible}
5269
5270 \keys_define:nn{notesslides / cls}{
5271   class .code:n = {
5272     \PassOptionsToClass{\CurrentOption}{omdoc}
5273     \str_if_eq:nnT{#1}{book}{
5274       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5275     }
5276     \str_if_eq:nnT{#1}{report}{
5277       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5278     }
5279   },
5280   notes .bool_set:N = \c__notesslides_notes_bool ,
5281   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
5282   unknown .code:n = {
5283     \PassOptionsToClass{\CurrentOption}{omdoc}
5284     \PassOptionsToClass{\CurrentOption}{beamer}
5285     \PassOptionsToPackage{\CurrentOption}{notesslides}
5286   }
5287 }
5288 \ProcessKeysOptions{ notesslides / cls }
5289 \bool_if:NTF \c__notesslides_notes_bool {
5290   \PassOptionsToPackage{notes=true}{notesslides}
5291 }{
5292   \PassOptionsToPackage{notes=false}{notesslides}
5293 }
5294 \</cls)
```

now we do the same for the notesslides package.

```

5295 <*package>
5296 \ProvidesExplPackage{notesslides}{2022/02/10}{3.0}{notesslides Package}
5297 \RequirePackage{13keys2e,expl-keystr-compat}
5298
5299 \keys_define:nn{notesslides / pkg}{
5300   topsect      .str_set_x:N = \c_notesslides_topsect_str,
5301   defaulttopsect .str_set_x:N = \c_notesslides_defaulttopsec_str,
5302   notes        .bool_set:N = \c_notesslides_notes_bool ,
5303   slides       .code:n      = { \bool_set_false:N \c_notesslides_notes_bool },
5304   sectocframes .bool_set:N = \c_notesslides_sectocframes_bool ,
5305   frameimages  .bool_set:N = \c_notesslides_frameimages_bool ,
5306   fiboxed      .bool_set:N = \c_notesslides_fiboxed_bool ,
5307   nopproblems  .bool_set:N = \c_notesslides_nopproblems_bool,
5308   unknown      .code:n      = {
5309     \PassOptionsToClass{\CurrentOption}{stex}
5310     \PassOptionsToClass{\CurrentOption}{tikzinput}
5311   }
5312 }
5313 \ProcessKeysOptions{ notesslides / pkg }
5314 \newif\ifnotes
5315 \bool_if:NTF \c_notesslides_notes_bool {
5316   \notesttrue
5317 }{
5318   \notesfalse
5319 }
5320

```

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

```

5321 \str_if_empty:NTF \c_notesslides_topsect_str {
5322   \str_set_eq:NN \__notesslides_topsect \c_notesslides_defaulttopsec_str
5323 }{
5324   \str_set_eq:NN \__notesslides_topsect \c_notesslides_topsect_str
5325 }
5326 </package>

```

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

```

5327 <*cls>
5328 \bool_if:NTF \c_notesslides_notes_bool {
5329   \LoadClass{document-structure}
5330 }{
5331   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
5332   \newcounter{Item}
5333   \newcounter{paragraph}
5334   \newcounter{subparagraph}
5335   \newcounter{Hfootnote}
5336   \RequirePackage{document-structure}
5337 }

```

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

```

5338 \RequirePackage{notesslides}
5339 </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.

```

5340 \*package>
5341 \bool_if:NT \c__notesslides_notes_bool {
5342   \RequirePackage{a4wide}
5343   \RequirePackage{marginnote}
5344   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
5345   \RequirePackage{mdframed}
5346   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
5347   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
5348 }
5349 \RequirePackage{stex-tikzinput}
5350 \RequirePackage{etoolbox}
5351 \RequirePackage{amssymb}
5352 \RequirePackage{amsmath}
5353 \RequirePackage{comment}
5354 \RequirePackage{textcomp}
5355 \RequirePackage{url}
5356 \RequirePackage{graphicx}
5357 \RequirePackage{pgf}

```

## 39.2 Notes and Slides

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

```

5358 \bool_if:NT \c__notesslides_notes_bool {
5359   \renewcommand\usetheme[2][\]{\usepackage[#1]{beamernotestheme#2}}
5360 }

```

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

```

5361 \newcounter{slide}
5362 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
5363 \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.

```

5364 \bool_if:NTF \c__notesslides_notes_bool {
5365   \renewenvironment{note}{\ignorespaces}{\ignorespaces}{}
5366 }{
5367   \excludecomment{note}
5368 }

```

---

<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.

```

5369 \bool_if:NT \c__notesslides_notes_bool {
5370   \newlength{\slideframewidth}
5371   \setlength{\slideframewidth}{1.5pt}

```

**frame** We first define the keys.

```

5372 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
5373   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
5374     \bool_set_true:N #1
5375   }{
5376     \bool_set_false:N #1
5377   }
5378 }
5379 \keys_define:nn{notesslides / frame}{
5380   label .str_set_x:N = \l__notesslides_frame_label_str,
5381   allowframebreaks .code:n = {
5382     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
5383   },
5384   allowdisplaybreaks .code:n = {
5385     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
5386   },
5387   fragile .code:n = {
5388     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
5389   },
5390   shrink .code:n = {
5391     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
5392   },
5393   squeeze .code:n = {
5394     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
5395   },
5396   t .code:n = {
5397     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
5398   },
5399 }
5400 \cs_new_protected:Nn \__notesslides_frame_args:n {
5401   \str_clear:N \l__notesslides_frame_label_str
5402   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
5403   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
5404   \bool_set_true:N \l__notesslides_frame_fragile_bool
5405   \bool_set_true:N \l__notesslides_frame_shrink_bool
5406   \bool_set_true:N \l__notesslides_frame_squeeze_bool
5407   \bool_set_true:N \l__notesslides_frame_t_bool
5408   \keys_set:nn { notesslides / frame }{ #1 }
5409 }

```

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

```

5410 \renewenvironment{frame}[1][]{
5411   \__notesslides_frame_args:n{#1}
5412   \sffamily
5413   \stepcounter{slide}
5414   \def\@currentlabel{\theslide}
5415   \str_if_empty:NF \l__notesslides_frame_label_str {
5416     \label{\l__notesslides_frame_label_str}

```

5417 }

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

```
5418 \def\itemize@level{outer}
5419 \def\itemize@outer{outer}
5420 \def\itemize@inner{inner}
5421 \renewcommand\newpage{\addtocounter{framenum}{1}}
5422 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
5423 \renewenvironment{itemize}{
5424   \ifx\itemize@level\itemize@outer
5425     \def\itemize@label{$\rhd$}
5426   \fi
5427   \ifx\itemize@level\itemize@inner
5428     \def\itemize@label{$\scriptstyle\rhd$}
5429   \fi
5430   \begin{list}
5431     {\itemize@label}
5432     {\setlength{\labelsep}{.3em}
5433      \setlength{\labelwidth}{.5em}
5434      \setlength{\leftmargin}{1.5em}
5435     }
5436   \edef\itemize@level{\itemize@inner}
5437 }{
5438   \end{list}
5439 }
```

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

```
5440 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth]
5441 }{
5442   \medskip\miko@slidelabel\end{mdframed}
5443 }
```

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

`\frametitle`

```
5444 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
5445 }
```

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

EdN:21

`\pause` 21

```
5446 \bool_if:NT \c__notesslides_notes_bool {
5447   \newcommand\pause{}
5448 }
```

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

`nparagraph`

```
5449 \bool_if:NTF \c__notesslides_notes_bool {
5450   \newenvironment{nparagraph}[1][\begin{sparagraph}[#1]}\end{sparagraph}}
5451 }{
5452   \excludecomment{nparagraph}
5453 }
```

---

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



```

nomgroup
5454 \bool_if:NTF \c__notesslides_notes_bool {
5455   \newenvironment{nomgroup}[2] [] {\begin{omgroup}[#1]{#2}}{\end{omgroup}}
5456 }{
5457   \excludecomment{nomgroup}
5458 }

ndefinition
5459 \bool_if:NTF \c__notesslides_notes_bool {
5460   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}{\end{sdefinition}}
5461 }{
5462   \excludecomment{ndefinition}
5463 }

nassertion
5464 \bool_if:NTF \c__notesslides_notes_bool {
5465   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}{\end{sassertion}}
5466 }{
5467   \excludecomment{nassertion}
5468 }

nsproof
5469 \bool_if:NTF \c__notesslides_notes_bool {
5470   \newenvironment{nproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
5471 }{
5472   \excludecomment{nproof}
5473 }

nexample
5474 \bool_if:NTF \c__notesslides_notes_bool {
5475   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}{\end{sexample}}
5476 }{
5477   \excludecomment{nexample}
5478 }

\inputref@*skip We customize the hooks for in \inputref.
5479 \def\inputref@preskip{\smallskip}
5480 \def\inputref@postskip{\medskip}

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

\inputref*
5481 \let\orig@inputref\inputref
5482 \def\inputref{\@ifstar\ninputref\orig@inputref}
5483 \newcommand\ninputref[2] [] {
5484   \bool_if:NT \c__notesslides_notes_bool {
5485     \orig@inputref[#1]{#2}
5486   }
5487 }

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

```

## 39.3 Header and Footer Lines

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

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

```
5488 \newlength{\slidelogoheight}
5489
5490 \bool_if:NTF \c__notesslides_notes_bool {
5491   \setlength{\slidelogoheight}{.4cm}
5492 }{
5493   \setlength{\slidelogoheight}{1cm}
5494 }
5495 \newsavebox{\slidelogo}
5496 \sbox{\slidelogo}{\TeX}
5497 \newrobustcmd{\setslidelogo}[1]{
5498   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
5499 }
```

(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.

```
5500 \def\source{Michael Kohlhase}% customize locally
5501 \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.

```
5502 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
5503 \newsavebox{\cclogo}
5504 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
5505 \newif\ifcchref\cchreffalse
5506 \AtBeginDocument{
5507   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
5508 }
5509 \def\licensing{
5510   \ifcchref
5511     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
5512   \else
5513     {\usebox{\cclogo}}
5514   \fi
5515 }
5516 \newrobustcmd{\setlicensing}[2][]{
5517   \def@url{#1}
5518   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
5519   \ifx@url@empty
5520     \def\licensing{{\usebox{\cclogo}}}
5521   \else
5522     \def\licensing{
```

```

5523     \ifcchref
5524     \href{#1}{\usebox{\cclogo}}
5525     \else
5526     {\usebox{\cclogo}}
5527     \fi
5528   }
5529 \fi
5530 }

```

(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>

```

5531 \newrobustcmd\miko@slidelabel{
5532   \vbox to \slidelogoheight{
5533     \vss\hbox to \slidewidth
5534     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
5535   }
5536 }

```

(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.

```

5537 \def\Gin@mhrepos{}
5538 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
5539 \define@key{Gin}{label}{\def\currentlabel{\arabic{slide}}\label{#1}}
5540 \newrobustcmd\frameimage[2][{}]{
5541   \stepcounter{slide}
5542   \bool_if:NT \c__notesslides_frameimages_bool {
5543     \def\Gin@ewidth{}\setkeys{Gin}{#1}
5544     \bool_if:NF \c__notesslides_notes_bool { \vfill }
5545     \begin{center}
5546       \bool_if:NTF \c__notesslides_fiboxed_bool {
5547         \fbox{
5548           \ifx\Gin@ewidth\@empty
5549             \ifx\Gin@mhrepos\@empty
5550               \mhgraphics[width=\slidewidth,#1]{#2}
5551             \else
5552               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5553             \fi
5554           \else% Gin@ewidth empty
5555             \ifx\Gin@mhrepos\@empty
5556               \mhgraphics[#1]{#2}
5557             \else
5558               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5559             \fi
5560           \fi% Gin@ewidth empty
5561         }
5562       }{
5563         \ifx\Gin@ewidth\@empty

```

---

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

```

5564         \ifx\Gin@mhrepos\empty
5565             \mhgraphics[width=\slidewidth,#1]{#2}
5566         \else
5567             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5568         \fi
5569         \ifx\Gin@mhrepos\empty
5570             \mhgraphics[#1]{#2}
5571         \else
5572             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5573         \fi
5574     \fi% Gin@ewidth empty
5575 }
5576 \end{center}
5577 \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
5578 \bool_if:NF \c__notesslides_notes_bool { \vfill }
5579 }
5580 } % 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.

```

5581 \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

```

5582 \AddToHook{begindocument}{
5583     \definecolor{green}{rgb}{0,.5,0}
5584     \definecolor{purple}{cmyk}{.3,1,0,.17}
5585 }

```

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`.

```

5586 % \def\STpresent#1{\textcolor{blue}{#1}}
5587 \def\defemph#1{\textcolor{magenta}{#1}}
5588 \def\symrefemph#1{\textcolor{cyan}{#1}}
5589 \def\compemph#1{\textcolor{blue}{#1}}
5590 \def\titleemph#1{\textcolor{blue}{#1}}
5591 \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.

```

5592 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
5593 \def\smalltextwarning{
5594     \pgfuseimage{miko@small@dbend}
5595     \xspace
5596 }
5597 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}

```

```

5598 \newrobustcmd\textwarning{
5599   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
5600   \xspace
5601 }
5602 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}
5603 \newrobustcmd\bigtextwarning{
5604   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
5605   \xspace
5606 }

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

5607 \newrobustcmd\putgraphicsat[3]{
5608   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
5609 }
5610 \newrobustcmd\putat[2]{
5611   \begin{picture}(0,0)\put(#1){#2}\end{picture}
5612 }

```

## 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`.

```

5613 \bool_if:NT \c__notesslides_sectocframes_bool {
5614   \str_if_eq:VnTF \__notesslidesstopsect{part}{
5615     \newcounter{chapter}\counterwithin*{section}{chapter}
5616   }{
5617     \str_if_eq:VnT\__notesslidesstopsect{chapter}{
5618       \newcounter{chapter}\counterwithin*{section}{chapter}
5619     }
5620   }
5621 }

```

`\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

5622 \def\part@prefix{}
5623 \@ifpackageloaded{document-structure}{}{
5624   \str_case:VnF \__notesslidesstopsect {
5625     {part}{
5626       \int_set:Nn \l_document_structure_section_level_int {0}
5627       \def\thesection{\arabic{chapter}.\arabic{section}}
5628       \def\part@prefix{\arabic{chapter}.}
5629     }
5630     {chapter}{
5631       \int_set:Nn \l_document_structure_section_level_int {1}
5632       \def\thesection{\arabic{chapter}.\arabic{section}}
5633       \def\part@prefix{\arabic{chapter}.}
5634     }
5635   }{
5636     \int_set:Nn \l_document_structure_section_level_int {2}
5637     \def\part@prefix{}

```

```

5638 }
5639 }
5640
5641 \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

```

5642 \renewenvironment{omgroup}[2][]{
5643   \__document_structure_omgroup_args:n { #1 }
5644   \int_incr:N \l_document_structure_omgroup_level_int
5645   \int_incr:N \l_document_structure_section_level_int
5646   \bool_if:NT \c__notesslides_sectocframes_bool {
5647     \stepcounter{slide}
5648     \begin{frame}[noframenumbering]
5649     \vfill\Large\centering
5650     \red{
5651       \ifcase\l_document_structure_section_level_int\or
5652         \stepcounter{part}
5653         \def\__notesslideslabel{\omdoc@part@kw~\Roman{part}}
5654         \def\currentsectionlevel{\omdoc@part@kw}
5655       \or
5656         \stepcounter{chapter}
5657         \def\__notesslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
5658         \def\currentsectionlevel{\omdoc@chapter@kw}
5659       \or
5660         \stepcounter{section}
5661         \def\__notesslideslabel{\part@prefix\arabic{section}}
5662         \def\currentsectionlevel{\omdoc@section@kw}
5663       \or
5664         \stepcounter{subsection}
5665         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
5666         \def\currentsectionlevel{\omdoc@subsection@kw}
5667       \or
5668         \stepcounter{subsubsection}
5669         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
5670         \def\currentsectionlevel{\omdoc@subsubsection@kw}
5671       \or
5672         \stepcounter{paragraph}
5673         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
5674         \def\currentsectionlevel{\omdoc@paragraph@kw}
5675       \else
5676         \def\__notesslideslabel{}
5677         \def\currentsectionlevel{\omdoc@paragraph@kw}
5678       \fi% end ifcase
5679       \__notesslideslabel%\sref@label@id\__notesslideslabel
5680       \quad #2%
5681     }%
5682     \vfill%
5683     \end{frame}%
5684   }
5685   \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str%

```

```

5686 }{}
5687 }

```

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

```

5688 \def\inserttheorembodyfont{\normalfont}
5689 %\bool_if:NF \c__notesslides_notes_bool {
5690 % \defbeamertemplate{theorem begin}{miko}
5691 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
5692 % \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
5693 % \inserttheorempunctuation\inserttheorembodyfont\hspace}
5694 % \defbeamertemplate{theorem end}{miko}{}}

```

and we set it as the default one.

```

5695 % \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.

```

5696 % \expandafter\def\csname Parent2\endcsname{}
5697 %}
5698
5699 \AddToHook{begindocument}{% this does not work for some reasons
5700 \setbeamertemplate{theorems}[ams style]
5701 }
5702 \bool_if:NT \c__notesslides_notes_bool {
5703 \renewenvironment{columns}[1][{}]{%
5704 \par\noindent%
5705 \begin{minipage}%
5706 \slidewidth\centering\leavevmode%
5707 }{%
5708 \end{minipage}\par\noindent%
5709 }%
5710 \newsavebox\columnbox%
5711 \renewenvironment<>{column}[2][{}]{%
5712 \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
5713 }{%
5714 \end{minipage}\end{lrbox}\usebox\columnbox%
5715 }%
5716 }
5717 \bool_if:NTF \c__notesslides_noproblems_bool {
5718 \newenvironment{problems}{}{}
5719 }{
5720 \excludecomment{problems}
5721 }

```

## 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.

```

5722 \gdef\printexcursions{}
5723 \newcommand\excursionref[2]{% label, text
5724 \bool_if:NT \c__notesslides_notes_bool {

```

```

5725     \begin{sparagraph}[title=Excursion]
5726     #2 \sref[fallback=the appendix]{#1}.
5727     \end{sparagraph}
5728   }
5729 }
5730 \newcommand\activate@excursion[2][] {
5731   \gappto\printexcursions{\inputref{#1}{#2}}
5732 }
5733 \newcommand\excursion[4][] {% repos, label, path, text
5734   \bool_if:NT \c__notesslides_notes_bool {
5735     \activate@excursion[1]{#3}\excursionref{#2}{#4}
5736   }
5737 }

```

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

\excursiongroup

```

5738 \keys_define:nn{notesslides / excursiongroup }{
5739   id      .str_set_x:N = \l__notesslides_excursion_id_str,
5740   intro   .tl_set:N    = \l__notesslides_excursion_intro_tl,
5741   mhrepos .str_set_x:N = \l__notesslides_excursion_mhrepos_str
5742 }
5743 \cs_new_protected:Nn \__notesslides_excursion_args:n {
5744   \tl_clear:N \l__notesslides_excursion_intro_tl
5745   \str_clear:N \l__notesslides_excursion_id_str
5746   \str_clear:N \l__notesslides_excursion_mhrepos_str
5747   \keys_set:nn {notesslides / excursiongroup }{ #1 }
5748 }
5749 \newcommand\excursiongroup[1][] {
5750   \__notesslides_excursion_args:n{ #1 }
5751   \ifdefempty\printexcursions{}% only if there are excursions
5752   {\begin{note}
5753     \begin{omgroup}[1]{Excursions}%
5754     \ifdefempty\l__notesslides_excursion_intro_tl{{
5755       \inputref[\l__notesslides_excursion_mhrepos_str]{
5756         \l__notesslides_excursion_intro_tl
5757       }
5758     }
5759     \printexcursions%
5760     \end{omgroup}
5761     \end{note}}
5762   }
5763   \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
5764   \end{package}

```

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



## Chapter 40

# The Implementation

### 40.1 Package Options

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

```
5765 <*package>
5766 <@@=problems>
5767 \ProvidesExplPackage{problem}{2019/03/20}{1.3}{Semantic Markup for Problems}
5768 \RequirePackage{l3keys2e,expl-keystr-compat}
5769
5770 \keys_define:nn { problem / pkg }{
5771   notes      .default:n    = { true },
5772   notes      .bool_set:N   = \c__problems_notes_bool,
5773   gnotes     .default:n    = { true },
5774   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
5775   hints      .default:n    = { true },
5776   hints      .bool_set:N   = \c__problems_hints_bool,
5777   solutions  .default:n    = { true },
5778   solutions  .bool_set:N   = \c__problems_solutions_bool,
5779   pts        .default:n    = { true },
5780   pts        .bool_set:N   = \c__problems_pts_bool,
5781   min        .default:n    = { true },
5782   min        .bool_set:N   = \c__problems_min_bool,
5783   boxed      .default:n    = { true },
5784   boxed      .bool_set:N   = \c__problems_boxed_bool,
5785   unknown    .code:n       = {}
5786 }
5787 \newif\ifsolutions
5788
5789 \ProcessKeysOptions{ problem / pkg }
5790 \bool_if:NTF \c__problems_solutions_bool {
5791   \solutionstrue
5792 }{
5793   \solutionsfalse
5794 }
```

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

```
5795 \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.

```
5796 \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.

```
5797 \def\prob@problem@kw{Problem}
5798 \def\prob@solution@kw{Solution}
5799 \def\prob@hint@kw{Hint}
5800 \def\prob@note@kw{Note}
5801 \def\prob@gnote@kw{Grading}
5802 \def\prob@pt@kw{pt}
5803 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
5804 \AddToHook{begindocument}{
5805   \ltx@ifpackageloaded{babel}{
5806     \makeatletter
5807     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5808     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5809       \input{problem-ngerman.ldf}
5810     }
5811     \clist_if_in:NnT \l_tmpa_clist {finnish}{
5812       \input{problem-finnish.ldf}
5813     }
5814     \clist_if_in:NnT \l_tmpa_clist {french}{
5815       \input{problem-french.ldf}
5816     }
5817     \clist_if_in:NnT \l_tmpa_clist {russian}{
5818       \input{problem-russian.ldf}
5819     }
5820     \makeatother
5821   }{ }
5822 }
```

## 40.2 Problems and Solutions

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

```
5823 \keys_define:nn{ problem / problem }{
5824   id      .str_set:x:N = \l__problems_prob_id_str,
5825   pts     .tl_set:N    = \l__problems_prob_pts_tl,
5826   min     .tl_set:N    = \l__problems_prob_min_tl,
5827   title   .tl_set:N    = \l__problems_prob_title_tl,
5828   type    .tl_set:N    = \l__problems_prob_type_tl,
5829   refnum  .int_set:N    = \l__problems_prob_refnum_int
5830 }
5831 \cs_new_protected:Nn \__problems_prob_args:n {
```

```

5832 \str_clear:N \l__problems_prob_id_str
5833 \tl_clear:N \l__problems_prob_pts_tl
5834 \tl_clear:N \l__problems_prob_min_tl
5835 \tl_clear:N \l__problems_prob_title_tl
5836 \tl_clear:N \l__problems_prob_type_tl
5837 \int_zero_new:N \l__problems_prob_refnum_int
5838 \keys_set:nn { problem / problem }{ #1 }
5839 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
5840   \let\l__problems_prob_refnum_int\undefined
5841 }
5842 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

5843 \newcounter{problem}
5844 \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.

```

5845 \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

```

5846 \newcommand\prob@number{
5847   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
5848     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
5849   }{
5850     \int_if_exist:NTF \l__problems_prob_refnum_int {
5851       \prob@label{\int_use:N \l__problems_prob_refnum_int }
5852     }{
5853       \prob@label\theproblem
5854     }
5855   }
5856 }

```

(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.

```

5857 \newcommand\prob@title[3]{%
5858   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
5859     #2 \l__problems_inclprob_title_tl #3
5860   }{
5861     \tl_if_exist:NTF \l__problems_prob_title_tl {
5862       #2 \l__problems_prob_title_tl #3
5863     }{
5864       #1
5865     }
5866   }
5867 }

```

(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.

```

5868 \def\prob@heading{
5869   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
5870   %\sref@label{id{\prob@problem@kw~\prob@number}}{~}
5871 }

```

(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`

```

5872 \newenvironment{sproblem}[1][{}]{
5873   \__problems_prob_args:n{#1}%\sref@target%
5874   \@in@omtexttrue% we are in a statement (for inline definitions)
5875   \stepcounter{problem}\record@problem
5876   \def\current@section@level{\prob@problem@kw}
5877   \tl_if_exist:NTF \l__problems_inclprob_type_tl {
5878     \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
5879   }{
5880     \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
5881   }
5882   \str_if_exist:NTF \l__problems_inclprob_id_str {
5883     \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
5884   }{
5885     \str_set_eq:NN \sproblemid \l__problems_prob_id_str
5886   }
5887
5888
5889   \clist_set:No \l_tmpa_clist \sproblemtype
5890   \tl_clear:N \l_tmpa_tl
5891   \clist_map_inline:Nn \l_tmpa_clist {
5892     \tl_if_exist:cT {\__problems_sproblem_##1_start:}{
5893       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_start:}}
5894     }
5895   }
5896   \tl_if_empty:NTF \l_tmpa_tl {
5897     \__problems_sproblem_start:
5898   }{
5899     \l_tmpa_tl
5900   }
5901   \stex_ref_new_doc_target:n \sproblemid
5902 }{
5903   \clist_set:No \l_tmpa_clist \sproblemtype
5904   \tl_clear:N \l_tmpa_tl
5905   \clist_map_inline:Nn \l_tmpa_clist {
5906     \tl_if_exist:cT {\__problems_sproblem_##1_end:}{
5907       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_end:}}
5908     }

```

```

5909 }
5910 \tl_if_empty:NTF \l_tmpa_tl {
5911   \__problems_sproblem_end:
5912 }{
5913   \l_tmpa_tl
5914 }
5915
5916
5917 \smallskip
5918 }
5919
5920
5921 \cs_new_protected:Nn \__problems_sproblem_start: {
5922   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
5923 }
5924 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
5925
5926 \newcommand\stexpatchproblem[3][] {
5927   \str_set:Nx \l_tmpa_str{ #1 }
5928   \str_if_empty:NTF \l_tmpa_str {
5929     \tl_set:Nn \__problems_sproblem_start: { #2 }
5930     \tl_set:Nn \__problems_sproblem_end: { #3 }
5931   }{
5932     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
5933     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
5934   }
5935 }
5936
5937
5938 \bool_if:NT \c__problems_boxed_bool {
5939   \surroundwithmdframed{problem}
5940 }

```

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

```

5941 \def\record@problem{
5942   \protected@write\@auxout{}
5943   {
5944     \string\@problem{\prob@number}
5945     {
5946       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
5947         \l__problems_inclprob_pts_tl
5948       }{
5949         \l__problems_prob_pts_tl
5950       }
5951     }%
5952     {
5953       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
5954         \l__problems_inclprob_min_tl
5955       }{
5956         \l__problems_prob_min_tl
5957       }
5958     }
5959   }
5960 }

```

(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).

```
5961 \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.

```
5962 \keys_define:nn { problem / solution }{
5963   id                .str_set_x:N = \l__problems_solution_id_str ,
5964   for               .tl_set:N   = \l__problems_solution_for_tl ,
5965   height            .dim_set:N   = \l__problems_solution_height_dim ,
5966   creators          .clist_set:N = \l__problems_solution_creators_clist ,
5967   contributors      .clist_set:N = \l__problems_solution_contributors_clist ,
5968   srccite           .tl_set:N   = \l__problems_solution_srccite_tl
5969 }
5970 \cs_new_protected:Nn \__problems_solution_args:n {
5971   \str_clear:N \l__problems_solution_id_str
5972   \tl_clear:N \l__problems_solution_for_tl
5973   \tl_clear:N \l__problems_solution_srccite_tl
5974   \clist_clear:N \l__problems_solution_creators_clist
5975   \clist_clear:N \l__problems_solution_contributors_clist
5976   \dim_zero:N \l__problems_solution_height_dim
5977   \keys_set:nn { problem / solution }{ #1 }
5978 }
```

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

```
5979 \newcommand\@startsolution[1][{}]{
5980   \__problems_solution_args:n { #1 }
5981   \@in@omtexttrue% we are in a statement.
5982   \bool_if:NF \c__problems_boxed_bool { \hrule }
5983   \smallskip\noindent
5984   {\textbf\prob@solution@kw : \enspace}
5985   \begin{small}
5986   \def\current@section@level{\prob@solution@kw}
5987   \ignorespacesandpars
5988 }
```

**\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.

```
5989 \newcommand\startsolutions{
5990   \specialcomment{solution}{\@startsolution}{
5991     \bool_if:NF \c__problems_boxed_bool {
5992       \hrule\medskip
5993     }
5994     \end{small}%
5995   }
5996   \bool_if:NT \c__problems_boxed_bool {
5997     \surroundwithmdframed{solution}
5998   }
5999 }
```

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

\stopsolutions

```
6000 \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.

```
6001 \ifsolutions
6002   \startsolutions
6003 \else
6004   \stopsolutions
6005 \fi
```

exnote

```
6006 \bool_if:NTF \c__problems_notes_bool {
6007   \newenvironment{exnote}[1][]{
6008     \par\smallskip\hrule\smallskip
6009     \noindent\textbf{\prob@note@kw : }\small
6010   }{
6011     \smallskip\hrule
6012   }
6013 }{
6014   \excludecomment{exnote}
6015 }
```

hint

```
6016 \bool_if:NTF \c__problems_notes_bool {
6017   \newenvironment{hint}[1][]{
6018     \par\smallskip\hrule\smallskip
6019     \noindent\textbf{\prob@hint@kw :~ }\small
6020   }{
6021     \smallskip\hrule
6022   }
6023 \newenvironment{exhint}[1][]{
6024   \par\smallskip\hrule\smallskip
6025   \noindent\textbf{\prob@hint@kw :~ }\small
6026 }{
6027   \smallskip\hrule
6028 }
6029 }{
6030   \excludecomment{hint}
6031   \excludecomment{exhint}
6032 }
```

gnote

```
6033 \bool_if:NTF \c__problems_notes_bool {
6034   \newenvironment{gnote}[1][]{
6035     \par\smallskip\hrule\smallskip
6036     \noindent\textbf{\prob@gnote@kw : }\small
6037   }{
6038     \smallskip\hrule
6039   }
6040 }{
6041   \excludecomment{gnote}
6042 }
```

## 40.3 Multiple Choice Blocks

```

6043 \newenvironment{mcb}{
6044   \begin{enumerate}
6045 }{
6046   \end{enumerate}
6047 }

```

we define the keys for the mcc macro

```

6048 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
6049   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
6050     \bool_set_true:N #1
6051   }{
6052     \bool_set_false:N #1
6053   }
6054 }
6055 \keys_define:nn { problem / mcc }{
6056   id          .str_set_x:N = \l__problems_mcc_id_str ,
6057   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
6058   T           .default:n   = { true } ,
6059   T           .bool_set:N  = \l__problems_mcc_t_bool ,
6060   F           .default:n   = { true } ,
6061   F           .bool_set:N  = \l__problems_mcc_f_bool ,
6062   Ttext       .code:n      = {
6063     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
6064   } ,
6065   Ftext       .code:n      = {
6066     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
6067   }
6068 }
6069 \cs_new_protected:Nn \l__problems_mcc_args:n {
6070   \str_clear:N \l__problems_mcc_id_str
6071   \tl_clear:N \l__problems_mcc_feedback_tl
6072   \bool_set_true:N \l__problems_mcc_t_bool
6073   \bool_set_true:N \l__problems_mcc_f_bool
6074   \bool_set_true:N \l__problems_mcc_Ttext_bool
6075   \bool_set_false:N \l__problems_mcc_Ftext_bool
6076   \keys_set:nn { problem / mcc }{ #1 }
6077 }

```

\mcc

```

6078 \newcommand\mcc[2][] {
6079   \l__problems_mcc_args:n{ #1 }
6080   \item #2
6081   \ifsolutions
6082     \\\
6083     \bool_if:NT \l__problems_mcc_t_bool {
6084       % TODO!
6085       % \ifcsstring{mcc@T}{T}{\mcc@Ttext}%
6086     }
6087     \bool_if:NT \l__problems_mcc_f_bool {

```

---

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



```

6088      % TODO!
6089      % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
6090    }
6091    \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
6092      !
6093    }{
6094      \l__problems_mcc_feedback_tl
6095    }
6096    \fi
6097  } %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.

```

6098
6099 \keys_define:nn{ problem / inclproblem }{
6100   id      .str_set:N = \l__problems_inclprob_id_str,
6101   pts     .tl_set:N  = \l__problems_inclprob_pts_tl,
6102   min     .tl_set:N  = \l__problems_inclprob_min_tl,
6103   title   .tl_set:N  = \l__problems_inclprob_title_tl,
6104   refnum  .int_set:N  = \l__problems_inclprob_refnum_int,
6105   type    .tl_set:N  = \l__problems_inclprob_type_tl,
6106   mhrepos .str_set:N = \l__problems_inclprob_mhrepos_str
6107 }
6108 \cs_new_protected:Nn \l__problems_inclprob_args:n {
6109   \str_clear:N \l__problems_prob_id_str
6110   \tl_clear:N \l__problems_inclprob_pts_tl
6111   \tl_clear:N \l__problems_inclprob_min_tl
6112   \tl_clear:N \l__problems_inclprob_title_tl
6113   \tl_clear:N \l__problems_inclprob_type_tl
6114   \int_zero_new:N \l__problems_inclprob_refnum_int
6115   \str_clear:N \l__problems_inclprob_mhrepos_str
6116   \keys_set:nn { problem / inclproblem }{ #1 }
6117   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
6118     \let\l__problems_inclprob_pts_tl\undefined
6119   }
6120   \tl_if_empty:NT \l__problems_inclprob_min_tl {
6121     \let\l__problems_inclprob_min_tl\undefined
6122   }
6123   \tl_if_empty:NT \l__problems_inclprob_title_tl {
6124     \let\l__problems_inclprob_title_tl\undefined
6125   }
6126   \tl_if_empty:NT \l__problems_inclprob_type_tl {
6127     \let\l__problems_inclprob_type_tl\undefined
6128   }
6129   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
6130     \let\l__problems_inclprob_refnum_int\undefined
6131   }
6132 }

```

```

6133
6134 \cs_new_protected:Nn \__problems_inclprob_clear: {
6135   \let\l__problems_inclprob_id_str\undefined
6136   \let\l__problems_inclprob_pts_tl\undefined
6137   \let\l__problems_inclprob_min_tl\undefined
6138   \let\l__problems_inclprob_title_tl\undefined
6139   \let\l__problems_inclprob_type_tl\undefined
6140   \let\l__problems_inclprob_refnum_int\undefined
6141   \let\l__problems_inclprob_mhrepos_str\undefined
6142 }
6143 \__problems_inclprob_clear:
6144
6145 \newcommand\includeproblem[2][ ]{
6146   \__problems_inclprob_args:n{ #1 }
6147   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
6148     \input{#2}
6149   }{
6150     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
6151       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
6152     }
6153   }
6154   \__problems_inclprob_clear:
6155 }

```

(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.

```

6156 \AddToHook{enddocument}{
6157   \bool_if:NT \c__problems_pts_bool {
6158     \message{Total:~\arabic{pts}~points}
6159   }
6160   \bool_if:NT \c__problems_min_bool {
6161     \message{Total:~\arabic{min}~minutes}
6162   }
6163 }

```

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

```

6164 \def\pts#1{
6165   \bool_if:NT \c__problems_pts_bool {
6166     \marginpar{#1~\prob@pt@kw}
6167   }
6168 }
6169 \def\min#1{
6170   \bool_if:NT \c__problems_min_bool {
6171     \marginpar{#1~\prob@min@kw}
6172   }
6173 }

```

`\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.

```

6174 \newcounter{pts}
6175 \def\show@pts{
6176   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
6177     \bool_if:NT \c__problems_pts_bool {
6178       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
6179       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
6180     }
6181   }{
6182     \tl_if_exist:NT \l__problems_prob_pts_tl {
6183       \bool_if:NT \c__problems_pts_bool {
6184         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
6185         \addtocounter{pts}{\l__problems_prob_pts_tl}
6186       }
6187     }
6188   }
6189 }

```

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

`\show@min`

```

6190 \newcounter{min}
6191 \def\show@min{
6192   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
6193     \bool_if:NT \c__problems_min_bool {
6194       \marginpar{\l__problems_inclprob_min_tl\ min}
6195       \addtocounter{min}{\l__problems_inclprob_min_tl}
6196     }
6197   }{
6198     \tl_if_exist:NT \l__problems_prob_min_tl {
6199       \bool_if:NT \c__problems_min_bool {
6200         \marginpar{\l__problems_prob_min_tl\ min}
6201         \addtocounter{min}{\l__problems_prob_min_tl}
6202       }
6203     }
6204   }
6205 }
6206 \</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.

```
6207 <@@=hwexam>
6208 <*cls>
6209 \ProvidesExplClass{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
6210 \RequirePackage{l3keys2e,expl-keystr-compatible}
6211 \DeclareOption*{
6212   \PassOptionsToClass{\CurrentOption}{document-structure}
6213   \PassOptionsToPackage{\CurrentOption}{stex}
6214   \PassOptionsToPackage{\CurrentOption}{hwexam}
6215   \PassOptionsToPackage{\CurrentOption}{tikzinput}
6216 }
6217 \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.

```
6218 \LoadClass{document-structure}
6219 \RequirePackage{stex}
6220 \RequirePackage{hwexam}
6221 \RequirePackage{tikzinput}
6222 \RequirePackage{graphicx}
6223 \RequirePackage{a4wide}
6224 \RequirePackage{amssymb}
6225 \RequirePackage{amstext}
6226 \RequirePackage{amsmath}
```

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

```

6227 \newcommand\assig@default@type{\hwexam@assignment@kw}
6228 \def\document@hwexamtype{\assig@default@type}
6229 <@@=document_structure>
6230 \keys_define:nn { document-structure / document }{
6231 id .str_set_x:N = \c_document_structure_document_id_str,
6232 hwexamtype .tl_set:N = \document@hwexamtype
6233 }
6234 <@@=hwexam>
6235 </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.

```
6236 \*package>
6237 \ProvidesExplPackage{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
6238 \RequirePackage{l3keys2e,expl-keystr-compat}
6239
6240 \newif\iftest\testfalse
6241 \DeclareOption{test}{\testtrue}
6242 \newif\ifmultiple\multiplefalse
6243 \DeclareOption{multiple}{\multipletrue}
6244 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
6245 \ProcessOptions
```

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

```
6246 \RequirePackage{keyval}[1997/11/10]
6247 \RequirePackage{problem}
```

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

```
6248 \newcommand\hwexam@assignment@kw{Assignment}
6249 \newcommand\hwexam@given@kw{Given}
6250 \newcommand\hwexam@due@kw{Due}
6251 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
6252 blank~for~extra~space}
6253 \def\hwexam@minutes@kw{minutes}
6254 \newcommand\correction@probs@kw{prob.}
6255 \newcommand\correction@pts@kw{total}
6256 \newcommand\correction@reached@kw{reached}
6257 \newcommand\correction@sum@kw{Sum}
6258 \newcommand\correction@grade@kw{grade}
6259 \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

```

6260 \AddToHook{begindocument}{
6261 \ltx@ifpackageloaded{babel}{
6262 \makeatletter
6263 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6264 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6265 \input{hwexam-ngerman.ldf}
6266 }
6267 \clist_if_in:NnT \l_tmpa_clist {finnish}{
6268 \input{hwexam-finnish.ldf}
6269 }
6270 \clist_if_in:NnT \l_tmpa_clist {french}{
6271 \input{hwexam-french.ldf}
6272 }
6273 \clist_if_in:NnT \l_tmpa_clist {russian}{
6274 \input{hwexam-russian.ldf}
6275 }
6276 \makeatother
6277 }{}
6278 }
6279

```

## 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.

```

6280 \newcounter{assignment}
6281 \numberproblemsin{assignment}
6282 \renewcommand\prob@label[1]{\assignment@number.#1}

```

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

```

6283 \keys_define:nn { hwexam / assignment } {
6284 id .str_set:N = \l__hwexam_assign_id_str,
6285 number .int_set:N = \l__hwexam_assign_number_int,
6286 title .tl_set:N = \l__hwexam_assign_title_tl,
6287 type .tl_set:N = \l__hwexam_assign_type_tl,
6288 given .tl_set:N = \l__hwexam_assign_given_tl,
6289 due .tl_set:N = \l__hwexam_assign_due_tl,
6290 loadmodules .code:n = {
6291 \bool_set_true:N \l__hwexam_assign_loadmodules_bool
6292 }
6293 }
6294 \cs_new_protected:Nn \__hwexam_assignment_args:n {
6295 \str_clear:N \l__hwexam_assign_id_str
6296 \int_set:Nn \l__hwexam_assign_number_int {-1}
6297 \tl_clear:N \l__hwexam_assign_title_tl
6298 \tl_clear:N \l__hwexam_assign_type_tl
6299 \tl_clear:N \l__hwexam_assign_given_tl
6300 \tl_clear:N \l__hwexam_assign_due_tl
6301 \bool_set_false:N \l__hwexam_assign_loadmodules_bool

```

```

6302 \keys_set:nn { hwexam / assignment }{ #1 }
6303 }

```

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.

```

6304 \newcommand\given@due[2]{
6305 \bool_lazy_all:nF {
6306 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
6307 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
6308 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
6309 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
6310 }{ #1 }
6311
6312 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
6313 \tl_if_empty:NF \l__hwexam_assign_given_tl {
6314 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
6315 }
6316 }{
6317 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
6318 }
6319
6320 \bool_lazy_or:nnF {
6321 \bool_lazy_and_p:nn {
6322 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
6323 }{
6324 \tl_if_empty_p:V \l__hwexam_assign_due_tl
6325 }
6326 }{
6327 \bool_lazy_and_p:nn {
6328 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
6329 }{
6330 \tl_if_empty_p:V \l__hwexam_assign_due_tl
6331 }
6332 }{ ,~ }
6333
6334 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
6335 \tl_if_empty:NF \l__hwexam_assign_due_tl {
6336 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
6337 }
6338 }{
6339 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
6340 }
6341
6342 \bool_lazy_all:nF {
6343 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
6344 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
6345 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
6346 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
6347 }{ #2 }
6348 }

```

`\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.

```

6349 \newcommand\assignment@title[3]{
6350 \tl_if_empty:NTF \l__hwexam_inclasssign_title_tl {
6351 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
6352 #1
6353 }{
6354 #2\l__hwexam_assign_title_tl#3
6355 }
6356 }{
6357 #2\l__hwexam_inclasssign_title_tl#3
6358 }
6359 }

```

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

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

```

6360 \newcommand\assignment@number{
6361 \int_compare:nNnTF \l__hwexam_inclasssign_number_int = {-1} {
6362 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
6363 \arabic{assignment}
6364 } {
6365 \int_use:N \l__hwexam_assign_number_int
6366 }
6367 }{
6368 \int_use:N \l__hwexam_inclasssign_number_int
6369 }
6370 }

```

(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.

```

6371 \newenvironment{assignment}[1][]{
6372 \__hwexam_assignment_args:n { #1 }
6373 %\sref@target
6374 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
6375 \global\stepcounter{assignment}
6376 }{
6377 \global\setcounter{assignment}{\int_use:N\l__hwexam_assign_number_int}
6378 }
6379 \setcounter{problem}{0}
6380 \def\current@section@level{\document@hwexamtype}
6381 %\sref@label@id{\document@hwexamtype \thesection}
6382 \begin{@assignment}
6383 }{
6384 \end{@assignment}
6385 }

```

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

```

6386 \def\ass@title{
6387 \protect\document@hwexamtype~\arabic{assignment}
6388 \assignment@title{}\{;\}\{;\} -- \given@due{}\{;\}
6389 }
6390 \ifmultiple
6391 \newenvironment{@assignment}{
6392 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
6393 \begin{omgroup}[loadmodules]{\ass@title}
6394 }{
6395 \begin{omgroup}{\ass@title}
6396 }
6397 }{
6398 \end{omgroup}
6399 }

```

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

```

6400 \else
6401 \newenvironment{@assignment}{
6402 \begin{center}\bf
6403 \Large@title\strut\
6404 \document@hwexamtype~\arabic{assignment}\assignment@title{}\{;\}\{;\}\{;\}
6405 \large\given@due{--;\}\{;\}--}
6406 \end{center}
6407 }{}
6408 \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.

```

6409 \keys_define:nn { hwexam / inclassignment } {
6410 %id .str_set_x:N = \l__hwexam_assign_id_str,
6411 number .int_set:N = \l__hwexam_inclassign_number_int,
6412 title .tl_set:N = \l__hwexam_inclassign_title_tl,
6413 type .tl_set:N = \l__hwexam_inclassign_type_tl,
6414 given .tl_set:N = \l__hwexam_inclassign_given_tl,
6415 due .tl_set:N = \l__hwexam_inclassign_due_tl,
6416 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
6417 }
6418 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
6419 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
6420 \tl_clear:N \l__hwexam_inclassign_title_tl
6421 \tl_clear:N \l__hwexam_inclassign_type_tl
6422 \tl_clear:N \l__hwexam_inclassign_given_tl
6423 \tl_clear:N \l__hwexam_inclassign_due_tl
6424 \str_clear:N \l__hwexam_inclassign_mhrepos_str
6425 \keys_set:nn { hwexam / inclassignment }{ #1 }
6426 }
6427 \__hwexam_inclassignment_args:n {}
6428
6429 \newcommand\inputassignment[2][{}]{

```

```

6430 \_hwexam_inclassnment_args:n { #1 }
6431 \str_if_empty:NTF \l__hwexam_inclassn_mhrepos_str {
6432 \input{#2}
6433 }{
6434 \stex_in_repository:nn{\l__hwexam_inclassn_mhrepos_str}{
6435 \input{\mhp{path}\l__hwexam_inclassn_mhrepos_str}{#2}}
6436 }
6437 }
6438 \_hwexam_inclassnment_args:n {}
6439 }
6440 \newcommand\includeassignment[2][]{
6441 \newpage
6442 \inputassignment[#1]{#2}
6443 }

```

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

## 42.4 Typesetting Exams

\quizheading

```

6444 \ExplSyntaxOff
6445 \newcommand\quizheading[1]{%
6446 \def\@tas{#1}%
6447 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
6448 \ifx\@tas\@empty\else%
6449 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
6450 \fi%
6451 }
6452 \ExplSyntaxOn

```

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

\testheading

```

6453
6454 \def\hwexamheader{\input{hwexam-default.header}}
6455
6456 \def\hwexamminutes{
6457 \tl_if_empty:NTF \testheading@duration {
6458 {\testheading@min}~\hwexam@minutes@kw
6459 }{
6460 \testheading@duration
6461 }
6462 }
6463
6464 \keys_define:nn { hwexam / testheading } {
6465 min .tl_set:N = \testheading@min,
6466 duration .tl_set:N = \testheading@duration,
6467 reqpts .tl_set:N = \testheading@reqpts,
6468 tools .tl_set:N = \testheading@tools
6469 }
6470 \cs_new_protected:Nn \_hwexam_testheading_args:n {
6471 \tl_clear:N \testheading@min
6472 \tl_clear:N \testheading@duration

```

```

6473 \tl_clear:N \testheading@reqpts
6474 \tl_clear:N \testheading@tools
6475 \keys_set:nn { hwexam / testheading }{ #1 }
6476 }
6477 \newenvironment{testheading}[1][]{
6478   \_hwexam_testheading_args:n{ #1 }
6479   \newcount\check@time\check@time=\testheading@min
6480   \advance\check@time by -\theassignment@totalmin
6481   \newif\if@bonuspoints
6482   \tl_if_empty:NTF \testheading@reqpts {
6483     \@bonuspointsfalse
6484   }{
6485     \newcount\bonus@pts
6486     \bonus@pts=\theassignment@totalpts
6487     \advance\bonus@pts by -\testheading@reqpts
6488     \edef\bonus@pts{\the\bonus@pts}
6489     \@bonuspointstrue
6490   }
6491   \edef\check@time{\the\check@time}
6492
6493   \makeatletter\hwexamheader\makeatother
6494 }{
6495   \newpage
6496 }

```

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

**\testspace**

```

6497 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

**\testnewpage**

```

6498 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

**\testemptypage**

```

6499 \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.

```

6500 <@=problems>
6501 \renewcommand\@problem[3]{
6502   \stepcounter{assignment@probs}
6503   \def\__problemspts{#2}
6504   \ifx\__problemspts\@empty\else
6505     \addtocounter{assignment@totalpts}{#2}
6506   \fi
6507   \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
6508   \xdef\correction@probs{\correction@probs & #1}%
6509   \xdef\correction@pts{\correction@pts & #2}
6510   \xdef\correction@reached{\correction@reached &}

```

```

6511 }
6512 <@@=hwexam>

```

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

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

```

6513 \newcounter{assignment@probs}
6514 \newcounter{assignment@totalpts}
6515 \newcounter{assignment@totalmin}
6516 \def\correction@probs{\correction@probs@kw}
6517 \def\correction@pts{\correction@pts@kw}
6518 \def\correction@reached{\correction@reached@kw}
6519 \stepcounter{assignment@probs}
6520 \newcommand\correction@table{
6521 \resizebox{\textwidth}{!}{%
6522 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
6523 &\multicolumn{\theassignment@probs}{c|}||%|
6524 {\footnotesize\correction@forgrading@kw} &\\ \hline
6525 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
6526 \correction@pts & \theassignment@totalpts & \\ \hline
6527 \correction@reached & & \[.7cm]\hline
6528 \end{tabular}}
6529 </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}

```