

# The sTeX3 Package \*

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

2022-02-14

## 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 2022-02-14)

# Contents

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

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

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

<b>25</b>	<b>STeX-Basics Implementation</b>	<b>72</b>
25.1	The STeXDocument Class . . . . .	72
25.2	Preliminaries . . . . .	72
25.3	Messages and logging . . . . .	73
25.4	Persistence . . . . .	74
25.5	HTML Annotations . . . . .	74
25.6	Languages . . . . .	77
25.7	Activating/Deactivating Macros . . . . .	78
<b>26</b>	<b>STeX-MathHub Implementation</b>	<b>80</b>
26.1	Generic Path Handling . . . . .	80
26.2	PWD and kpsewhich . . . . .	82
26.3	File Hooks and Tracking . . . . .	83
26.4	MathHub Repositories . . . . .	84
<b>27</b>	<b>STeX-References Implementation</b>	<b>92</b>
27.1	Document URIs and URLs . . . . .	92
27.2	Setting Reference Targets . . . . .	94
27.3	Using References . . . . .	95
<b>28</b>	<b>STeX-Modules Implementation</b>	<b>98</b>
28.1	The module environment . . . . .	101
28.2	Invoking modules . . . . .	107
<b>29</b>	<b>STeX-Module Inheritance Implementation</b>	<b>109</b>
29.1	SMS Mode . . . . .	109
29.2	Inheritance . . . . .	112
<b>30</b>	<b>STeX-Symbols Implementation</b>	<b>117</b>
30.1	Symbol Declarations . . . . .	117
30.2	Notations . . . . .	124
<b>31</b>	<b>STeX-Terms Implementation</b>	<b>133</b>
31.1	Symbol Invocations . . . . .	133
31.2	Terms . . . . .	136
31.3	Notation Components . . . . .	142
<b>32</b>	<b>STeX-Structural Features Implementation</b>	<b>145</b>
32.1	Imports with modification . . . . .	145
32.2	The feature environment . . . . .	152
32.3	Features . . . . .	153
<b>33</b>	<b>STeX-Statements Implementation</b>	<b>159</b>
33.1	Definitions . . . . .	159
33.2	Assertions . . . . .	164
33.3	Examples . . . . .	167
33.4	Logical Paragraphs . . . . .	169

<b>34 The Implementation</b>	<b>174</b>
34.1 Package Options . . . . .	174
34.2 Proofs . . . . .	174
34.3 Justifications . . . . .	180
<b>35 <math>\text{\LaTeX}</math>-Others Implementation</b>	<b>182</b>
<b>36 <math>\text{\LaTeX}</math>-Metatheory Implementation</b>	<b>183</b>
<b>37 Tikzinput Implementation</b>	<b>186</b>
<b>38 document-structure.sty Implementation</b>	<b>188</b>
38.1 The document-structure Class . . . . .	188
38.2 Class Options . . . . .	188
38.3 Beefing up the <code>document</code> environment . . . . .	189
38.4 Implementation: document-structure Package . . . . .	189
38.5 Package Options . . . . .	189
38.6 Document Structure . . . . .	191
38.7 Front and Backmatter . . . . .	194
38.8 Global Variables . . . . .	196
<b>39 NotesSlides – Implementation</b>	<b>197</b>
39.1 Class and Package Options . . . . .	197
39.2 Notes and Slides . . . . .	199
39.3 Header and Footer Lines . . . . .	203
39.4 Frame Images . . . . .	204
39.5 Colors and Highlighting . . . . .	205
39.6 Sectioning . . . . .	206
39.7 Excursions . . . . .	208
<b>40 The Implementation</b>	<b>210</b>
40.1 Package Options . . . . .	210
40.2 Problems and Solutions . . . . .	211
40.3 Multiple Choice Blocks . . . . .	217
40.4 Including Problems . . . . .	218
40.5 Reporting Metadata . . . . .	219
<b>41 Implementation: The hwexam Class</b>	<b>221</b>
41.1 Class Options . . . . .	221
<b>42 Implementation: The hwexam Package</b>	<b>223</b>
42.1 Package Options . . . . .	223
42.2 Assignments . . . . .	224
42.3 Including Assignments . . . . .	227
42.4 Typesetting Exams . . . . .	228
42.5 Leftovers . . . . .	230

**Part I**  
**Manual**

# Chapter 1

## What is sTeX?

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

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

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

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

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

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



# Chapter 2

## Quickstart

### 2.1 Setup

#### 2.1.1 The sTeX IDE

TODO: VSCode Plugin

#### 2.1.2 Manual Setup

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

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

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

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

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

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

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

---

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

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

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

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

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

The document we will consider is the following:

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

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

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

Compiling this document with `pdflatex` should yield the output

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

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

### \usemodule

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

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

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

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

---

`\symref`  
`\symname`

---

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

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

---

`\importmodule`

---

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

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

## Chapter 3

# Using Semantic Macros

TODO

## Chapter 4

# TeX Archives

### 4.1 The Local MathHub-Directory

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

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

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

### 4.2 The Structure of TeX Archives

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

Each such archive needs two subdirectories:

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

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

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

### 4.3 MANIFEST.MF-Files

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

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

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

- `id`: The name of the archive, including its group (e.g. `smglom/calculus`),
- `source-base` or  
`ns`: The namespace from which all symbol and module URIs in this repository are formed, see (TODO),
- `narration-base`: The namespace from which all document URIs in this repository are formed, see (TODO),
- `url`: The URL that is formed as a basis for *external references*, see (TODO),
- `dependencies`: All archives that this archive depends on.  $\text{\TeX}$  ignores this field, but MMT can pick up on them to resolve dependencies, e.g. for `lmh install`.

## Chapter 5

# Creating New Modules and Symbols

TODO

### 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`  
`\stex`

---

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

### 8.1.1 Semantic Macros and Notations

Semantic macros invoke a formally declared symbol.

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

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

Multiplying again by  $b$  yields...

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

### Example 8

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

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

<sup>4</sup>EdNOTE: TODO

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

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

### Example 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://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://stextest?STEXModuleTest1  
file://stextest?STEXModuleTest2  
file://stextest?STEXModuleTest3  
foo1  
foo2  
foo3

---

`\stex_activate_module:n`

---

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

## Chapter 13

# STEX-Module Inheritance

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

### 13.1 Macros and Environments

#### 13.1.1 SMS Mode

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

---

`\g_stex_smsmode_allowedmacros_tl`

---

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

---

`\g_stex_smsmode_allowedmacros_escape_tl`

---

Macros that are executed with the category codes restored.

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

---

`\g_stex_smsmode_allowedenvs_seq`

---

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

---

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

---

Tests whether SMS mode is currently active.

---

`\stex_smsmode_set_codes:`

---

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

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

---

**`\stex_in_smsmode:nn`**

---

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

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

#### Test 7

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

### 13.1.2 Imports and Inheritance

---

**`\importmodule`**

---

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

Imports a module by reading it from a file and “activating” it.  $\text{\TeX}$  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://stextest?Foo?foo}<
                Meaning: >macro:->\protect \bar <
Module 15:      Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?Foo?foo}<
Module 16:      Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?Foo?foo}<
```

---

**`\usemodule`**

---

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

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



---

---

`\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://stextest?SymdeclTest?foo}<
Result: file://stextest?SymdeclTest?foo
Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?SymdeclTest?bardef}<
```

---

---

`\l_stex_all_symbols_seq`

Stores full URIs for all modules currently in scope.

---

---

`\stex_get_symbol:n`

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

---

---

`\notation`

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

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

---

---

`\stex_notation_do:nn`

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

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

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

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

### Test 12

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

Module 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  $\text{\S T E X}$  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  $\text{\S T E X}$  for automated bracketing (by default  $($  and  $)$ ) to  $\langle left \rangle$  and  $\langle right \rangle$ .

Note that  $\langle left \rangle$  and  $\langle right \rangle$  need to be allowed after `\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}_ {com
$\foobar a{b,c,d,e,f}g$ 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  $\langle a \mid [b]^c \rangle$

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

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

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

---

`\stex_term_custom:nn`

---

`\stex_term_custom:nn{⟨URI⟩}{⟨args⟩}`

Implements custom one-time notation. Invoked by `\stex_invoke_symbol:n` in text mode, or if followed by  $*$  in math mode, or whenever followed by  $!$ .

## Test 16

```
\begin{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`    `\comp{<args>}`

---

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

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

---

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

---



---

`\ellipses`    TODO

---

## Chapter 16

# TeX-Structural Features

Code related to structural features

### 16.1 Macros and Environments

#### 16.1.1 Structures

`mathstructure` TODO

## Chapter 17

# sTeX-Statements

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

### 17.1 Macros and Environments

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



## Chapter 18

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

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

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

## Contents

## 18.1 Introduction

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

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

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

Example 1: A very explicit proof, marked up semantically

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

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

## 18.2 The User Interface

### 18.2.1 Package Options

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

### 18.2.2 Proofs and Proof steps

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

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

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

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

### 18.2.3 Justifications

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

#### 18.2.4 Proof Structure

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

### 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 L<sup>A</sup>T<sub>E</sub>X `\@for...:=... \do{...}` macro; see Figure 2 for examples.

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

Figure 2: Configuration Proof Step Label Styles

## 18.3 Limitations

In this section we document known limitations. If you want to help alleviate them, please feel free to contact the package author. Some of them are currently discussed in the S<sub>T</sub>E<sub>X</sub> issue tracker at [\[sTeX\]](#).

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

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

## Chapter 19

# sTeX-Metatheory

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

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

### 19.1 Symbols



**Part III**  
**Extensions**

## Chapter 20

# Tikzinput

### 20.1 Macros and Environments

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

## Chapter 21

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

The `document-structure` package is part of the  $\S\TeX$  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 $\text{\LaTeX}$ packages

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

### 21.2.2 Document Structure

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

```

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

```

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

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

`blindomgroup`

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

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

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

Example 3: A typical Document Structure of a Book

`\skipomgroup`

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

`\currentsectionlevel`  
`\CurrentSectionLevel`

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

---

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

### 21.2.3 Ignoring Inputs

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

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

**narrative/content markup** In  $\text{\LaTeX}$  we mark up narrative-structured documents. In the generated OMDoc documents we want to be able to cache content objects that are not directly visible. For instance in the `statements` package [Koh20d] we use the `\inlinedef` macro to mark up phrase-level definitions, which verbalize more formal definitions. The latter can be hidden by an `ignore` and referenced by the `verbalizes` key in `\inlinedef`.

`\prematurestop` For prematurely stopping the formatting of a document,  $\text{\LaTeX}$  provides the `\prematurestop` macro. It can be used everywhere in a document and ignores all input after that – backing out of the `omgroup` environment as needed. After that – and before the implicit `\end{document}` it calls the internal `\afterprematurestop`, which can be customized to do additional cleanup or e.g. print the bibliography.

`\afterprematurestop` `\prematurestop` is useful when one has a driver file, e.g. for a course taught multiple years and wants to generate course notes up to the current point in the lecture. Instead of commenting out the remaining parts, one can just move the `\prematurestop` macro. This is especially useful, if we need the rest of the file for processing, e.g. to generate a theory graph of the whole course with the already-covered parts marked up as an overview over the progress; see `import_graph.py` from the `lmhtools` utilities [LMH].

### 21.2.4 Structure Sharing

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

`\STRsemantics` The `\STRlabel` macro has a variant `\STRsemantics`, where the label argument is optional, and which takes a third argument, which is ignored in  $\text{\LaTeX}$ . This allows to specify the meaning of the content (whatever that may mean) in cases, where the source document is not formatted for presentation, but is transformed into some content markup format.<sup>10</sup>

### 21.2.5 Global Variables

Text fragments and modules can be made more re-usable by the use of global variables. For instance, the admin section of a course can be made course-independent (and therefore re-usable) by using variables (actually token registers) `courseAcronym` and `courseTitle` instead of the text itself. The variables can then be set in the  $\text{\LaTeX}$  preamble of the course notes file. `\setSGvar{\langle vname \rangle}{\langle text \rangle}` to set the global variable `\langle vname \rangle` to `\langle text \rangle` and `\useSGvar{\langle vname \rangle}` to reference it.

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

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

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

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

### 21.2.6 Colors

For convenience, the `document-structure` package defines a couple of color macros for the `color` package: For instance `\blue` abbreviates `\textcolor{blue}`, so that `\blue{⟨something⟩}` writes `⟨something⟩` in blue. The macros `\red`, `\green`, `\cyan`, `\magenta`, `\brown`, `\yellow`, `\orange`, `\gray`, and finally `\black` are analogous.

## 21.3 Limitations

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

1. when option `book` which uses `\pagestyle{headings}` is given and semantic macros are given in the `omgroup` titles, then they sometimes are not defined by the time the heading is formatted. Need to look into how the headings are made.

## Chapter 22

# NotesSlides – Slides and Course Notes

We present a document class from which we can generate both course slides and course notes in a transparent way.

### 22.1 Introduction

The `notesslides` document class is derived from `beamer.cls` [Tana], it adds a “notes version” for course notes derived from the `omdoc` class [Kohlhase:smomdl] that is more suited to printing than the one supplied by `beamer.cls`.

### 22.2 The User Interface

The `notesslides` class takes the notion of a slide frame from Till Tantau’s excellent `beamer` class and adapts its notion of frames for use in the  $\text{\LaTeX}$  and OMDoc. To support semantic course notes, it extends the notion of mixing frames and explanatory text, but rather than treating the frames as images (or integrating their contents into the flowing text), the `notesslides` package displays the slides as such in the course notes to give students a visual anchor into the slide presentation in the course (and to distinguish the different writing styles in slides and course notes).

In practice we want to generate two documents from the same source: the slides for presentation in the lecture and the course notes as a narrative document for home study. To achieve this, the `notesslides` class has two modes: *slides mode* and *notes mode* which are determined by the package option.

#### 22.2.1 Package Options

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

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



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

### 22.2.2 Notes and Slides

`frame` Slides are represented with the `frame` just like in the `beamer` class, see [Tanb] for details.  
`note` The `notesslides` class adds the `note` environment for encapsulating the course note fragments.<sup>4</sup>

⚠ Note that it is essential to start and end the `notes` environment at the start of the line – in particular, there may not be leading blanks – else L<sup>A</sup>T<sub>E</sub>X becomes confused and throws error messages that are difficult to decipher.

```
\ifnotes\maketitle\else
\frame[noframenumbering]\maketitle\fi

\begin{note}
  We start this course with ...
\end{note}

\begin{frame}
  \frametitle{The first slide}
  ...
\end{frame}
\begin{note}
  ... and more explanatory text
\end{note}

\begin{frame}
  \frametitle{The second slide}
  ...
\end{frame}
...
```

Example 4: A typical Course Notes File

By interleaving the `frame` and `note` environments, we can build course notes as shown in Figure 4.

`\ifnotes` Note the use of the `\ifnotes` conditional, which allows different treatment between

<sup>11</sup>EDNOTE: leaving out `noproblems` for the moment until we decide what to do with it.

<sup>4</sup>MK: it would be very nice, if we did not need this environment, and this should be possible in principle, but not without intensive L<sup>A</sup>T<sub>E</sub>X trickery. Hints to the author are welcome.

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

⚠: We need to give the title frame the `noframenumbering` option so that the frame numbering is kept in sync between the slides and the course notes.

⚠: The `beamer` class recommends not to use the `allowframebreaks` option on frames (even though it is very convenient). This holds even more in the `notesslides` case: At least in conjunction with `\newpage`, frame numbering behaves funnily (we have tried to fix this, but who knows).

If we want to transclude a the contents of a file as a note, we can use a new variant `\inputref*` of the `\inputref` macro from [KGA20]: `\inputref*{foo}` is equivalent to `\begin{note}\inputref{foo}\end{note}`.

There are some environments that tend to occur at the top-level of `note` environments. We make convenience versions of these: e.g. the `nparagraph` environment is just an `sparagraph` inside a `note` environment (but looks nicer in the source, since it avoids one level of source indenting). Similarly, we have the `nomgroup`, `ndefinition`, `nexample`, `nsproof`, and `nassertion` environments.

### 22.2.3 Header and Footer Lines of the Slides

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

The default footer line of the `notesslides` package mentions copyright and licensing. In the `beamer` class, `\source` stores the author's name as the copyright holder . By default it is *Michael Kohlhase* in the `notesslides` package since he is the main user and designer of this package. `\setsource{<name>}` can change the writer's name. For licensing, we use the Creative Commons Attribution-ShareAlike license by default to strengthen the public domain. If package `hyperref` is loaded, then we can attach a hyperlink to the license logo. `\setlicensing[<url>]{<logo name>}` is used for customization, where `<url>` is optional.

### 22.2.4 Frame Images

Sometimes, we want to integrate slides as images after all – e.g. because we already have a PowerPoint presentation, to which we want to add  $\text{\TeX}$ notes. In this case we can use `\frameimage[<opt>]{<path>}`, where `<opt>` are the options of `\includegraphics` from the `graphicx` package [CR99] and `<path>` is the file path (extension can be left off like in `\includegraphics`). We have added the `label` key that allows to give a frame label that can be referenced like a regular `beamer` frame.<sup>12</sup>

The `\mhframeimage` macro is a variant of `\frameimage` with repository support. Instead of writing

```
\frameimage{\MathHub{fooMH/bar/source/baz/foobar}}
```

we can simply write (assuming that `\MathHub` is defined as above)

```
\mhframeimage[fooMH/bar]{baz/foobar}
```


<sup>12</sup>EdNOTE: MK: the `hyperref` link does not seem to work yet. I wonder why but do not have the time to fix it.

Note that the `\mhframeimage` form is more semantic, which allows more advanced document management features in MathHub.

If `baz/foobar` is the “current module”, i.e. if we are on the MathHub path `...MathHub/fooMH/bar...`, then stating the repository in the first optional argument is redundant, so we can just use

```
\mhframeimage{baz/foobar}
```

## 22.2.5 Colors and Highlighting

`\textwarning` The `\textwarning` macro generates a warning sign: 

## 22.2.6 Front Matter, Titles, etc.

### 22.2.7 Excursions

In course notes, we sometimes want to point to an “excursion” – material that is either presupposed or tangential to the course at the moment – e.g. in an appendix. The typical setup is the following:

```
\excursion{founif}{../ex/founif}{We will cover first-order unification in}
...
\begin{appendix}\printexcursions\end{appendix}
```

```
\excursion      The \excursion{<ref>}{<path>}{<text>} is syntactic sugar for
\activateexcursion
\begin{nparagraph}[title=Excursion]
  \activateexcursion{founif}{../ex/founif}
  We will cover first-order unification in \sref{founif}.
\end{nparagraph}
```

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

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

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

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

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

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

### 22.2.8 Miscellaneous

## 22.3 Limitations

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

1. when option `book` which uses `\pagestyle{headings}` is given and semantic macros are given in the `omgroup` titles, then they sometimes are not defined by the time the heading is formatted. Need to look into how the headings are made. This is a problem of the underlying `omdoc` package.

## Chapter 23

# problem.sty: An Infrastructure for formatting Problems

The `problem` package supplies an infrastructure that allows specify problems and to reuse them efficiently in multiple environments.

### 23.1 Introduction

The `problem` package supplies an infrastructure that allows specify problem. Problems are text fragments that come with auxiliary functions: hints, notes, and solutions<sup>5</sup>. Furthermore, we can specify how long the solution to a given problem is estimated to take and how many points will be awarded for a perfect solution.

Finally, the `problem` package facilitates the management of problems in small files, so that problems can be re-used in multiple environment.

### 23.2 The User Interface

#### 23.2.1 Package Options

<code>solutions</code>	The <code>problem</code> package takes the options <code>solutions</code> (should solutions be output?), <code>notes</code>
<code>notes</code>	(should the problem notes be presented?), <code>hints</code> (do we give the hints?), <code>gnotes</code> (do we
<code>hints</code>	show grading notes?), <code>pts</code> (do we display the points awarded for solving the problem?),
<code>gnotes</code>	<code>min</code> (do we display the estimated minutes for problem soling). If theses are specified, then
<code>pts</code>	the corresponding auxiliary parts of the problems are output, otherwise, they remain
<code>min</code>	invisible.
<code>boxed</code>	The <code>boxed</code> option specifies that problems should be formatted in framed boxes so
<code>test</code>	that they are more visible in the text. Finally, the <code>test</code> option signifies that we are in
	a test situation, so this option does not show the solutions (of course), but leaves space
	for the students to solve them.
<code>mh</code>	The <code>mh</code> option turns on MathHub support; see [ <code>Kohlhase:mss</code> ].
<code>showmeta</code>	Finally, if the <code>showmeta</code> is set, then the metadata keys are shown (see [ <code>Kohlhase:metakeys</code> ]
	for details and customization options).

---

<sup>5</sup>for the moment multiple choice problems are not supported, but may well be in a future version

### 23.2.2 Problems and Solutions

**problem** The main environment provided by the **problem** package is (surprise surprise) the **problem** environment. It is used to mark up problems and exercises. The environment takes an optional KeyVal argument with the keys **id** as an identifier that can be reference later, **pts** for the points to be gained from this exercise in homework or quiz situations, **min** for the estimated minutes needed to solve the problem, and finally **title** for an informative title of the problem. For an example of a marked up problem see Figure 5 and the resulting markup see Figure 6.

```
\usepackage[solutions,hints,pts,min]{problem}
\begin{document}
  \begin{sproblem}[id=elephants,pts=10,min=2,title=Fitting Elephants]
    How many Elephants can you fit into a Volkswagen beetle?
  \begin{hint}
    Think positively, this is simple!
  \end{hint}
  \begin{exnote}
    Justify your answer
  \end{exnote}
  \begin{solution}[for=elephants,height=3cm]
    Four, two in the front seats, and two in the back.
  \begin{gnote}
    if they do not give the justification deduct 5 pts
  \end{gnote}
  \end{solution}
  \end{sproblem}
\end{document}
```

Example 5: A marked up Problem

**solution** The **solution** environment can be to specify a solution to a problem. If the **solutions** option is set or **\solutionstrue** is set in the text, then the solution will be presented in the output. The **solution** environment takes an optional KeyVal argument with the keys **id** for an identifier that can be reference **for** to specify which problem this is a solution for, and **height** that allows to specify the amount of space to be left in test situations (i.e. if the **test** option is set in the **\usepackage** statement).

```
Problem 0.1 (Fitting Elephants)
How many Elephants can you fit into a Volkswagen beetle?


---


Hint: Think positively, this is simple!


---


Note:Justify your answer


---


Solution: Four, two in the front seats, and two in the back.


---


```

Example 6: The Formatted Problem from Figure 5

**hint** The **hint** and **exnote** environments can be used in a **problem** environment to give hints and to make notes that elaborate certain aspects of the problem.

**exnote**

**gnote** The **gnote** (grading notes) environment can be used to document situations that

may arise in grading.

Sometimes we would like to locally override the `solutions` option we have given to the package. To turn on solutions we use the `\startsolutions`, to turn them off, `\stopsolutions`. These two can be used at any point in the documents.

Also, sometimes, we want content (e.g. in an exam with master solutions) conditional on whether solutions are shown. This can be done with the `\ifsolutions` conditional.

### 23.2.3 Multiple Choice Blocks

Multiple choice blocks can be formatted using the `mcb` environment, in which single choices are marked up with `\mcc[⟨keyvals⟩]{⟨text⟩}` macro, which takes an optional key/value argument `⟨keyvals⟩` for choice metadata and a required argument `⟨text⟩` for the proposed answer text. The following keys are supported

- `T` • `T` for true answers, `F` for false ones,
- `F` • `Ttext` the verdict for true answers, `Ftext` for false ones, and
- `Ttext` • `feedback` for a short feedback text given to the student.
- `Ftext`
- `feedback`

See Figure ?? for an example

### 23.2.4 Including Problems

The `\includeproblem` macro can be used to include a problem from another file. It takes an optional `KeyVal` argument and a second argument which is a path to the file containing the problem (the macro assumes that there is only one problem in the include file). The keys `title`, `min`, and `pts` specify the problem title, the estimated minutes for solving the problem and the points to be gained, and their values (if given) overwrite the ones specified in the `problem` environment in the included file.

### 23.2.5 Reporting Metadata

The sum of the points and estimated minutes (that we specified in the `pts` and `min` keys to the `problem` environment or the `\includeproblem` macro) to the log file and the screen after each run. This is useful in preparing exams, where we want to make sure that the students can indeed solve the problems in an allotted time period.

The `\min` and `\pts` macros allow to specify (i.e. to print to the margin) the distribution of time and reward to parts of a problem, if the `pts` and `pts` package options are set. This allows to give students hints about the estimated time and the points to be awarded.

## 23.3 Limitations

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

1. none reported yet

```

\begin{sproblem}[title=Functions]
  What is the keyword to introduce a function definition in python?
  \begin{mcb}
    \mcc[T]{def}
    \mcc[F,feedback=that is for C and C++){function}
    \mcc[F,feedback=that is for Standard ML]{fun}
    \mcc[F,Ftext=Noooooooooooo,feedback=that is for Java]{public static void}
  \end{mcb}
\end{sproblem}

```

---

**Problem 0.2 (Functions)**

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

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

---

**Problem 0.3 (Functions)**

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

1. def  
!
2. function  
that is for C and C++
3. fun  
that is for Standard ML
4. public static void  
that is for Java

Example 7: A Problem with a multiple choice block



## Chapter 24

# **`hwexam.sty/cls`: An Infrastructure for formatting Assignments and Exams**

The `hwexam` package and class allows individual course assignment sheets and compound assignment documents using problem files marked up with the `problem` package.

### Contents

## 24.1 Introduction

The `hwexam` package and class supplies an infrastructure that allows to format nice-looking assignment sheets by simply including problems from problem files marked up with the `problem` package [Kohlhase:problem]. It is designed to be compatible with `problems.sty`, and inherits some of the functionality.

## 24.2 The User Interface

### 24.2.1 Package and Class Options

The `hwexam` package and class take the options `solutions`, `notes`, `hints`, `gnotes`, `pts`, `min`, and `boxed` that are just passed on to the `problems` package (cf. its documentation for a description of the intended behavior).

`showmeta` If the `showmeta` option is set, then the metadata keys are shown (see [Kohlhase:metakeys] for details and customization options).

The `hwexam` class additionally accepts the options `report`, `book`, `chapter`, `part`, and `showignores`, of the `omdoc` package [Kohlhase:smomdl] on which it is based and passes them on to that. For the `extrefs` option see [Kohlhase:sref].

### 24.2.2 Assignments

`assignment` This package supplies the `assignment` environment that groups problems into assignment sheets. It takes an optional KeyVal argument with the keys `number` (for the assignment number; if none is given, 1 is assumed as the default or — in multi-assignment documents  
`number` — the ordinal of the `assignment` environment), `title` (for the assignment title; this is referenced in the title of the assignment sheet), `type` (for the assignment type; e.g. “quiz”, or “homework”), `given` (for the date the assignment was given), and `due` (for the date the assignment is due).

### 24.2.3 Typesetting Exams

`multiple` Furthermore, the `hwexam` package takes the option `multiple` that allows to combine multiple assignment sheets into a compound document (the assignment sheets are treated as section, there is a table of contents, etc.).

`test` Finally, there is the option `test` that modifies the behavior to facilitate formatting tests. Only in `test` mode, the macros `\testspace`, `\testnewpage`, and `\testemptypage` have an effect: they generate space for the students to solve the given problems. Thus they can be left in the L<sup>A</sup>T<sub>E</sub>X source.

`\testspace` `\testspace` takes an argument that expands to a dimension, and leaves vertical space accordingly. `\testnewpage` makes a new page in `test` mode, and `\testemptypage` generates an empty page with the cautionary message that this page was intentionally left empty.

`testheading` Finally, the `\testheading` takes an optional keyword argument where the keys  
`duration` `duration` specifies a string that specifies the duration of the test, `min` specifies the equivalent in number of minutes, and `reqpts` the points that are required for a perfect grade.  
`min`  
`reqpts`

### 24.2.4 Including Assignments

`\inputassignment` The `\inputassignment` macro can be used to input an assignment from another file. It takes an optional `KeyVal` argument and a second argument which is a path to the file containing the problem (the macro assumes that there is only one `assignment` environment in the included file). The keys `number`, `title`, `type`, `given`, and `due` are just as for the `assignment` environment and (if given) overwrite the ones specified in the `assignment` environment in the included file.

## 24.3 Limitations

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

1. none reported yet.



**Part IV**  
**Implementation**

## Chapter 25

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

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

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

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

### 25.2 Preliminaries

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

Package options:

```

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

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

```

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

## 25.3 Messages and logging

```

42 <@@=stex_log>

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

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

```

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

Redirecting messages:

```

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

```

## 25.4 Persistence

77 `<@=stex_persist>`

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

```

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

```

(End definition for `\c__stex_persist_sms_iow`.)

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

```

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

```

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

## 25.5 HTML Annotations

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

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

```

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

```

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

```

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

```



```

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

```

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

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

```

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

```

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

`\_stex_annotate_checkempty:n`

```

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

```

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

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

```

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

```

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

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

```

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

```

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

`\stex_annotate:env`

`\stex_annotate_invisible:n`

`\stex_annotate_invisible:nnn`

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

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

```

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

```

```

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

```

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

## 25.6 Languages

```

233 <@=stex_language>

```

`\c_stex_languages_prop`  
`\c_stex_language_abbrevs_prop`

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

```

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

```

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

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

```

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

```

## 25.7 Activating/Deactivating Macros

`\stex_deactivate_macro:Nn`

```

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

```

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

`\stex_reactivate_macro:N`

```

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

```

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

`\stex_do_aftergroup:nn`

```

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

```

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

```

300 </package>

```

## Chapter 26

# STEX -MathHub Implementation

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

### 26.1 Generic Path Handling

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

```

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

```

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

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

```

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

```

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

## 26.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

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

```

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

We determine the PWD

`\c_stex_pwd_seq`  
`\c_stex_pwd_str`

```

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

```



```

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

```

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

## 26.3 File Hooks and Tracking

```

409 <@@=stex_files>

```

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

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

`\g_stex_files_stack` keeps track of file changes

```

410 \seq_gclear_new:N\g_stex_files_stack

```

(End definition for `\g_stex_files_stack`.)

`\c_stex_mainfile_seq`  
`\c_stex_mainfile_str`

```

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

```

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

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

```

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

```

```

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

```

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

## 26.4 MathHub Repositories

```

443 <@@=stex_mathhub>

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

```

(End definition for `\mathhub`, `\c_stex_mathhub_seq`, and `\c_stex_mathhub_str`. These variables are documented on page 23.)

```

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

```

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

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

480 \str\_new:N\l\_stex\_mathhub\_manifest\_file\_seq

(End definition for \l\_stex\_mathhub\_manifest\_file\_seq.)

\\_stex\_mathhub\_find\_manifest:N Attempts to find the MANIFEST.MF in some file path and stores its path in \l\_stex\_mathhub\_manifest\_file\_seq:

```

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

```

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

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

516 \ior\_new:N \c\_stex\_mathhub\_manifest\_ior

(End definition for \c\_stex\_mathhub\_manifest\_ior.)

`\_stex_mathhub_parse_manifest:n` Stores the entries in manifest file in the corresponding property list:

```

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

```

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

`\stex_set_current_repository:n`

```

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

```

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

`\stex_require_repository:n`

```

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

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

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

```

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

`\stex_in_repository:nn` Executes the code in the second argument in the context of the repository whose ID is provided as the first argument.

```

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

```

```

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

```

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

```

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

```

```

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

```

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

#### `\mhpath`

```

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

```

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

#### `\libinput`

```

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

```

```

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

```

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

`\libusepackage`

```

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

```



```

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

```

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

```

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

```

## Chapter 27

# STEX -References Implementation

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

### 27.1 Document URIs and URLs

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

```

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

```

```

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

```

## 27.2 Setting Reference Targets

```

896 \str_const:Nn \c__stex_refs_url_str{URL}
897 \str_const:Nn \c__stex_refs_ref_str{REF}
898 % @currentlabel -> number
899 % @currentlabelname -> title
900 % @currentHref -> name.number <- id of some kind
901 % \theH# -> \arabic{section}
902 % \the# -> number
903 % \hyper@makecurrent{#}
904 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
905   \stex_get_document_uri:
906   \str_set:Nx \l_tmpa_str { #1 }
907   \str_if_empty:NT \l_tmpa_str {
908     \int_zero:N \l_tmpa_int
909     \bool_set_true:N \l_tmpa_bool
910     \bool_while_do:Nn \l_tmpa_bool {
911       \cs_if_exist:cTF {
912         sref_\l_stex_current_docns_str?? REF_\int_use:N \l_tmpa_int _type
913       }{
914         \int_incr:N \l_tmpa_int
915       }{
916         \str_set:Nx \l_tmpa_str { REF_\int_use:N \l_tmpa_int }
917         \bool_set_false:N \l_tmpa_bool
918       }
919     }
920   }
921   \str_set:Nx \l_tmpa_str {
922     \l_stex_current_docns_str??\l_tmpa_str

```

```

923 }
924 \seq_gput_right:No \g__stex_refs_all_refs_seq \l_tmpa_str
925 \stex_if_smsmode:TF {
926   \stex_get_document_url:
927   \str_gset_eq:cN {sref_url_\l_tmpa_str _str}\l_stex_current_docurl_str
928   \str_gset_eq:cN {sref_\l_tmpa_str _type}\c__stex_refs_url_str
929 }{
930   \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter\unexpanded\expandafter{
931   \exp_args:Nx\label{sref_\l_tmpa_str}
932   \exp_args:NNNx\immediate\write\@auxout{\stexauxadddocref{\l_tmpa_str}}
933   \str_gset:cx {sref_\l_tmpa_str _type}\c__stex_refs_ref_str
934 }
935 }
936 \cs_new_protected:Npn \stexauxadddocref #1 {
937   \str_set:Nx \l_tmpa_str {#1}
938   \str_gset_eq:cN{sref_\l_tmpa_str _type}\c__stex_refs_ref_str
939   \seq_gput_right:Nx \g__stex_refs_all_refs_seq {\l_tmpa_str}
940 }
941 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
942   \stex_get_document_uri:
943   \stex_if_smsmode:TF {
944     \stex_get_document_url:
945     \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str
946     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
947   }
948   }{
949     \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter{\@currentlabel\iffalse}}
950     \exp_args:Nx\label{sref_sym_#1}
951
952     \exp_args:NNNx\immediate\write\@auxout{\stexauxadddocref{sym_#1}}
953     \str_gset:cx {sref_sym_#1_type}\c__stex_refs_ref_str
954   }
955 }

```

## 27.3 Using References

```

956 \str_new:N \l__stex_refs_indocument_str
957 \keys_define:nn { stex / sref } {
958   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
959   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
960   pre           .tl_set:N = \l__stex_refs_pre_tl ,
961   post          .tl_set:N = \l__stex_refs_post_tl ,
962   %indoc        .str_set_x:N = \l__stex_refs_repo_str ,
963 }
964
965 \bool_new:N \c__stex_refs_hyperref_bool
966 \bool_set_false:N \c__stex_refs_hyperref_bool
967 \AddToHook{begindocument}{
968   \@ifpackageloaded{hyperref}{
969     \bool_set_true:N \c__stex_refs_hyperref_bool
970   }{}
971 }
972
973

```

```

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

```

```

1028 \str_if_empty:NTF \l__stex_refs_indocument_str {
1029   \tl_set:Nn \l_tmpa_tl {
1030     \l__stex_refs_fallback_tl
1031   }
1032   \tl_if_exist:cT{sref_sym_\l_stex_get_symbol_uri_str_type}{
1033     \tl_set:Nn \l_tmpa_tl {
1034       % doc uri in \l_tmpb_str
1035       \str_set:Nx \l_tmpa_str {\use:c{sref_sym_\l_stex_get_symbol_uri_str_type}}
1036       \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1037         % reference
1038         \cs_if_exist:cTF{autoref}{
1039           \l__stex_refs_pre_tl\autoref{sref_sym_\l_stex_get_symbol_uri_str}\l__stex_refs_post_tl
1040         }{
1041           \l__stex_refs_pre_tl\ref{sref_sym_\l_stex_get_symbol_uri_str}\l__stex_refs_post_tl
1042         }
1043       }{
1044         % URL
1045         \if_bool:N \c__stex_refs_hyperref_bool {
1046           \exp_args:Nx \href{\use:c{sref_sym_url_\l_stex_get_symbol_uri_str_str}}{\l__stex_refs_ref_str}
1047         }{
1048           \l__stex_refs_fallback_tl
1049         }
1050       }
1051     }
1052   }
1053   \l_tmpa_tl
1054 }{
1055   % TODO
1056 }
1057 }
1058
1059 \cs_new_protected:Npn \srefsymuri #1 #2 {
1060   \hyperref[sref_sym_#1]{#2}
1061 }
1062
1063 </package>

```

## Chapter 28

# STEX -Modules Implementation

```
1064 <*package>
1065
1066 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1067
1068 <@@=stex_modules>
1069
1070 Warnings and error messages
1071 \msg_new:nnn{stex}{error/unknownmodule}{
1072   No~module~#1~found
1073 }
1074 \msg_new:nnn{stex}{error/syntax}{
1075   Syntax~error:~#1
1076 }
1077 \msg_new:nnn{stex}{error/siglanguage}{
1078   Module~#1~declares~signature~#2,~but~does~not~
1079   declare~its~language
1080 }
1081 \msg_new:nnn{stex}{error/conflictingmodules}{
1082   Conflicting~imports~for~module~#1
1083 }
1084
1085 \l_stex_current_module_str The current module:
1086 \str_new:N \l_stex_current_module_str
1087
1088 (End definition for \l_stex_current_module_str. This variable is documented on page 26.)
1089
1090 \l_stex_all_modules_seq Stores all available modules
1091 \seq_new:N \l_stex_all_modules_seq
1092
1093 (End definition for \l_stex_all_modules_seq. This variable is documented on page 26.)
1094
1095 \stex_if_in_module_p:
1096 \stex_if_in_module:TF
1097 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1098   \str_if_empty:NTF \l_stex_current_module_str
1099   \prg_return_false: \prg_return_true:
1100 }
```



(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`

```
1089 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1090   \prop_if_exist:cTF { c_stex_module_#1_prop }
1091   \prg_return_true: \prg_return_false:
1092 }
```

(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:

```
1093 \cs_new_protected:Nn \stex_add_to_current_module:n {
1094   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1095 }
1096 \cs_new_protected:Npn \STEXexport {
1097   \begingroup
1098   \newlinechar=-1\relax
1099   \endlinechar=-1\relax
1100   %\catcode'\ = 9\relax
1101   \expandafter\endgroup\STEXexport:n
1102 }
1103 \cs_new_protected:Nn \STEXexport:n {
1104   \ignorespaces #1
1105   \stex_add_to_current_module:n { \ignorespaces #1 }
1106   \stex_smsmode_do:
1107 }
1108 \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`

```
1109 \cs_new_protected:Nn \stex_add_constant_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 _constants} { \l_tmpa_str }
1112 }
1113
1114 %\cs_new_protected:Nn \stex_add_field_to_current_module:n {
1115 % \str_set:Nx \l_tmpa_str { #1 }
1116 % \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _fields} { \l_tmpa_str }
1117 %}
```

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

`\stex_collect_imports:n`

```
1118 \cs_new_protected:Nn \stex_collect_imports:n {
1119   \seq_clear:N \l_stex_collect_imports_seq
1120   \__stex_modules_collect_imports:n {#1}
1121 }
1122 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1123   \seq_map_inline:cn {c_stex_module_#1_imports} {
1124     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1125       \__stex_modules_collect_imports:n { ##1 }
1126     }
1127 }
```

```

1127 }
1128 \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1129   \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1130 }
1131 }

```

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

`\stex_add_import_to_current_module:n`

```

1132 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1133   \str_set:Nx \l_tmpa_str { #1 }
1134   \exp_args:Nno
1135   \seq_if_in:cnF{c_stex_module\_l_stex_current_module_str_imports}\l_tmpa_str{
1136     \seq_gput_right:co{c_stex_module\_l_stex_current_module_str_imports}\l_tmpa_str
1137   }
1138 }

```

(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).

```

1139 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
1140   \str_set:Nx \l_tmpa_str { #1 }
1141   \seq_set_eq:NN \l_tmpa_seq #2
1142   % split off file extension
1143   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1144   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1145   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1146   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1147
1148   \bool_set_true:N \l_tmpa_bool
1149   \bool_while_do:Nn \l_tmpa_bool {
1150     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1151     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1152       {source} { \bool_set_false:N \l_tmpa_bool }
1153     }{}{
1154       \seq_if_empty:NT \l_tmpa_seq {
1155         \bool_set_false:N \l_tmpa_bool
1156       }
1157     }
1158   }
1159
1160   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
1161   \str_if_empty:NTF \l_stex_modules_subpath_str {
1162     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
1163   }{
1164     \str_set:Nx \l_stex_modules_ns_str {
1165       \l_tmpa_str/\l_stex_modules_subpath_str
1166     }
1167   }
1168 }

```

(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
1169 \str_new:N \l_stex_modules_ns_str
1170 \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.

```

1171 \cs_new_protected:Nn \stex_modules_current_namespace: {
1172   \str_clear:N \l_stex_modules_subpath_str
1173   \prop_if_exist:NTF \l_stex_current_repository_prop {
1174     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1175     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1176   }{
1177     % split off file extension
1178     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1179     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1180     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1181     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1182     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1183     \str_set:Nx \l_stex_modules_ns_str {
1184       file:/\stex_path_to_string:N \l_tmpa_seq
1185     }
1186   }
1187 }

```

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

## 28.1 The module environment

module arguments:

```

1188 \keys_define:nn { stex / module } {
1189   title      .tl_set:N      = \smodulename ,
1190   type       .str_set_x:N   = \smodulename ,
1191   id         .str_set_x:N   = \smoduleid ,
1192   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1193   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1194   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1195   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1196   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1197   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1198   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1199 }
1200
1201 \cs_new_protected:Nn \__stex_modules_args:n {
1202   \str_clear:N \smodulename
1203   \str_clear:N \smodulename
1204   \str_clear:N \smoduleid
1205   \str_clear:N \l_stex_module_ns_str
1206   \str_clear:N \l_stex_module_lang_str
1207   \str_clear:N \l_stex_module_sig_str
1208   \str_clear:N \l_stex_module_creators_str

```

```

1209 \str_clear:N \l_stex_module_contributors_str
1210 \str_clear:N \l_stex_module_meta_str
1211 \str_clear:N \l_stex_module_srccite_str
1212 \keys_set:nn { stex / module } { #1 }
1213 }
1214
1215 % module parameters here? In the body?
1216

```

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

```

1217 \cs_new_protected:Nn \stex_module_setup:nn {
1218   \str_set:Nx \l_stex_module_name_str { #2 }
1219   \__stex_modules_args:n { #1 }

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

1220   \stex_if_in_module:TF {
1221     % Nested module
1222     \prop_get:cnN {c_stex_module\_l_stex_current_module_str _prop}
1223       { ns } \l_stex_module_ns_str
1224     \str_set:Nx \l_stex_module_name_str {
1225       \prop_item:cn {c_stex_module\_l_stex_current_module_str _prop}
1226         { name } / \l_stex_module_name_str
1227     }
1228   }{
1229     % not nested:
1230     \str_if_empty:NT \l_stex_module_ns_str {
1231       \stex_modules_current_namespace:
1232       \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1233       \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1234         / { \l_stex_module_ns_str }
1235       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1236       \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1237         \str_set:Nx \l_stex_module_ns_str {
1238           \stex_path_to_string:N \l_tmpa_seq
1239         }
1240       }
1241     }
1242   }

```

Next, we determine the language of the module:

```

1243 \str_if_empty:NT \l_stex_module_lang_str {
1244   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1245   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1246   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1247   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1248   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1249     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1250       inferred~from~file~name}
1251     \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1252   }
1253 }
1254
1255 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {

```

```

1256 \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1257 \l_tmpa_str {
1258   \ltx@ifpackageloaded{babel}{
1259     \exp_args:Nx \selectlanguage { \l_tmpa_str }
1260   }{}
1261 } {
1262   \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
1263 }
1264 }}

```

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

```

1265 \str_if_empty:NTF \l_stex_module_sig_str {
1266   \exp_args:Nnx \prop_gset_from_keyval:cn {
1267     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1268   } {
1269     name      = \l_stex_module_name_str ,
1270     ns        = \l_stex_module_ns_str ,
1271     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1272     lang      = \l_stex_module_lang_str ,
1273     sig       = \l_stex_module_sig_str ,
1274     meta      = \l_stex_module_meta_str
1275   }
1276   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1277   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _fields}
1278   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1279   \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1280   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1281 \str_if_empty:NT \l_stex_module_meta_str {
1282   \str_set:Nx \l_stex_module_meta_str {
1283     \c_stex_metatheory_ns_str ? Metatheory
1284   }
1285 }
1286 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1287   \bool_set_true:N \l_stex_in_meta_bool
1288   \exp_args:Nx \stex_add_to_current_module:n {
1289     \bool_set_true:N \l_stex_in_meta_bool
1290     \stex_activate_module:n {\l_stex_module_meta_str}
1291     \bool_set_false:N \l_stex_in_meta_bool
1292   }
1293   \stex_activate_module:n {\l_stex_module_meta_str}
1294   \bool_set_false:N \l_stex_in_meta_bool
1295 }
1296 }{
1297   \str_if_empty:NT \l_stex_module_lang_str {
1298     \msg_error:nnxx{stex}{error/siglanguage}{
1299       \l_stex_module_ns_str?\l_stex_module_name_str
1300     }{\l_stex_module_sig_str}
1301   }
1302
1303   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1304   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str

```

```

1305 \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1306 \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1307 \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1308 \str_set:Nx \l_tmpa_str {
1309   \stex_path_to_string:N \l_tmpa_seq /
1310   \l_tmpa_str . \l_stex_module_sig_str .tex
1311 }
1312 \IfFileExists \l_tmpa_str {
1313   \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1314     \str_clear:N \l_stex_current_module_str
1315     \seq_clear:N \l_stex_all_modules_seq
1316     \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1317   }
1318 }{
1319   \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1320 }
1321 \stex_if_smsmode:F {
1322   \stex_activate_module:n {
1323     \l_stex_module_ns_str ? \l_stex_module_name_str
1324   }
1325 }
1326 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1327 }
1328 }

```

(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}

1329 \int_new:N \l_stex_module_group_depth_int
1330 \cs_new_protected:Nn \__stex_modules_begin_module: {
1331   \stex_reactivate_macro:N \STEXexport
1332   \stex_reactivate_macro:N \importmodule
1333   \stex_reactivate_macro:N \symdecl
1334   \stex_reactivate_macro:N \notation
1335   \stex_reactivate_macro:N \symdef
1336
1337   \stex_debug:nn{modules}{
1338     New~module:\\
1339     Namespace:~\l_stex_module_ns_str\\
1340     Name:~\l_stex_module_name_str\\
1341     Language:~\l_stex_module_lang_str\\
1342     Signature:~\l_stex_module_sig_str\\
1343     Metatheory:~\l_stex_module_meta_str\\
1344     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1345   }
1346
1347   \seq_put_right:Nx \l_stex_all_modules_seq {
1348     \l_stex_module_ns_str ? \l_stex_module_name_str
1349   }
1350
1351   % \seq_gput_right:Nx \g_stex_modules_in_file_seq
1352   % { \l_stex_module_ns_str ? \l_stex_module_name_str }

```

```

1353
1354
1355 \stex_if_smsmode:F{
1356   \begin{stex_annotate_env} {theory} {
1357     \l_stex_module_ns_str ? \l_stex_module_name_str
1358   }
1359
1360   \stex_annotate_invisible:nnn{header}{} {
1361     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1362     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1363     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1364       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1365     }
1366     \str_if_empty:NF \smoduletype {
1367       \stex_annotate:nnn{type}{\smoduletype}{}
1368     }
1369   }
1370 }
1371 \int_set:Nn \l_stex_module_group_depth_int {\currentgrouplevel}
1372 % TODO: Inherit metatheory for nested modules?
1373 }
1374 \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}

1375 \cs_new_protected:Nn \_stex_modules_end_module: {
1376 % \str_set:Nx \l_tmpa_str {
1377 %   c_stex_module_
1378 %   \prop_item:Nn \l_stex_current_module_prop { ns } ?
1379 %   \prop_item:Nn \l_stex_current_module_prop { name }
1380 %   _prop
1381 % }
1382 %^^A \prop_new:c { \l_tmpa_str }
1383 % \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
1384 \stex_debug:nn{modules}{Closing-module-\prop_item:cn {c_stex_module_\l_stex_current_module_
1385 }

```

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

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

```

1386 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1387 \NewDocumentEnvironment { smodule } { 0{} m } {
1388   \stex_module_setup:nn{#1}{#2}
1389   \par
1390   \stex_if_smsmode:F{
1391     \tl_clear:N \l_tmpa_tl
1392     \clist_map_inline:Nn \smoduletype {
1393       \tl_if_exist:cT {\_stex_modules_smodule_##1_start:}{
1394         \tl_set:Nn \l_tmpa_tl {\use:c{\_stex_modules_smodule_##1_start:}}
1395       }
1396     }
1397     \tl_if_empty:NTF \l_tmpa_tl {
1398       \_stex_modules_smodule_start:

```

```

1399     }{
1400         \l_tmpa_tl
1401     }
1402 }
1403 \__stex_modules_begin_module:
1404 \stex_ref_new_doc_target:n \smoduleid
1405 \stex_smsmode_do:
1406 } {
1407 \__stex_modules_end_module:
1408 \stex_if_smsmode:TF {
1409 %     \exp_args:Nx \stex_add_to_sms:n {
1410 %         \prop_gset_from_keyval:cn {
1411 %             c_stex_module_
1412 %             \prop_item:Nn \l_stex_current_module_prop { ns } ?
1413 %             \prop_item:Nn \l_stex_current_module_prop { name }
1414 %             _prop
1415 %         } {
1416 %             name      = \prop_item:cn { \l_tmpa_str } { name } ,
1417 %             ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
1418 %             file      = \prop_item:cn { \l_tmpa_str } { file } ,
1419 %             lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
1420 %             sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
1421 %             meta      = \prop_item:cn { \l_tmpa_str } { meta }
1422 %         }
1423 %     }
1424 }{
1425     \end{stex_annotate_env}
1426     \clist_set:No \l_tmpa_clist \smoduletype
1427     \tl_clear:N \l_tmpa_tl
1428     \clist_map_inline:Nn \l_tmpa_clist {
1429         \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1430             \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1431         }
1432     }
1433     \tl_if_empty:NTF \l_tmpa_tl {
1434         \__stex_modules_smodule_end:
1435     }{
1436         \l_tmpa_tl
1437     }
1438 }
1439 }
1440
1441 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1442 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1443
1444 \newcommand\stexpatchmodule[3] [] {
1445     \str_set:Nx \l_tmpa_str{ #1 }
1446     \str_if_empty:NTF \l_tmpa_str {
1447         \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1448         \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1449     }{
1450         \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1451         \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1452     }

```



```
1453 }
1454
```

## 28.2 Invoking modules

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

```
1455 \NewDocumentCommand \STEXModule { m } {
1456   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1457   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1458   \tl_set:Nn \l_tmpa_tl {
1459     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1460   }
1461   \seq_map_inline:Nn \l_stex_all_modules_seq {
1462     \str_set:Nn \l_tmpb_str { ##1 }
1463     \str_if_eq:eeT { \l_tmpa_str } {
1464       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1465     } {
1466       \seq_map_break:n {
1467         \tl_set:Nn \l_tmpa_tl {
1468           \stex_invoke_module:n { ##1 }
1469         }
1470       }
1471     }
1472   }
1473   \l_tmpa_tl
1474 }
1475
1476 \cs_new_protected:Nn \stex_invoke_module:n {
1477   \stex_debug:nn{modules}{Invoking~module~#1}
1478   \peek_charcode_remove:NTF ! {
1479     \__stex_modules_invoke_uri:nN { #1 }
1480   } {
1481     \peek_charcode_remove:NTF ? {
1482       \__stex_modules_invoke_symbol:nn { #1 }
1483     } {
1484       \msg_error:nnx{stex}{error/syntax}{
1485         ?~or~!~expected~after~
1486         \c_backslash_str STEXModule{#1}
1487       }
1488     }
1489   }
1490 }
1491
1492 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1493   \str_set:Nn #2 { #1 }
1494 }
1495
1496 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1497   \stex_invoke_symbol:n{#1?#2}
1498 }
```

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

`\stex_activate_module:n`

```
1499 \bool_new:N \l_stex_in_meta_bool
1500 \bool_set_false:N \l_stex_in_meta_bool
1501 \cs_new_protected:Nn \stex_activate_module:n {
1502   \stex_debug:nn{modules}{Activating~module~#1}
1503   \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1504     \msg_error:nnn{stex}{error/conflictingmodules}{ #1 }
1505   }
1506   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1507     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1508     \use:c{ c_stex_module_#1_code }
1509   }
1510 }
```

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

```
1511 \</package>
```

## Chapter 29

# STEX -Module Inheritance Implementation

```
1512 <*package>
1513
1514 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1515
```

### 29.1 SMS Mode

```
1516 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1517 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1518 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1519 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1520
1521 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1522   \makeatletter
1523   \makeatother
1524   \ExplSyntaxOn
1525   \ExplSyntaxOff
1526   \rustexBREAK
1527 }
1528
1529 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1530   \symdef
1531   \importmodule
1532   \notation
1533   \symdecl
1534   \STEXexport
1535   \inlineass
1536   \inlinedef
1537   \inlineex
1538   \endinput
1539 }
```

```

1540
1541 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1542   \tl_to_str:n {
1543     smodule,
1544     copymodule,
1545     interpretmodule
1546     sdefinition,
1547     sexample,
1548     sassertion,
1549     sparagraph
1550   }
1551 }

```

(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
1552 \bool_new:N \g__stex_smsmode_bool
1553 \bool_set_false:N \g__stex_smsmode_bool
1554 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1555   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1556 }

```

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

`\stex_in_smsmode:nn`

```

1557 \cs_new_protected:Nn \stex_in_smsmode:nn {
1558   \vbox_set:Nn \l_tmpa_box {
1559     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1560     \bool_gset_true:N \g__stex_smsmode_bool
1561     #2
1562     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1563   }
1564   \box_clear:N \l_tmpa_box
1565 }
1566
1567 \quark_new:N \q__stex_smsmode_break
1568
1569 %\ior_new:N \c__stex_smsmode_ior
1570 %\tl_new:N \l__stex_smsmode_filecontent_tl
1571 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1572   % \tl_clear:N \l__stex_smsmode_filecontent_tl
1573   % \ior_open:Nn \c__stex_smsmode_ior {#1}
1574   % \ior_map_inline:Nn \c__stex_smsmode_ior {
1575     %   \tl_put_right:Nn \l__stex_smsmode_filecontent_tl { ##1 }
1576   % }
1577   % \ior_close:N \c__stex_smsmode_ior
1578   \stex_filestack_push:n{#1}
1579   \stex_in_smsmode:nn{#1} {
1580     #2
1581     \everyeof{\q__stex_smsmode_break\noexpand}
1582     \expandafter\expandafter\expandafter
1583     \stex_smsmode_do:
1584     \csname @ @ input\endcsname "#1"\relax
1585     %\expandafter \stex_smsmode_do: \l__stex_smsmode_filecontent_tl

```

```

1586 }
1587 \stex_filestack_pop:
1588 }

```

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

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

```

1589 \cs_new_protected:Npn \stex_smsmode_do: {
1590   \stex_if_smsmode:T {
1591     \__stex_smsmode_do:w
1592   }
1593 }
1594 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1595   \int_compare:nNnTF {\tl_count:n{#1}} > 1 { % todo: optimize
1596     \__stex_smsmode_do:w % #1
1597   }{
1598     \expandafter\if\expandafter\relax\noexpand#1
1599     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1600     \else\expandafter\__stex_smsmode_do:w\fi
1601   }
1602 }
1603 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1604   \cs_if_eq:NnF #1 \q__stex_smsmode_break {
1605     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1606       #1\__stex_smsmode_do:w
1607     }{
1608       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1609         #1
1610       }{
1611         \cs_if_eq:NnTF \begin #1 {
1612           \__stex_smsmode_check_begin:n
1613         }{
1614           \cs_if_eq:NnTF \end #1 {
1615             \__stex_smsmode_check_end:n
1616           }{
1617             \__stex_smsmode_do:w
1618           }
1619         }
1620       }
1621     }
1622   }
1623 }
1624
1625 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1626   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1627     \begin{#1}
1628   }{
1629     \__stex_smsmode_do:w
1630   }
1631 }
1632 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1633   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1634     \end{#1}\__stex_smsmode_do:w

```

```

1635   }{
1636     \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1637   }
1638 }
1639
1640 \cs_new_protected:Nn \stex_smsmode_do_ii: {
1641   \stex_if_smsmode:T {
1642     \peek_analysis_map_inline:n {
1643       \int_compare:nNtT ##3 = 0 {
1644         \exp_args:Nx \cs_if_eq:NNTF {##1} \q_stex_smsmode_break {
1645           \peek_analysis_map_break:
1646         }{
1647           \tl_if_in:NxTF \g_stex_smsmode_allowedmacros_tl {##1} {
1648             \peek_analysis_map_break:n{ ##1 \stex_smsmode_do:w }
1649           }{
1650             \tl_if_in:NxTF \g_stex_smsmode_allowedmacros_escape_tl {##1} {
1651               \peek_analysis_map_break:n{ ##1 }
1652             }{
1653               \exp_args:Nx \cs_if_eq:NNTF {##1} \begin {
1654                 \__stex_smsmode_check_begin:n
1655               }{
1656                 \exp_args:Nx \cs_if_eq:NNT {##1} \end {
1657                   \__stex_smsmode_check_end:n
1658                 }
1659               }
1660             }
1661           }
1662         }
1663       }
1664     }
1665   }
1666 }
1667 \cs_new_protected:Nn \__stex_smsmode_check_begin_ii:n {
1668   \seq_if_in:NnTF \g_stex_smsmode_allowedenvs_seq { #1 }{
1669     \peek_analysis_map_break:n{ \begin{#1} }
1670   }{
1671     \stex_smsmode_do:w
1672   }
1673 }
1674 \cs_new_protected:Nn \__stex_smsmode_check_end_ii:n {
1675   \seq_if_in:NnTF \g_stex_smsmode_allowedenvs_seq { #1 }{
1676     \peek_analysis_map_break:n{
1677       \end{#1}
1678     }
1679   }
1680 }
1681 }

```

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

## 29.2 Inheritance

```

1682 <@@=stex_importmodule>

```

`\stex_import_module_uri:nn`

```

1683 \cs_new_protected:Nn \stex_import_module_uri:nn {
1684   \str_set:Nx \l_stex_import_archive_str { #1 }
1685   \str_set:Nn \l_stex_import_path_str { #2 }
1686
1687   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1688   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1689   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1690
1691   \stex_modules_current_namespace:
1692   \bool_lazy_all:nTF {
1693     {\str_if_empty_p:N \l_stex_import_archive_str}
1694     {\str_if_empty_p:N \l_stex_import_path_str}
1695     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1696   }{
1697     \str_set_eq:NN \l_stex_import_path_str \l_stex_modules_subpath_str
1698     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1699   }{
1700     \str_if_empty:NT \l_stex_import_archive_str {
1701       \prop_if_exist:NT \l_stex_current_repository_prop {
1702         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1703       }
1704     }
1705     \str_if_empty:NTF \l_stex_import_archive_str {
1706       \str_if_empty:NF \l_stex_import_path_str {
1707         \str_set:Nx \l_stex_import_ns_str {
1708           \l_stex_module_ns_str / \l_stex_import_path_str
1709         }
1710       }
1711     }{
1712       \stex_require_repository:n \l_stex_import_archive_str
1713       \prop_get:cnN { c_stex_mathhub\_l_stex_import_archive_str\_manifest_prop } { ns }
1714       \l_stex_import_ns_str
1715       \str_if_empty:NF \l_stex_import_path_str {
1716         \str_set:Nx \l_stex_import_ns_str {
1717           \l_stex_import_ns_str / \l_stex_import_path_str
1718         }
1719       }
1720     }
1721   }
1722 }

```

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

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

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

<code>\stex_import_require_module:nmmn</code>	<code>{\langle ns \rangle} {\langle archive-ID \rangle} {\langle path \rangle} {\langle name \rangle}</code>
1727	<code>\cs_new_protected:Nn \stex_import_require_module:nmmn {</code>

```

1728 \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1729
1730   % archive
1731   \str_set:Nx \l_tmpa_str { #2 }
1732   \str_if_empty:NTF \l_tmpa_str {
1733     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1734   } {
1735     \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1736     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1737     \seq_put_right:Nn \l_tmpa_seq { source }
1738   }
1739
1740   % path
1741   \str_set:Nx \l_tmpb_str { #3 }
1742   \str_if_empty:NTF \l_tmpb_str {
1743     \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1744
1745     \ltx@ifpackageloaded{babel} {
1746       \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1747         { \languagename } \l_tmpb_str {
1748         \msg_error:nnx{stex}{error/unknownlanguage}{\languagename}
1749       }
1750     } {
1751       \str_clear:N \l_tmpb_str
1752     }
1753
1754     \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1755     \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1756       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1757     }{
1758       \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1759       \IfFileExists{ \l_tmpa_str.tex }{
1760         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1761       }{
1762         % try english as default
1763         \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1764         \IfFileExists{ \l_tmpa_str.en.tex }{
1765           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1766         }{
1767           \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1768         }
1769       }
1770     }
1771
1772   } {
1773     \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1774     \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1775
1776     \ltx@ifpackageloaded{babel} {
1777       \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1778         { \languagename } \l_tmpb_str {
1779       \msg_error:nnx{stex}{error/unknownlanguage}{\languagename}
1780     }
1781   } {

```



```

1782     \str_clear:N \l_tmpb_str
1783   }
1784
1785   \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1786
1787   \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1788   \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1789     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1790   }{
1791     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1792     \IfFileExists{ \l_tmpa_str/#4.tex }{
1793       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1794     }{
1795       % try english as default
1796       \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1797       \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1798         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1799       }{
1800         \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1801         \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1802           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1803         }{
1804           \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1805           \IfFileExists{ \l_tmpa_str.tex }{
1806             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1807           }{
1808             % try english as default
1809             \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1810             \IfFileExists{ \l_tmpa_str.en.tex }{
1811               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1812             }{
1813               \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1814             }
1815           }
1816         }
1817       }
1818     }
1819   }
1820 }
1821
1822 \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
1823   \seq_clear:N \l_stex_all_modules_seq
1824   \str_clear:N \l_stex_current_module_str
1825   \str_set:Nx \l_tmpb_str { #2 }
1826   \str_if_empty:NF \l_tmpb_str {
1827     \stex_set_current_repository:n { #2 }
1828   }
1829   \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1830 }
1831
1832 \stex_if_module_exists:nF { #1 ? #4 } {
1833   \msg_error:nnx{stex}{error/unknownmodule}{
1834     #1?#4~(in~file~\g__stex_importmodule_file_str)
1835   }

```

```

1836     }
1837   }
1838   \stex_activate_module:n { #1 ? #4 }
1839 }

```

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

## `\importmodule`

```

1840 \NewDocumentCommand \importmodule { 0{} m } {
1841   \stex_import_module_uri:nn { #1 } { #2 }
1842   \stex_debug:nn{modules}{Importing~module:~
1843     \l_stex_import_ns_str ? \l_stex_import_name_str
1844   }
1845   \stex_if_smsmode:F {
1846     \stex_import_require_module:nnnn
1847     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1848     { \l_stex_import_path_str } { \l_stex_import_name_str }
1849     \stex_annotate_invisible:nnn
1850     {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1851   }
1852   \exp_args:Nx \stex_add_to_current_module:n {
1853     \stex_import_require_module:nnnn
1854     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1855     { \l_stex_import_path_str } { \l_stex_import_name_str }
1856   }
1857   \exp_args:Nx \stex_add_import_to_current_module:n {
1858     \l_stex_import_ns_str ? \l_stex_import_name_str
1859   }
1860   \stex_smsmode_do:
1861 }
1862 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

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

## `\usemodule`

```

1863 \NewDocumentCommand \usemodule { 0{} m } {
1864   \stex_if_smsmode:F {
1865     \stex_import_module_uri:nn { #1 } { #2 }
1866     \stex_import_require_module:nnnn
1867     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1868     { \l_stex_import_path_str } { \l_stex_import_name_str }
1869     \stex_annotate_invisible:nnn
1870     {usemodule} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1871   }
1872   \stex_smsmode_do:
1873 }

```

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

```

1874 \</package>

```

## Chapter 30

# STEX -Symbols Implementation

```
1875 <*package>
1876
1877 %%%%%%%%%% symbols.dtx %%%%%%%%%%
1878
Warnings and error messages
1879
```

### 30.1 Symbol Declarations

```
1880 <@@=stex_symdecl>
\l_stex_all_symbols_seq Stores all available symbols
1881 \seq_new:N \l_stex_all_symbols_seq
(End definition for \l_stex_all_symbols_seq. This variable is documented on page 36.)

\STEXsymbol
1882 \NewDocumentCommand \STEXsymbol { m } {
1883   \stex_get_symbol:n { #1 }
1884   \exp_args:No
1885   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1886 }
(End definition for \STEXsymbol. This function is documented on page 38.)
symdecl arguments:
1887 \keys_define:nn { stex / symdecl } {
1888   name      .str_set:N = \l_stex_symdecl_name_str ,
1889   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1890   args      .str_set:N = \l_stex_symdecl_args_str ,
1891   type      .tl_set:N   = \l_stex_symdecl_type_tl ,
1892   align     .str_set:N   = \l_stex_symdecl_align_str , % TODO(?)
1893   gfc       .str_set:N   = \l_stex_symdecl_gfc_str , % TODO(?)
1894   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
1895   def       .tl_set:N   = \l_stex_symdecl_definiens_tl
1896 }
```

```

1897
1898 \bool_new:N \l_stex_symdecl_make_macro_bool
1899
1900 \cs_new_protected:Nn \__stex_symdecl_args:n {
1901   \str_clear:N \l_stex_symdecl_name_str
1902   \str_clear:N \l_stex_symdecl_args_str
1903   \bool_set_false:N \l_stex_symdecl_local_bool
1904   \tl_clear:N \l_stex_symdecl_type_tl
1905   \tl_clear:N \l_stex_symdecl_definiens_tl
1906
1907   \keys_set:nn { stex / symdecl } { #1 }
1908 }

```

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

```

1909
1910 \NewDocumentCommand \symdecl { s O{} m } {
1911   \__stex_symdecl_args:n { #2 }
1912   \IfBooleanTF #1 {
1913     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1914   } {
1915     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1916   }
1917   \stex_symdecl_do:n { #3 }
1918   \stex_smsmode_do:
1919 }
1920 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

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

```

1921 \cs_new_protected:Nn \stex_symdecl_do:n {
1922   \stex_if_in_module:F {
1923     % TODO throw error? some default namespace?
1924   }
1925
1926   \str_if_empty:NT \l_stex_symdecl_name_str {
1927     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1928   }
1929
1930   \prop_if_exist:cT { l_stex_symdecl_
1931     \l_stex_current_module_str ?
1932     \l_stex_symdecl_name_str
1933     _prop
1934   }{
1935     % TODO throw error (beware of circular dependencies)
1936   }
1937
1938   \prop_clear:N \l_tmpa_prop
1939   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
1940   \seq_clear:N \l_tmpa_seq
1941   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1942   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1943

```

```

1944 \exp_args:No \stex_add_constant_to_current_module:n {
1945   \l_stex_symdecl_name_str
1946 }
1947
1948 % arity/args
1949 \int_zero:N \l_tmpb_int
1950
1951 \bool_set_true:N \l_tmpa_bool
1952 \str_map_inline:Nn \l_stex_symdecl_args_str {
1953   \token_case_meaning:NnF ##1 {
1954     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1955     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1956     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1957     {\tl_to_str:n a} {
1958       \bool_set_false:N \l_tmpa_bool
1959       \int_incr:N \l_tmpb_int
1960     }
1961     {\tl_to_str:n B} {
1962       \bool_set_false:N \l_tmpa_bool
1963       \int_incr:N \l_tmpb_int
1964     }
1965   }{
1966     \msg_set:nnn{stex}{error/wrongargs}{
1967       args~value~in~symbol~declaration~for~
1968       \l_stex_current_module_str ?
1969       \l_stex_symdecl_name_str ~
1970       needs~to~be~
1971       i,~a,~b~or~B,~but~##1~given
1972     }
1973     \msg_error:nn{stex}{error/wrongargs}
1974   }
1975 }
1976 \bool_if:NTF \l_tmpa_bool {
1977   % possibly numeric
1978   \str_if_empty:NTF \l_stex_symdecl_args_str {
1979     \prop_put:Nnn \l_tmpa_prop { args } {}
1980     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1981   }{
1982     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1983     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1984     \str_clear:N \l_tmpa_str
1985     \int_step_inline:nn \l_tmpa_int {
1986       \str_put_right:Nn \l_tmpa_str i
1987     }
1988     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1989   }
1990 } {
1991   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1992   \prop_put:Nnx \l_tmpa_prop { arity }
1993     { \str_count:N \l_stex_symdecl_args_str }
1994 }
1995 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
1996
1997

```

```

1998 % semantic macro
1999
2000 \bool_if:NT \l_stex_symdecl_make_macro_bool {
2001   \exp_args:Nx \stex_do_aftergroup:n {
2002     \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2003       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2004     }}
2005   }
2006
2007   \bool_if:NF \l_stex_symdecl_local_bool {
2008     \exp_args:Nx \stex_add_to_current_module:n {
2009       \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2010         \l_stex_current_module_str ? \l_stex_symdecl_name_str
2011       } }
2012     }
2013   }
2014 }
2015
2016 % add to all symbols
2017
2018 \bool_if:NF \l_stex_symdecl_local_bool {
2019   \exp_args:Nx \stex_add_to_current_module:n {
2020     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2021       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2022     }
2023   }
2024 %   \exp_args:Nx \stex_add_field_to_current_module:n {
2025 %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2026 %   }
2027 }
2028
2029 \stex_debug:nn{symbols}{New~symbol:~
2030   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2031   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2032   Args:~\prop_item:Nn \l_tmpa_prop { args }
2033 }
2034
2035 % circular dependencies require this:
2036
2037 \prop_if_exist:cF {
2038   l_stex_symdecl_
2039   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2040   _prop
2041 } {
2042   \prop_set_eq:cN {
2043     l_stex_symdecl_
2044     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2045     _prop
2046   } \l_tmpa_prop
2047 }
2048
2049 \seq_clear:c {
2050   l_stex_symdecl_
2051   \l_stex_current_module_str ? \l_stex_symdecl_name_str

```

```

2052   _notations
2053 }
2054
2055 \bool_if:NF \l_stex_symdecl_local_bool {
2056   \exp_args:Nx
2057   \stex_add_to_current_module:n {
2058     \seq_clear:c {
2059       l_stex_symdecl_
2060       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2061       _notations
2062     }
2063     \prop_set_from_keyval:cn {
2064       l_stex_symdecl_
2065       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2066       _prop
2067     } {
2068       name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2069       module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2070       type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2071       args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2072       arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2073       assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
2074     }
2075   }
2076 }
2077
2078 \stex_if_smsmode:TF {
2079   \bool_if:NF \l_stex_symdecl_local_bool {
2080     % \exp_args:Nx \stex_add_to_sms:n {
2081     %   \prop_set_from_keyval:cn {
2082     %     l_stex_symdecl_
2083     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2084     %     _prop
2085     %   } {
2086     %     name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2087     %     module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2088     %     local     = \prop_item:Nn \l_tmpa_prop { local }     ,
2089     %     type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2090     %     args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2091     %     arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2092     %     assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
2093     %   }
2094     %   \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2095     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2096     %   }
2097     % }
2098   }
2099 }{
2100   \exp_args:Nx \stex_do_aftergroup:n {
2101     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2102       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2103     }
2104   }
2105   \stex_if_do_html:T {

```

```

2106 \stex_annotate_invisible:nnn {symdecl} {
2107   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2108 } {
2109   \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}{}}{ $\l_st
2110   \stex_annotate_invisible:nnn{args}{}{
2111     \prop_item:Nn \l_tmpa_prop { args }
2112   }
2113   \stex_annotate_invisible:nnn{macroname}{#1}{}
2114   \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2115     \stex_annotate_invisible:nnn{definiens}{}
2116     { $\l_stex_symdecl_definiens_tl$ }
2117   }
2118 }
2119 }
2120 }
2121 }

```

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

`\stex_get_symbol:n`

```

2122 \str_new:N \l_stex_get_symbol_uri_str
2123
2124 \cs_new_protected:Nn \stex_get_symbol:n {
2125   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2126     \__stex_symdecl_get_symbol_from_cs:n { #1 }
2127   }{
2128     % argument is a string
2129     % is it a command name?
2130     \cs_if_exist:cTF { #1 }{
2131       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2132       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2133       \str_if_empty:NNTF \l_tmpa_str {
2134         \exp_args:Nx \cs_if_eq:NNTF {
2135           \tl_head:N \l_tmpa_tl
2136         } \stex_invoke_symbol:n {
2137           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
2138         }{
2139           \__stex_symdecl_get_symbol_from_string:n { #1 }
2140         }
2141       } {
2142         \__stex_symdecl_get_symbol_from_string:n { #1 }
2143       }
2144     }{
2145       % argument is not a command name
2146       \__stex_symdecl_get_symbol_from_string:n { #1 }
2147       % \l_stex_all_symbols_seq
2148     }
2149   }
2150 }
2151
2152 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2153   \str_set:Nn \l_tmpa_str { #1 }
2154   \bool_set_false:N \l_tmpa_bool
2155   \stex_if_in_module:T {

```



```

2156 \exp_args:Nno \seq_if_in:cnT {c_stex_module_\l_stex_current_module_str _constants} { \l_
2157 \bool_set_true:N \l_tmpa_bool
2158 \str_set:Nx \l_stex_get_symbol_uri_str {
2159 \l_stex_current_module_str ? #1
2160 }
2161 }
2162 }
2163 \bool_if:NF \l_tmpa_bool {
2164 \tl_set:Nn \l_tmpa_tl {
2165 \msg_set:nnn{stex}{error/unknownsymbol}{
2166 No~symbol~#1~found!
2167 }
2168 \msg_error:nn{stex}{error/unknownsymbol}
2169 }
2170 \str_set:Nn \l_tmpa_str { #1 }
2171 \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2172 \seq_map_inline:Nn \l_stex_all_symbols_seq {
2173 \str_set:Nn \l_tmpb_str { ##1 }
2174 \str_if_eq:eeT { \l_tmpa_str } {
2175 \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
2176 } {
2177 \seq_map_break:n {
2178 \tl_set:Nn \l_tmpa_tl {
2179 \str_set:Nn \l_stex_get_symbol_uri_str {
2180 ##1
2181 }
2182 }
2183 }
2184 }
2185 }
2186 \l_tmpa_tl
2187 }
2188 }
2189
2190 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
2191 \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2192 { \tl_tail:N \l_tmpa_tl }
2193 \tl_if_single:NTF \l_tmpa_tl {
2194 \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2195 \exp_after:wN \str_set:Nn \exp_after:wN
2196 \l_stex_get_symbol_uri_str \l_tmpa_tl
2197 }{
2198 % TODO
2199 % tail is not a single group
2200 }
2201 }{
2202 % TODO
2203 % tail is not a single group
2204 }
2205 }

```

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

## 30.2 Notations

```

2206 <@@=stex_notation>

      notation arguments:
2207 \keys_define:nn { stex / notation } {
2208   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2209   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2210   prec .str_set_x:N = \l__stex_notation_prec_str ,
2211   op .tl_set:N = \l__stex_notation_op_tl ,
2212   primary .bool_set:N = \l__stex_notation_primary_bool ,
2213   primary .default:n = {true} ,
2214   unknown .code:n = \str_set:Nx
2215     \l__stex_notation_variant_str \l_keys_key_str
2216 }
2217
2218 \cs_new_protected:Nn \stex_notation_args:n {
2219   \str_clear:N \l__stex_notation_lang_str
2220   \str_clear:N \l__stex_notation_variant_str
2221   \str_clear:N \l__stex_notation_prec_str
2222   \tl_clear:N \l__stex_notation_op_tl
2223   \bool_set_false:N \l__stex_notation_primary_bool
2224
2225   \keys_set:nn { stex / notation } { #1 }
2226 }

```

**\notation**

```

2227 \NewDocumentCommand \notation { 0{ } m } {
2228   \stex_notation_args:n { #1 }
2229   \tl_clear:N \l_stex_symdecl_definiens_tl
2230   \stex_get_symbol:n { #2 }
2231   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
2232 }
2233 \stex_deactivate_macro:Nn \notation {module-environments}

```

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

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

```

2234 \cs_new_protected:Nn \stex_notation_do:nn {
2235   \let\l_stex_current_symbol_str\relax
2236   \prop_set_eq:Nc \l_tmpa_prop {
2237     \l_stex_symdecl_ #1 _prop
2238   }
2239
2240   \prop_clear:N \l_tmpb_prop
2241   \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
2242   \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
2243   \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
2244
2245   % precedences
2246   \seq_clear:N \l_tmpb_seq
2247   \exp_args:NNno
2248   \str_if_empty:NTF \l__stex_notation_prec_str {
2249     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2250     \int_compare:nNnTF \l_tmpa_str = 0 {

```

```

2251     \exp_args:NNx
2252     \prop_put:Nno \l_tmpb_prop { opprec }
2253     { \neginfprec }
2254   }{
2255     \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
2256   }
2257 } {
2258   \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2259     \exp_args:NNx
2260     \prop_put:Nno \l_tmpb_prop { opprec }
2261     { \neginfprec }
2262     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2263     \int_step_inline:nn { \l_tmpa_str } {
2264       \exp_args:NNx
2265       \seq_put_right:Nn \l_tmpb_seq { \infprec }
2266     }
2267   }{
2268     \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2269     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2270       \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
2271       \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2272         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2273         \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2274         \seq_map_inline:Nn \l_tmpa_seq {
2275           \seq_put_right:Nn \l_tmpb_seq { ##1 }
2276         }
2277       }
2278       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2279     }{
2280       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2281       \int_compare:nNnTF \l_tmpa_str = 0 {
2282         \exp_args:NNx
2283         \prop_put:Nno \l_tmpb_prop { opprec }
2284         { \infprec }
2285       }{
2286         \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
2287       }
2288     }
2289   }
2290 }
2291
2292 \seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
2293 \int_step_inline:nn { \l_tmpa_str } {
2294   \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
2295     \exp_args:NNx
2296     \seq_put_right:Nn \l_tmpb_seq {
2297       \prop_item:Nn \l_tmpb_prop { opprec }
2298     }
2299   }
2300 }
2301
2302 \prop_put:Nno \l_tmpb_prop { argprec } \l_tmpb_seq
2303 \tl_clear:N \l_tmpa_tl
2304

```

```

2305 \int_compare:nNnTF \l_tmpa_str = 0 {
2306   \exp_args:NNe
2307   \cs_set:Npn \l__stex_notation_macrocode_cs {
2308     \_stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2309     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2310     { \prop_item:Nn \l_tmpb_prop { opprec } }
2311     { \exp_not:n { #2 } }
2312   }
2313   \__stex_notation_final:
2314 }{
2315   \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
2316   \str_if_in:NnTF \l_tmpb_str b {
2317     \exp_args:Nne \use:nn
2318     {
2319       \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2320       \cs_set:Npn \l_tmpa_str } { {
2321         \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2322         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2323         { \prop_item:Nn \l_tmpb_prop { opprec } }
2324         { \exp_not:n { #2 } }
2325       } }
2326   }{
2327     \str_if_in:NnTF \l_tmpb_str B {
2328       \exp_args:Nne \use:nn
2329       {
2330         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2331         \cs_set:Npn \l_tmpa_str } { {
2332           \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2333           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2334           { \prop_item:Nn \l_tmpb_prop { opprec } }
2335           { \exp_not:n { #2 } }
2336         } }
2337     }{
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_oma: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   }
2349
2350   \int_zero:N \l_tmpa_int
2351   \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2352   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2353   \__stex_notation_arguments:
2354 }
2355 }

```

(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

```

2356 \cs_new_protected:Nn \_stex_notation_arguments: {
2357   \int_incr:N \l_tmpa_int
2358   \str_if_empty:NTF \l_tmpa_str {
2359     \_stex_notation_final:
2360   }{
2361     \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2362     \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2363     \str_if_eq:VnTF \l_tmpb_str a {
2364       \_stex_notation_argument_assoc:n
2365     }{
2366       \str_if_eq:VnTF \l_tmpb_str B {
2367         \_stex_notation_argument_assoc:n
2368       }{
2369         \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2370         \tl_put_right:Nx \l_tmpa_tl {
2371           { \_stex_term_math_arg:nnn
2372             { \int_use:N \l_tmpa_int }
2373             { \l_tmpb_str }
2374             { ####\int_use:N \l_tmpa_int }
2375           }
2376         }
2377         \_stex_notation_arguments:
2378       }
2379     }
2380   }
2381 }

```

(End definition for `\_stex_notation_arguments:.`)

`\_stex_notation_argument_assoc:n`

```

2382 \cs_new_protected:Nn \_stex_notation_argument_assoc:n {
2383   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2384   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
2385   \tl_put_right:Nx \l_tmpa_tl {
2386     { \_stex_term_math_assoc_arg:nnnn
2387       { \int_use:N \l_tmpa_int }
2388       { \l_tmpb_str }
2389       \exp_args:No \exp_not:n
2390       {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
2391       { ####\int_use:N \l_tmpa_int }
2392     }
2393   }
2394   \_stex_notation_arguments:
2395 }

```

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

`\_stex_notation_final:` Called after processing all notation arguments

```

2396 \cs_new_protected:Nn \_stex_notation_final: {
2397   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
2398   \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
2399   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2400   \exp_args:Nne \use:nn

```

```

2401 {
2402 \cs_generate_from_arg_count:cNnn {
2403   stex_notation_ \l_tmpa_str \c_hash_str
2404   \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2405   _cs
2406 }
2407 \cs_set:Npn \l_tmpb_str } { {
2408   \exp_after:wN \exp_after:wN \exp_after:wN
2409   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2410   { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2411 } }
2412
2413 \tl_if_empty:NF \l__stex_notation_op_tl {
2414   \cs_set:cpx {
2415     stex_op_notation_ \l_tmpa_str \c_hash_str
2416     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2417     _cs
2418   } {
2419     \_stex_term_oms:nnn {
2420       \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2421       \l__stex_notation_lang_str
2422     }{
2423       \l_tmpa_str
2424     }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2425   }
2426 }
2427
2428 \exp_args:Ne
2429 \stex_add_to_current_module:n {
2430   \cs_generate_from_arg_count:cNnn {
2431     stex_notation_ \l_tmpa_str \c_hash_str
2432     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2433     _cs
2434   } \cs_set:Npn {\l_tmpb_str} {
2435     \exp_after:wN \exp_after:wN \exp_after:wN
2436     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2437     { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2438   }
2439   \tl_if_empty:NF \l__stex_notation_op_tl {
2440     \cs_set:cpn {
2441       stex_op_notation_ \l_tmpa_str \c_hash_str
2442       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2443       _cs
2444     } {
2445       \_stex_term_oms:nnn {
2446         \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2447         \l__stex_notation_lang_str
2448       }{
2449         \l_tmpa_str
2450       }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2451     }
2452   }
2453 }
2454

```

```

2455 \seq_put_right:cx {
2456   l_stex_symdecl_
2457   \prop_item:Nn \l_tmpb_prop { symbol }
2458   _notations
2459 } {
2460   \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2461 }
2462
2463 \stex_debug:nn{symbols}{
2464   Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2465   ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
2466   Operator~precedence:~
2467   \prop_item:Nn \l_tmpb_prop { opprec }^^J
2468   Argument~precedences:~
2469   \seq_use:Nn \l_tmpa_seq {,~}^^J
2470   Notation: \cs_meaning:c {
2471     stex_notation_ \l_tmpa_str \c_hash_str
2472     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2473     _cs
2474   }
2475 }
2476
2477 \prop_set_eq:cN {
2478   l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2479   \c_hash_str \l__stex_notation_lang_str _prop
2480 } \l_tmpb_prop
2481
2482 \exp_args:Ne
2483 \stex_add_to_current_module:n {
2484   \seq_put_right:cn {
2485     l_stex_symdecl_
2486     \prop_item:Nn \l_tmpb_prop { symbol }
2487     _notations
2488   } {
2489     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2490   }
2491   \prop_set_from_keyval:cn {
2492     l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2493     \c_hash_str \l__stex_notation_lang_str _prop
2494   } {
2495     symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2496     language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2497     variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,
2498     opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2499     argprecs    = \prop_item:Nn \l_tmpb_prop { argprecs }    ,
2500   }
2501 }
2502
2503 \stex_if_smsmode:TF {
2504 %   \exp_args:Nx \stex_add_to_sms:n {
2505 %     \prop_set_from_keyval:cn {
2506 %       l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2507 %       \c_hash_str \l__stex_notation_lang_str _prop
2508 %     } {

```

```

2509 %      symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2510 %      language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2511 %      variant      = \prop_item:Nn \l_tmpb_prop { variant }      ,
2512 %      opprec       = \prop_item:Nn \l_tmpb_prop { opprec }       ,
2513 %      argprec      = \prop_item:Nn \l_tmpb_prop { argprec }      ,
2514 %    }
2515 %  }
2516 }{
2517
2518 % HTML annotations
2519 \stex_if_do_html:T {
2520   \stex_annotate_invisible:nnn { notation }
2521   { \prop_item:Nn \l_tmpb_prop { symbol } } {
2522     \stex_annotate_invisible:nnn { notationfragment }
2523     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
2524     \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2525     \stex_annotate_invisible:nnn { precedence }
2526     { \prop_item:Nn \l_tmpb_prop { opprec };
2527       \seq_use:Nn \l_tmpa_seq { x }
2528     }{}
2529
2530     \int_zero:N \l_tmpa_int
2531     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2532     \tl_clear:N \l_tmpa_tl
2533     \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{}{
2534       \int_incr:N \l_tmpa_int
2535       \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2536       \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2537       \str_if_eq:VnTF \l_tmpb_str a {
2538         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2539           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2540           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2541         } }
2542       }{
2543         \str_if_eq:VnTF \l_tmpb_str B {
2544           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2545             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2546             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2547           } }
2548         }{
2549           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2550             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2551           } }
2552         }
2553       }
2554     }
2555     \stex_annotate_invisible:nnn { notationcomp }{}{
2556       \str_set:Nx \l_stex_current_symbol_str {\prop_item:Nn \l_tmpb_prop { symbol }}
2557       $ \exp_args:Nno \use:nn { \use:c {
2558         stex_notation_ \l_stex_current_symbol_str
2559         \c_hash_str \l__stex_notation_variant_str
2560         \c_hash_str \l__stex_notation_lang_str _cs
2561       } } { \l_tmpa_tl } $
2562     }

```



```

2563     }
2564   }
2565 }
2566 \stex_smsmode_do:
2567 }

```

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

\setnotation

```

2568 \keys_define:nn { stex / setnotation } {
2569   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2570   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2571   unknown .code:n = \str_set:Nx
2572     \l__stex_notation_variant_str \l_keys_key_str
2573 }
2574
2575 \cs_new_protected:Nn \_stex_setnotation_args:n {
2576   \str_clear:N \l__stex_notation_lang_str
2577   \str_clear:N \l__stex_notation_variant_str
2578   \keys_set:nn { stex / setnotation } { #1 }
2579 }
2580
2581 \NewDocumentCommand \setnotation {m m} {
2582   \stex_get_symbol:n { #1 }
2583   \_stex_setnotation_args:n { #2 }
2584   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl\_l_stex_get_symbol_uri_str _notations }
2585     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2586     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notation
2587       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2588     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notation
2589       { \c_hash_str }
2590     \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notations
2591       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2592     \exp_args:Nx \stex_add_to_current_module:n {
2593       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notati
2594         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2595       \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notation
2596         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2597       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notati
2598         { \c_hash_str }
2599     }
2600     \stex_debug:nn {notations}{
2601       Setting~default~notation~
2602       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }~for~
2603       \l_stex_get_symbol_uri_str \
2604       \expandafter\meaning\csname
2605       l_stex_symdecl\_l_stex_get_symbol_uri_str _notations\endcsname
2606     }
2607   }{
2608     % todo throw error
2609   }
2610 }
2611

```

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

**\symdef**

```
2612 \keys_define:nn { stex / symdef } {
2613   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2614   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2615   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2616   type      .tl_set:N   = \l_stex_symdecl_type_tl ,
2617   def       .tl_set:N   = \l_stex_symdecl_definiens_tl ,
2618   op        .tl_set:N   = \l__stex_notation_op_tl ,
2619   lang      .str_set_x:N = \l__stex_notation_lang_str ,
2620   variant   .str_set_x:N = \l__stex_notation_variant_str ,
2621   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2622   unknown   .code:n     = \str_set:Nx
2623             \l__stex_notation_variant_str \l_keys_key_str
2624 }
2625
2626 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2627   \str_clear:N \l_stex_symdecl_name_str
2628   \str_clear:N \l_stex_symdecl_args_str
2629   \bool_set_false:N \l_stex_symdecl_local_bool
2630   \tl_clear:N \l_stex_symdecl_type_tl
2631   \tl_clear:N \l_stex_symdecl_definiens_tl
2632   \str_clear:N \l__stex_notation_lang_str
2633   \str_clear:N \l__stex_notation_variant_str
2634   \str_clear:N \l__stex_notation_prec_str
2635   \tl_clear:N \l__stex_notation_op_tl
2636
2637   \keys_set:nn { stex / symdef } { #1 }
2638 }
2639
2640 \NewDocumentCommand \symdef { 0{} m } {
2641   \__stex_notation_symdef_args:n { #1 }
2642   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2643   \stex_symdecl_do:n { #2 }
2644   \exp_args:Nx \stex_notation_do:nn {
2645     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2646   }
2647 }
2648 \stex_deactivate_macro:Nn \symdef {module~environments}
2649
2650 (End definition for \symdef. This function is documented on page 37.)
2649 \endpackage
```

## Chapter 31

# STEX -Terms Implementation

```
2650 <*package>
2651
2652 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2653
2654 <@@=stex_terms>
2655
2656   Warnings and error messages
2657   \msg_new:nnn{stex}{error/nonotation}{
2658     Symbol~#1~invoked,~but~has~no~notation#2!
2659   }
2660   \msg_new:nnn{stex}{error/notationarg}{
2661     Error~in~parsing~notation~#1
2662   }
2663   \msg_new:nnn{stex}{error/noop}{
2664     Symbol~#1~has~no~operator~notation~for~notation~#2
2665   }
2666
```

### 31.1 Symbol Invocations

Arguments:

```
2665 \keys_define:nn { stex / terms } {
2666   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2667   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2668   unknown .code:n = \str_set:Nx
2669     \l__stex_terms_variant_str \l_keys_key_str
2670 }
2671
2672 \cs_new_protected:Nn \__stex_terms_args:n {
2673   \str_clear:N \l__stex_terms_lang_str
2674   \str_clear:N \l__stex_terms_variant_str
2675   \str_clear:N \l__stex_terms_prec_str
2676   \tl_clear:N \l__stex_terms_op_tl
2677
2678   \keys_set:nn { stex / terms } { #1 }
```

2679 }

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

```
2680 \cs_new_protected:Nn \stex_invoke_symbol:n {
2681   \if_mode_math:
2682     \exp_after:wN \__stex_terms_invoke_math:n
2683   \else:
2684     \exp_after:wN \__stex_terms_invoke_text:n
2685   \fi: { #1 }
2686 }
```

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

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

```
2687 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2688   \peek_charcode_remove:NTF ! {
2689     \peek_charcode:NTF [ {
2690       \__stex_terms_invoke_op:nw { #1 }
2691     }{
2692       \peek_charcode_remove:NTF ! {
2693         \peek_charcode:NTF [ {
2694           \__stex_terms_invoke_op_custom:nw
2695         }{
2696           % TODO throw error
2697         }
2698       }{
2699         \__stex_terms_invoke_op:nw { #1 } []
2700       }
2701     }
2702   }{
2703     \peek_charcode_remove:NTF * {
2704       \__stex_terms_invoke_text:n { #1 }
2705     }{
2706       \peek_charcode:NTF [ {
2707         \__stex_terms_invoke_math:nw { #1 }
2708       }{
2709         \__stex_terms_invoke_math:nw { #1 } []
2710       }
2711     }
2712   }
2713 }
```

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

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

```
2714 \cs_new_protected:Npn \__stex_terms_invoke_op_custom:nw #1 [#2] {
2715   \stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
2716     \stex_highlight_term:nn{#1}{#2}
2717   }
2718 }
```

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

\\_stex\_terms\_invoke\_op:nw

```

2719 \cs_new_protected:Npn \_stex_terms_invoke_op:nw #1 [#2] {
2720   \_stex_terms_args:n { #2 }
2721   \cs_if_exist:cTF {
2722     stex_op_notation_ #1 \c_hash_str
2723     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2724   }{
2725     \csname stex_op_notation_ #1 \c_hash_str
2726       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2727     \endcsname
2728   }{
2729     \msg_error:nnxx{stex}{error/noop}{#1}{\l__stex_terms_variant_str \c_hash_str \l__stex_te
2730   }
2731 }

```

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

\\_stex\_terms\_invoke\_math:nw

```

2732 \cs_new_protected:Npn \_stex_terms_invoke_math:nw #1 [#2] {
2733   \_stex_terms_args:n { #2 }
2734   \seq_if_empty:cTF {
2735     l_stex_symdecl_ #1 _notations
2736   } {
2737     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
2738   } {
2739     \seq_if_in:cxTF {
2740       l_stex_symdecl_ #1 _notations
2741     }
2742     { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
2743       \str_set:Nn \l_stex_current_symbol_str { #1 }
2744       \use:c{
2745         stex_notation_ #1 \c_hash_str
2746         \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2747         _cs
2748       }
2749     }{
2750       \str_if_empty:NTF \l__stex_terms_variant_str {
2751         \str_if_empty:NTF \l__stex_terms_lang_str {
2752           \seq_get_left:cN {
2753             l_stex_symdecl_ #1 _notations
2754           } \l_tmpa_str
2755           \str_set:Nn \l_stex_current_symbol_str { #1 }
2756           \use:c{
2757             stex_notation_ #1 \c_hash_str \l_tmpa_str
2758             _cs
2759           }
2760         }{
2761           \msg_error:nnxx{stex}{error/nonotation}{#1}{
2762             ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2763           }
2764         }
2765       }{
2766         \msg_error:nnxx{stex}{error/nonotation}{#1}{
2767           ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str

```

```

2768     }
2769   }
2770 }
2771 }
2772 }

```

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

`\_stex_terms_invoke_text:n`

```

2773 \cs_new_protected:Nn \_stex_terms_invoke_text:n {
2774   \peek_charcode_remove:NTF ! {
2775     \stex_term_custom:nn { #1 } { }
2776   }{
2777     \prop_set_eq:Nc \l_tmpa_prop {
2778       l_stex_symdecl_ #1 _prop
2779     }
2780     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2781     \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2782   }
2783 }

```

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

## 31.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
2784 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2785 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2786 \int_new:N \l__stex_terms_downprec
2787 \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
2788 \tl_set:Nn \l__stex_terms_left_bracket_str (
2789 \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

```

2790 \cs_new_protected:Nn \_stex_terms_maybe_brackets:nn {
2791   \bool_if:NTF \l__stex_terms_brackets_done_bool {
2792     \bool_set_false:N \l__stex_terms_brackets_done_bool
2793     #2
2794   } {
2795     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
2796       \bool_if:NTF \l_stex_inarray_bool { #2 }{
2797         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
2798         \dobrackets { #2 }
2799       }

```

```

2800     }{ #2 }
2801   }
2802 }

```

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

### `\dobrackets`

```

2803 \bool_new:N \l__stex_terms_brackets_done_bool
2804 %\RequirePackage{scalerel}
2805 \cs_new_protected:Npn \dobrackets #1 {
2806   %\ThisStyle{\if D\m@switch
2807   %   \exp_args:Nnx \use:nn
2808   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
2809   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
2810   % \else
2811   \exp_args:Nnx \use:nn
2812   {
2813     \bool_set_true:N \l__stex_terms_brackets_done_bool
2814     \int_set:Nn \l__stex_terms_downprec \infprec
2815     \l__stex_terms_left_bracket_str
2816     #1
2817   }
2818   {
2819     \bool_set_false:N \l__stex_terms_brackets_done_bool
2820     \l__stex_terms_right_bracket_str
2821     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2822   }
2823   %\fi}
2824 }

```

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

### `\withbrackets`

```

2825 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2826   \exp_args:Nnx \use:nn
2827   {
2828     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
2829     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
2830     #3
2831   }
2832   {
2833     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
2834     {\l__stex_terms_left_bracket_str}
2835     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
2836     {\l__stex_terms_right_bracket_str}
2837   }
2838 }

```

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

### `\STEXinvisible`

```

2839 \cs_new_protected:Npn \STEXinvisible #1 {
2840   \stex_annotate_invisible:n { #1 }
2841 }

```

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

OMDoc terms:

`\_stex_term_math_oms:nnnn`

```

2842 \cs_new_protected:Nn \_stex_term_oms:nnn {
2843   \stex_annotate:nnn{ OMID }{ #2 }{
2844     \stex_highlight_term:nn { #1 } { #3 }
2845   }
2846 }
2847
2848 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2849   \__stex_terms_maybe_brackets:nn { #3 }{
2850     \stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2851   }
2852 }

```

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

`\_stex_term_math_oma:nnnn`

```

2853 \cs_new_protected:Nn \_stex_term_oma:nnn {
2854   \stex_annotate:nnn{ OMA }{ #2 }{
2855     \stex_highlight_term:nn { #1 } { #3 }
2856   }
2857 }
2858
2859 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2860   \__stex_terms_maybe_brackets:nn { #3 }{
2861     \stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2862   }
2863 }

```

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

`\_stex_term_math_omb:nnnn`

```

2864 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2865   \stex_annotate:nnn{ OMBIND }{ #2 }{
2866     \stex_highlight_term:nn { #1 } { #3 }
2867   }
2868 }
2869
2870 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2871   \__stex_terms_maybe_brackets:nn { #3 }{
2872     \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2873   }
2874 }

```

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

`\_stex_term_math_arg:nnn`

```

2875 \cs_new_protected:Nn \_stex_term_arg:nn {
2876   \stex_unhighlight_term:n {
2877     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2878   }
2879 }

```



```

2880 \cs_new_protected:Nn \stex_term_math_arg:nnn {
2881   \exp_args:Nnx \use:nn
2882     { \int_set:Nn \l__stex_terms_downprec { #2 }
2883       \stex_term_arg:nn { #1 }{ #3 }
2884     }
2885     { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2886   }

```

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

`\stex_term_math_assoc_arg:nnnn`

```

2887 \cs_new_protected:Nn \stex_term_math_assoc_arg:nnnn {
2888   \clist_set:Nn \l_tmpa_clist{ #4 }
2889   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
2890     \tl_set:Nn \l_tmpa_tl { #4 }
2891   }{
2892     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
2893     \clist_reverse:N \l_tmpa_clist
2894     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
2895
2896     \clist_map_inline:Nn \l_tmpa_clist {
2897       \exp_args:NNNo \exp_args:Nno \tl_set:No \l_tmpa_tl {
2898         \exp_args:Nno
2899           \l_tmpa_cs { ##1 } \l_tmpa_tl
2900       }
2901     }
2902
2903   }
2904   \exp_args:Nnno
2905   \stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2906 }

```

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

`\stex_term_custom:nn`

```

2907 \cs_new_protected:Nn \stex_term_custom:nn {
2908   \str_set:Nn \l__stex_terms_custom_uri { #1 }
2909   \str_set:Nn \l_tmpa_str { #2 }
2910   \tl_clear:N \l_tmpa_tl
2911   \int_zero:N \l_tmpa_int
2912   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2913   \__stex_terms_custom_loop:
2914 }

```

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

`\__stex_terms_custom_loop:`

```

2915 \cs_new_protected:Nn \__stex_terms_custom_loop: {
2916   \bool_set_false:N \l_tmpa_bool
2917   \bool_while_do:nn {
2918     \str_if_eq_p:ee X {
2919       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2920     }
2921   }{
2922     \int_incr:N \l_tmpa_int

```

```

2923 }
2924
2925 \peek_charcode:NTF [ {
2926   % notation/text component
2927   \__stex_terms_custom_component:w
2928 } {
2929   \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2930     % all arguments read => finish
2931     \__stex_terms_custom_final:
2932   } {
2933     % arguments missing
2934     \peek_charcode_remove:NTF * {
2935       % invisible, specific argument position or both
2936       \peek_charcode:NTF [ {
2937         % visible specific argument position
2938         \__stex_terms_custom_arg:wn
2939       } {
2940         % invisible
2941         \peek_charcode_remove:NTF * {
2942           % invisible specific argument position
2943           \__stex_terms_custom_arg_inv:wn
2944         } {
2945           % invisible next argument
2946           \__stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2947         }
2948       }
2949     } {
2950       % next normal argument
2951       \__stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
2952     }
2953   }
2954 }
2955 }

```

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

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

```

2956 \cs_new_protected:Npn \__stex_terms_custom_arg_inv:wn [ #1 ] #2 {
2957   \bool_set_true:N \l_tmpa_bool
2958   \__stex_terms_custom_arg:wn [ #1 ] { #2 }
2959 }

```

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

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

```

2960 \cs_new_protected:Npn \__stex_terms_custom_arg:wn [ #1 ] #2 {
2961   \str_set:Nx \l_tmpb_str {
2962     \str_item:Nn \l_tmpa_str { #1 }
2963   }
2964   \str_case:VnTF \l_tmpb_str {
2965     { X } {
2966       \msg_error:nnx{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2967     }
2968     { i } { \__stex_terms_custom_set_X:n { #1 } }
2969     { b } { \__stex_terms_custom_set_X:n { #1 } }

```

```

2970 { a } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
2971 { B } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
2972 }{}{
2973 \msg_error:nnx{stex}{error/notationarg}{\l\_stex_terms_custom_uri}
2974 }
2975
2976 \bool_if:nTF \l_tmpa_bool {
2977   \tl_put_right:Nx \l_tmpa_tl {
2978     \stex_annotate_invisible:n {
2979       \stex_term_arg:nn { \int_eval:n { #1 } }
2980       \exp_not:n { { #2 } }
2981     }
2982   }
2983 } {
2984   \tl_put_right:Nx \l_tmpa_tl {
2985     \stex_term_arg:nn { \int_eval:n { #1 } }
2986     \exp_not:n { { #2 } }
2987   }
2988 }
2989
2990 \_stex_terms_custom_loop:
2991 }

```

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

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

```

2992 \cs_new_protected:Nn \_stex_terms_custom_set_X:n {
2993   \str_set:Nx \l_tmpa_str {
2994     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
2995     X
2996     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
2997   }
2998 }

```

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

\\_stex\_terms\_custom\_component:

```

2999 \cs_new_protected:Npn \_stex_terms_custom_component:w [ #1 ] {
3000   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
3001   \_stex_terms_custom_loop:
3002 }

```

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

\\_stex\_terms\_custom\_final:

```

3003 \cs_new_protected:Nn \_stex_terms_custom_final: {
3004   \int_compare:nNnTF \l_tmpb_int = 0 {
3005     \exp_args:Nnno \stex_term oms:nnn
3006   }{
3007     \str_if_in:NnTF \l_tmpa_str {b} {
3008       \exp_args:Nnno \stex_term ombind:nnn
3009     } {
3010       \exp_args:Nnno \stex_term oma:nnn
3011     }
3012   }

```

```

3013 { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
3014 }

```

(End definition for `\_stex_terms_custom_final:`.)

`\symref`

`\symname`

```

3015 \NewDocumentCommand \symref { m m }{
3016   \let\compemph_uri_prev:\compemph@uri
3017   \let\compemph@uri\symrefemph@uri
3018   \STEXsymbol{#1}! [#2]
3019   \let\compemph@uri\compemph_uri_prev:
3020 }
3021
3022 \keys_define:nn { stex / symname } {
3023   post      .str_set_x:N    = \l_stex_symname_post_str
3024 }
3025
3026 \cs_new_protected:Nn \stex_symname_args:n {
3027   \str_clear:N \l_stex_symname_post_str
3028   \keys_set:nn { stex / symname } { #1 }
3029 }
3030
3031 \NewDocumentCommand \symname { O{} m }{
3032   \stex_symname_args:n { #1 }
3033   \stex_get_symbol:n { #2 }
3034   \str_set:Nx \l_tmpa_str {
3035     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3036   }
3037   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {-}
3038
3039   \let\compemph_uri_prev:\compemph@uri
3040   \let\compemph@uri\symrefemph@uri
3041   \exp_args:NNx \use:nn
3042   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }! [
3043     \l_tmpa_str \l_stex_symname_post_str
3044   ] }
3045   \let\compemph@uri\compemph_uri_prev:
3046 }

```

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

### 31.3 Notation Components

```

3047 <@@=stex_notationcomps>

```

`\stex_highlight_term:nn`

```

3048
3049 \str_new:N \l_stex_current_symbol_str
3050 \cs_new_protected:Nn \stex_highlight_term:nn {
3051   \exp_args:Nnx
3052   \use:nn {
3053     \str_set:Nx \l_stex_current_symbol_str { #1 }
3054     #2
3055   } {

```

```

3056 \str_set:Nx \exp_not:N \l_stex_current_symbol_str
3057 { \l_stex_current_symbol_str }
3058 }
3059 }
3060
3061 \cs_new_protected:Nn \stex_unhighlight_term:n {
3062 % \latexml_if:TF {
3063 % #1
3064 % } {
3065 % \rustex_if:TF {
3066 % #1
3067 % } {
3068 #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
3069 % }
3070 % }
3071 }

```

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

```

\comp
\compemph@uri 3072 \cs_new_protected:Npn \comp #1 {
\compemph 3073 \str_if_empty:NF \l_stex_current_symbol_str {
\defemph 3074 \rustex_if:TF {
\defemph@uri 3075 \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
\symrefemph 3076 }{
\symrefemph@uri 3077 \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3078 }
3079 }
3080 }
3081
3082 \cs_new_protected:Npn \compemph@uri #1 #2 {
3083 \compemph{ #1 }
3084 }
3085
3086
3087 \cs_new_protected:Npn \compemph #1 {
3088 #1
3089 }
3090
3091 \cs_new_protected:Npn \defemph@uri #1 #2 {
3092 \defemph{#1}
3093 }
3094
3095 \cs_new_protected:Npn \defemph #1 {
3096 \textbf{#1}
3097 }
3098
3099 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3100 \symrefemph{#1}
3101 }
3102
3103 \cs_new_protected:Npn \symrefemph #1 {
3104 \textbf{#1}
3105 }

```

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

## `\ellipses`

```
3106 \NewDocumentCommand \ellipses {} { \ldots }
```

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

```

\parray
\prmatrix 3107 \bool_new:N \l_stex_inarray_bool
\parrayline 3108 \bool_set_false:N \l_stex_inarray_bool
\parraylineh 3109 \NewDocumentCommand \parray { m m } {
\parraycell 3110 \begin{group}
3111 \bool_set_true:N \l_stex_inarray_bool
3112 \begin{array}{#1}
3113 #2
3114 \end{array}
3115 \end{group}
3116 }
3117
3118 \NewDocumentCommand \prmatrix { m } {
3119 \begin{group}
3120 \bool_set_true:N \l_stex_inarray_bool
3121 \begin{matrix}
3122 #1
3123 \end{matrix}
3124 \end{group}
3125 }
3126
3127 \def \maybepline {
3128 \bool_if:NT \l_stex_inarray_bool {\hline}
3129 }
3130
3131 \def \parrayline #1 #2 {
3132 #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3133 }
3134
3135 \def \pmrow #1 { \parrayline{}{ #1 } }
3136
3137 \def \parraylineh #1 #2 {
3138 #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3139 }
3140
3141 \def \parraycell #1 {
3142 #1 \bool_if:NT \l_stex_inarray_bool {&}
3143 }

```

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

```
3144 \end{package}
```

## Chapter 32

# STEX -Structural Features Implementation

```
3145 <*package>
3146
3147 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3148
3149 <@@=stex_features>
3150
3151   Warnings and error messages
3152 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3153   Symbol~#1~can~not~be~assigned~in~copymodule~#2
3154 }
3155 \msg_new:nnn{stex}{error/interpretmodule/noddefinens}{
3156   Symbol~#1~not~assigned~in~interpretmodule~#2
3157 }
```

### 32.1 Imports with modification

```
3157 \cs_new_protected:Nn \stex_get_symbol_in_copymodule:n {
3158   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3159     \__stex_features_get_symbol_from_cs:n { #1 }
3160   }{
3161     % argument is a string
3162     % is it a command name?
3163     \cs_if_exist:cTF { #1 }{
3164       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3165       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3166       \str_if_empty:NNTF \l_tmpa_str {
3167         \exp_args:Nx \cs_if_eq:NNTF {
3168           \tl_head:N \l_tmpa_tl
3169         } \stex_invoke_symbol:n {
3170           \exp_args:No \__stex_features_get_symbol_from_cs:n { \use:c { #1 } }
3171         }{
3172           \__stex_features_get_symbol_from_string:n { #1 }
3173         }
3174       }
3175     }
3176   }
```

```

3173     }
3174   } {
3175     \__stex_features_get_symbol_from_string:n { #1 }
3176   }
3177   ){
3178     % argument is not a command name
3179     \__stex_features_get_symbol_from_string:n { #1 }
3180     % \l_stex_all_symbols_seq
3181   }
3182 }
3183 }
3184
3185 \cs_new_protected:Nn \__stex_features_get_symbol_from_string:n {
3186   \str_set:Nn \l_tmpa_str { #1 }
3187   \bool_set_false:N \l_tmpa_bool
3188   \bool_if:NF \l_tmpa_bool {
3189     \tl_set:Nn \l_tmpa_tl {
3190       \msg_set:nnn{stex}{error/unknownsymbol}{
3191         No~symbol~#1~found!
3192       }
3193       \msg_error:nn{stex}{error/unknownsymbol}
3194     }
3195     \str_set:Nn \l_tmpa_str { #1 }
3196     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3197     \seq_map_inline:Nn \l__stex_features_copymodule_fields_seq {
3198       \str_set:Nn \l_tmpb_str { ##1 }
3199       \str_if_eq:eeT { \l_tmpa_str } {
3200         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3201       } {
3202         \seq_map_break:n {
3203           \tl_set:Nn \l_tmpa_tl {
3204             \str_set:Nn \l_stex_get_symbol_uri_str {
3205               ##1
3206             }
3207             \__stex_features_get_symbol_check:
3208           }
3209         }
3210       }
3211     }
3212     \l_tmpa_tl
3213   }
3214 }
3215
3216 \cs_new_protected:Nn \__stex_features_get_symbol_from_cs:n {
3217   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3218   { \tl_tail:N \l_tmpa_tl }
3219   \tl_if_single:NTF \l_tmpa_tl {
3220     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3221       \exp_after:wN \str_set:Nn \exp_after:wN
3222       \l_stex_get_symbol_uri_str \l_tmpa_tl
3223       \__stex_features_get_symbol_check:
3224     }{
3225       % TODO
3226       % tail is not a single group

```



```

3227     }
3228   }{
3229     % TODO
3230     % tail is not a single group
3231   }
3232 }
3233
3234 \cs_new_protected:Nn \__stex_features_get_symbol_check: {
3235   \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq {?} \l_stex_get_symbol_uri_str
3236   \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 3 {
3237     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
3238     \str_set:Nx \l_tmpa_str {\seq_use:Nn \l_tmpa_seq ?}
3239     \seq_if_in:Nof \l__stex_features_copymodule_modules_seq \l_tmpa_str {
3240       \msg_error:nxxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3241         \l_stex_current_copymodule_name_str\Allowed:~\seq_use:Nn \l__stex_features_copymodule_modules_seq \l_tmpa_str
3242       }
3243     }
3244   }{
3245     \msg_error:nxxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3246       \l_stex_current_copymodule_name_str~(inexplicably)
3247     }
3248   }
3249 }
3250
3251 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
3252   \stex_import_module_uri:nn { #1 } { #2 }
3253   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3254   \stex_import_require_module:nnnn
3255     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3256     { \l_stex_import_path_str } { \l_stex_import_name_str }
3257   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3258   \seq_set_eq:NN \l__stex_features_copymodule_modules_seq \l_stex_collect_imports_seq
3259   \seq_clear:N \l__stex_features_copymodule_fields_seq
3260   \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3261     \seq_map_inline:cn {c_stex_module_###1_constants}{
3262       \exp_args:NNx \seq_put_right:Nn \l__stex_features_copymodule_fields_seq {
3263         ###1 ? #####1
3264       }
3265     }
3266   }
3267   \seq_clear:N \l_tmpa_seq
3268   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3269     name      = \l_stex_current_copymodule_name_str ,
3270     module    = \l_stex_current_module_str ,
3271     from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3272     includes  = \l_tmpa_seq ,
3273     fields    = \l_tmpa_seq
3274   }
3275   \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3276     as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3277   \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_features_copymodule_modules_seq
3278     \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_features_copymodule_fields_seq {,~}
3279   \stex_if_smsmode:F {
3280     \begin{stex_annotate_env} {#4} {

```

```

3281     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3282   }
3283   \stex_annotate_invisible:nnn{from}{\l_stex_import_ns_str ?\l_stex_import_name_str}{ }
3284 }
3285 \bool_set_eq:NN \l__stex_features_oldhtml_bool \l_stex_html_do_output_bool
3286 \bool_set_false:N \l_stex_html_do_output_bool
3287 }
3288 \cs_new_protected:Nn \stex_copymodule_end:n {
3289   \def \l_tmpa_cs ##1 ##2 {#1}
3290   \bool_set_eq:NN \l_stex_html_do_output_bool \l__stex_features_oldhtml_bool
3291   \tl_clear:N \l_tmpa_tl
3292   \prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3293   \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3294     \seq_map_inline:cn {c_stex_module_##1_constants}{\stex_annotate:nnn{assignment} {##1?###
3295       \l_tmpa_cs{##1}{####1}
3296       \str_if_exist:cTF {l__stex_features_copymodule_##1?####1_name_str} {
3297         \tl_put_right:Nx \l_tmpa_tl {
3298           \prop_set_from_keyval:cn {
3299             l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_#
3300           }{
3301             \exp_after:wN \prop_to_keyval:N \csname
3302               l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_#
3303             \endcsname
3304           }
3305           \seq_clear:c {
3306             l_stex_symdecl_
3307             \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_name_s
3308             _notations
3309           }
3310         }
3311         \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_features_copymodule_##1?####1_name
3312         \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{l__stex_features_
3313         \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3314           \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?###
3315           \tl_put_right:Nx \l_tmpa_tl {
3316             \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3317             \stex_invoke_symbol:n {
3318               \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_na
3319             }
3320           }
3321         }
3322       }
3323     }{
3324       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1_prop}
3325       \prop_put:Nnx \l_tmpa_prop { name }{\l_stex_current_copymodule_name_str / ####1 }
3326       \prop_put:Nnx \l_tmpa_prop { module }{\l_stex_current_module_str }
3327       \tl_put_right:Nx \l_tmpa_tl {
3328         \prop_set_from_keyval:cn {
3329           l_stex_symdecl_\l_stex_current_module_str ? \l_stex_current_copymodule_name_str
3330         }{
3331           \prop_to_keyval:N \l_tmpa_prop
3332         }
3333         \seq_clear:c {
3334           l_stex_symdecl_

```

```

3335         \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3336         _notations
3337     }
3338 }
3339 \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3340 \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3341     \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}
3342     \tl_put_right:Nx \l_tmpa_tl {
3343         \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3344             \stex_invoke_symbol:n {
3345                 \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3346             }
3347         }
3348     }
3349 }
3350 }
3351 \tl_if_exist:cT {l__stex_features_copymodule_##1?####1_def_tl}{
3352     \stex_annotate_invisible:nnn{definiens}{\use:c{l__stex_features_copymodule_##1?####1_def_tl}}
3353 }
3354 % todo notations
3355 }}
3356 }
3357 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3358 \tl_put_left:Nx \l_tmpa_tl {
3359     \prop_set_from_keyval:cn {
3360         l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _pro
3361     }{
3362         \prop_to_keyval:N \l_stex_current_copymodule_prop
3363     }
3364 }
3365 \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl
3366 \stex_debug:nn{copymodule}{result:\meaning \l_tmpa_tl}
3367 \exp_args:Nx \stex_do_aftergroup:n {
3368     \exp_args:No \exp_not:n \l_tmpa_tl
3369 }
3370 \stex_if_smsmode:F {
3371     \end{stex_annotate_env}
3372 }
3373 }
3374
3375 \NewDocumentEnvironment {copymodule} { 0{} m m}{
3376     \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ structure }
3377     \stex_deactivate_macro:Nn \symdecl {module~environments}
3378     \stex_deactivate_macro:Nn \symdef {module~environments}
3379     \stex_deactivate_macro:Nn \notation {module~environments}
3380     \stex_reactivate_macro:N \assign
3381     \stex_reactivate_macro:N \renamedec1
3382     \stex_reactivate_macro:N \donotcopy
3383     \stex_smsmode_do:
3384 }{
3385     \stex_copymodule_end:n {}
3386 }
3387
3388 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{

```

```

3389 \stex_copymodule_start:nnnn { #1 } { #2 } { #3 } { realization }
3390 \stex_deactivate_macro:Nn \symdecl {module~environments}
3391 \stex_deactivate_macro:Nn \symdef {module~environments}
3392 \stex_deactivate_macro:Nn \notation {module~environments}
3393 \stex_reactivate_macro:N \assign
3394 \stex_reactivate_macro:N \renamedec1
3395 \stex_reactivate_macro:N \donotcopy
3396 \stex_smsmode_do:
3397 }{
3398 \stex_copymodule_end:n {
3399 \tl_if_exist:cF {
3400 l__stex_features_copymodule_##1?##2_def_tl
3401 }{
3402 \msg_error:nnxx{stex}{error/interpretmodule/nodedefiniens}{
3403 ##1?##2
3404 }{\l_stex_current_copymodule_name_str}
3405 }
3406 }
3407 }
3408
3409 \NewDocumentCommand \donotcopy { 0{} m }{
3410 \stex_import_module_uri:nn { #1 } { #2 }
3411 \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3412 \seq_map_inline:Nn \l_stex_collect_imports_seq {
3413 \seq_remove_all:Nn \l_stex_features_copymodule_modules_seq { ##1 }
3414 \seq_map_inline:cn {c_stex_module_##1_constants}{
3415 \seq_remove_all:Nn \l_stex_features_copymodule_fields_seq { ##1 ? #####1 }
3416 \bool_lazy_any_p:nT {
3417 { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_name_str}}
3418 { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_macroname_str}}
3419 { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_def_tl}}
3420 }{
3421 % TODO throw error
3422 }
3423 }
3424 }
3425
3426 \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
3427 \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_ns_str ?\l_stex_import_name_str }
3428 \prop_put:Nnx \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
3429 }
3430
3431 \NewDocumentCommand \assign { m m }{
3432 \stex_get_symbol_in_copymodule:n {#1}
3433 \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
3434 \tl_set:cn {l__stex_features_copymodule_\l_stex_get_symbol_uri_str_def_tl}{#2}
3435 }
3436
3437 \keys_define:nn { stex / renamedec1 } {
3438 name .str_set_x:N = \l_stex_renamedec1_name_str
3439 }
3440 \cs_new_protected:Nn \__stex_features_renamedec1_args:n {
3441 \str_clear:N \l_stex_renamedec1_name_str
3442

```

```

3443 \keys_set:nn { stex / renamedecl } { #1 }
3444 }
3445
3446 \NewDocumentCommand \renamedecl { 0{} m m}{
3447   \__stex_features_renamedecl_args:n { #1 }
3448   \stex_get_symbol_in_copymodule:n {#2}
3449   \stex_debug:nn{renamedecl}{renaming-{\l_stex_get_symbol_uri_str}-to~#3}
3450   \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
3451   \str_if_empty:NTF \l_stex_renamedecl_name_str {
3452     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3453       \l_stex_get_symbol_uri_str
3454     } }
3455   } {
3456     \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex_r
3457     \stex_debug:nn{renamedecl}{@~\l_stex_current_module_str ? \l_stex_renamedecl_name_str}
3458     \prop_set_eq:cc {l_stex_symdecl_
3459       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
3460     _prop
3461     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
3462     \seq_set_eq:cc {l_stex_symdecl_
3463       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
3464     _notations
3465     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
3466     \prop_put:cnx {l_stex_symdecl_
3467       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
3468     _prop
3469     }{ name }{ \l_stex_renamedecl_name_str }
3470     \prop_put:cnx {l_stex_symdecl_
3471       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
3472     _prop
3473     }{ module }{ \l_stex_current_module_str }
3474     \exp_args:NNx \seq_put_left:Nn \l__stex_features_copymodule_fields_seq {
3475       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
3476     }
3477     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3478       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
3479     } }
3480   }
3481 }
3482 %\NewDocumentCommand \notation_in_copymodules: { 0{} m } {
3483 % \_stex_notation_args:n { #1 }
3484 % \tl_clear:N \l_stex_symdecl_definiens_tl
3485 % \stex_get_symbol_in_copymodule:n { #2 }
3486 % \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
3487 % % todo
3488 %}
3489 \stex_deactivate_macro:Nn \assign {copymodules}
3490 \stex_deactivate_macro:Nn \renamedecl {copymodules}
3491 \stex_deactivate_macro:Nn \donotcopy {copymodules}
3492
3493
3494 \seq_new:N \l_stex_implicit_morphisms_seq
3495 \NewDocumentCommand \implicitmorphism { 0{} m m}{
3496   \stex_import_module_uri:nn { #1 } { #2 }

```

```

3497 \stex_debug:nn{implicits}{
3498   Implicit~morphism:~
3499   \l_stex_module_ns_str ? \l__stex_features_name_str
3500 }
3501 \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
3502   \l_stex_module_ns_str ? \l__stex_features_name_str
3503 }{
3504   \msg_error:nnn{stex}{error/conflictingmodules}{
3505     \l_stex_module_ns_str ? \l__stex_features_name_str
3506   }
3507 }
3508
3509 % TODO
3510
3511
3512
3513 \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
3514   \l_stex_module_ns_str ? \l__stex_features_name_str
3515 }
3516 }
3517

```

## 32.2 The feature environment

structural@feature

```

3518
3519 \NewDocumentEnvironment{structural@feature}{ m m m }{
3520   \stex_if_in_module:F {
3521     \msg_set:nnn{stex}{error/nomodule}{
3522       Structural~Feature~has~to~occur~in~a~module:\\
3523       Feature~#2~of~type~#1\\
3524       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
3525     }
3526     \msg_error:nn{stex}{error/nomodule}
3527   }
3528
3529   \str_set:Nx \l_stex_module_name_str {
3530     \prop_item:Nn \l_stex_current_module_prop
3531     { name } / #2 - feature
3532   }
3533
3534   \str_set:Nx \l_stex_module_ns_str {
3535     \prop_item:Nn \l_stex_current_module_prop
3536     { ns }
3537   }
3538
3539
3540   \str_clear:N \l_tmpa_str
3541   \seq_clear:N \l_tmpa_seq
3542   \tl_clear:N \l_tmpa_tl
3543   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
3544     origname = #2,
3545     name      = \l_stex_module_name_str ,
3546     ns        = \l_stex_module_ns_str ,

```

```

3547 imports = \exp_not:o { \l_tmpa_seq } ,
3548 constants = \exp_not:o { \l_tmpa_seq } ,
3549 content = \exp_not:o { \l_tmpa_tl } ,
3550 file = \exp_not:o { \g_stex_currentfile_seq } ,
3551 lang = \l_stex_module_lang_str ,
3552 sig = \l_tmpa_str ,
3553 meta = \l_tmpa_str ,
3554 feature = #1 ,
3555 }
3556
3557 \stex_if_smsmode:F {
3558   \begin{stex_annotate_env}{ feature:#1 }{}
3559   \stex_annotate_invisible:nnn{header}{}{ #3 }
3560 }
3561 {}{
3562   \str_set:Nx \l_tmpa_str {
3563     c_stex_feature_
3564     \prop_item:Nn \l_stex_current_module_prop { ns } ?
3565     \prop_item:Nn \l_stex_current_module_prop { name }
3566     _prop
3567   }
3568   \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
3569   \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
3570   \stex_if_smsmode:TF {
3571     \exp_args:Nx \stex_add_to_sms:n {
3572       \prop_gset_from_keyval:cn {
3573         c_stex_feature_
3574         \prop_item:Nn \l_stex_current_module_prop { ns } ?
3575         \prop_item:Nn \l_stex_current_module_prop { name }
3576         _prop
3577       } {
3578         origname = #2,
3579         name = \prop_item:cn { \l_tmpa_str } { name } ,
3580         ns = \prop_item:cn { \l_tmpa_str } { ns } ,
3581         imports = \prop_item:cn { \l_tmpa_str } { imports } ,
3582         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
3583         content = \prop_item:cn { \l_tmpa_str } { content } ,
3584         file = \prop_item:cn { \l_tmpa_str } { file } ,
3585         lang = \prop_item:cn { \l_tmpa_str } { lang } ,
3586         sig = \prop_item:cn { \l_tmpa_str } { sig } ,
3587         meta = \prop_item:cn { \l_tmpa_str } { meta } ,
3588         feature = \prop_item:cn { \l_tmpa_str } { feature }
3589       }
3590     }
3591   } {
3592     \end{stex_annotate_env}
3593   }
3594 }
3595

```

## 32.3 Features

structure

```

3596
3597 \prop_new:N \l_stex_all_structures_prop
3598
3599 \keys_define:nn { stex / features / structure } {
3600   name          .str_set_x:N = \l__stex_features_structure_name_str ,
3601 }
3602
3603 \cs_new_protected:Nn \__stex_features_structure_args:n {
3604   \str_clear:N \l__stex_features_structure_name_str
3605   \keys_set:nn { stex / features / structure } { #1 }
3606 }
3607
3608 %\stex_new_feature:nnnn { structure } { 0{} m } {
3609   % \__stex_features_structure_args:n { ##1 }
3610   % \str_if_empty:NT \l__stex_features_structure_name_str {
3611     % \str_set:Nx \l__stex_features_structure_name_str { ##2 }
3612   % }
3613   %} {
3614   %
3615   %}
3616
3617 \NewDocumentEnvironment{mathstructure}{ 0{} m }{
3618   \__stex_features_structure_args:n { #1 }
3619   \str_if_empty:NT \l__stex_features_structure_name_str {
3620     \str_set:Nx \l__stex_features_structure_name_str { #2 }
3621   }
3622   \exp_args:Nnnx
3623   \begin{structural@feature}{ structure }
3624     { \l__stex_features_structure_name_str }{}
3625     \seq_clear:N \l_tmpa_seq
3626     \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
3627     \stex_smsmode_do:
3628   }{
3629     \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
3630     \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
3631     \str_set:Nx \l_tmpa_str {
3632       \prop_item:Nn \l_stex_current_module_prop { ns } ?
3633       \prop_item:Nn \l_stex_current_module_prop { name }
3634     }
3635     \seq_map_inline:Nn \l_tmpa_seq {
3636       \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
3637     }
3638     \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
3639     \exp_args:Nnx
3640     \AddToHookNext { env / mathstructure / after }{
3641       \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}}{
3642         \stex_term_math_oms:nnnn { \l_tmpa_str }{}{0}{}
3643       }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
3644     \STEXexport {
3645       \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3646         {\prop_item:Nn \l_stex_current_module_prop { origname }}
3647         {\l_tmpa_str}
3648       \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3649         {#2}{\l_tmpa_str}

```



```

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

```

\instantiate

```

3665 \seq_new:N \l__stex_features_structure_field_seq
3666 \str_new:N \l__stex_features_structure_field_str
3667 \str_new:N \l__stex_features_structure_def_tl
3668 \prop_new:N \l__stex_features_structure_prop
3669 \NewDocumentCommand \instantiate { m O{} m }{
3670   \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
3671   \prop_set_eq:Nc \l__stex_features_structure_prop {
3672     c_stex_feature_\l_tmpa_str _prop
3673   }
3674   \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
3675   \seq_map_inline:Nn \l__stex_features_structure_field_seq {
3676     \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
3677     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3678       \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
3679       \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
3680         {!} \l_tmpa_tl
3681       \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3682         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
3683         \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3684         \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3685       }{
3686         \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
3687         \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3688         \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
3689           \l_tmpa_tl
3690         \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3691           \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
3692           \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3693         }{
3694           \tl_clear:N \l_tmpb_tl
3695         }
3696       }
3697     }{
3698       \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
3699       \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3700         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
3701         \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl

```

```

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

```

```

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

```

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

\stex\_invoke\_structure:nnn

```

3807 % #1: URI of the instance
3808 % #2: URI of the instantiated module
3809 \cs_new_protected:Nn \stex_invoke_structure:nnn {
3810   \tl_if_empty:nTF{ #3 }{
3811     \prop_set_eq:Nc \l__stex_features_structure_prop {
3812       c_stex_feature_ #2 _prop
3813     }
3814     \tl_clear:N \l_tmpa_tl
3815     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3816     \seq_map_inline:Nn \l_tmpa_seq {
3817       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3818       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3819       \cs_if_exist:cT {
3820         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
3821       }{
3822         \tl_if_empty:NF \l_tmpa_tl {
3823           \tl_put_right:Nn \l_tmpa_tl {,}
3824         }
3825         \tl_put_right:Nx \l_tmpa_tl {
3826           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
3827         }
3828       }
3829     }
3830     \exp_args:No \mathstrut \l_tmpa_tl
3831   }{
3832     \stex_invoke_symbol:n{#1/#3}
3833   }
3834 }

```

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

3835 </package>

## Chapter 33

# STEX -Statements Implementation

```
3836 <*package>
3837
3838 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3839
3840 \protected\def\ignorespacesandpars{
3841   \begingroup\catcode13=10\relax
3842   \@ifnextchar\par{
3843     \endgroup\expandafter\ignorespacesandpars\@gobble
3844   }{
3845     \endgroup
3846   }
3847 }
3848
3849 <@@=stex_statements>
3850
3851 Warnings and error messages
```

\titleemph

```
3851 \def\titleemph#1{\textbf{#1}}
```

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

### 33.1 Definitions

definiendum

```
3852 \keys_define:nn {stex / definiendum }{
3853   post      .tl_set:N      = \l__stex_statements_definiendum_post_tl,
3854   root      .str_set_x:N    = \l__stex_statements_definiendum_root_str,
3855   gfa       .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
3856 }
3857 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
3858   \str_clear:N \l__stex_statements_definiendum_root_str
3859   \tl_clear:N \l__stex_statements_definiendum_post_tl
3860   \str_clear:N \l__stex_statements_definiendum_gfa_str
}
```

```

3861 \keys_set:nn { stex / definiendum }{ #1 }
3862 }
3863 \NewDocumentCommand \definiendum { 0{} m m } {
3864   \__stex_statements_definiendum_args:n { #1 }
3865   \stex_get_symbol:n { #2 }
3866   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3867   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
3868     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
3869       \tl_set:Nn \l_tmpa_tl { #3 }
3870     } {
3871       \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
3872       \tl_set:Nn \l_tmpa_tl {
3873         \l__stex_statements_definiendum_root_str\l__stex_statements_definiendum_post_tl
3874       }
3875     }
3876   } {
3877     \tl_set:Nn \l_tmpa_tl { #3 }
3878   }
3879
3880   % TODO root
3881   \rustex_if:TF {
3882     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
3883   } {
3884     \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
3885   }
3886 }
3887 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

#### definame

```

3888
3889 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3890
3891 \NewDocumentCommand \definame { 0{} m } {
3892   \__stex_statements_definiendum_args:n { #1 }
3893   % TODO: root
3894   \stex_get_symbol:n { #2 }
3895   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3896   \str_set:Nx \l_tmpa_str {
3897     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3898   }
3899   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3900   \rustex_if:TF {
3901     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3902       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3903     }
3904   } {
3905     \defemph@uri {
3906       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3907     } { \l_stex_get_symbol_uri_str }
3908   }
3909 }
3910 \stex_deactivate_macro:Nn \definame {definition~environments}

```

```

3911
3912 \NewDocumentCommand \Definame { 0{ } m } {
3913   \_stex_statements_definiendum_args:n { #1 }
3914   \stex_get_symbol:n { #2 }
3915   \str_set:Nx \l_tmpa_str {
3916     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3917   }
3918   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3919   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3920   \rustex_if:TF {
3921     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3922       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3923     }
3924   } {
3925     \defemph@uri {
3926       \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
3927     } { \l_stex_get_symbol_uri_str }
3928   }
3929 }
3930 \stex_deactivate_macro:Nn \Definame {definition-environments}
3931
3932 \NewDocumentCommand \Symname { 0{ } m }{
3933   \stex_symname_args:n { #1 }
3934   \stex_get_symbol:n { #2 }
3935   \str_set:Nx \l_tmpa_str {
3936     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3937   }
3938   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3939   \let\compemph_uri_prev:\compemph@uri
3940   \let\compemph@uri\symrefemph@uri
3941   \exp_args:NNx \use:nn
3942   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }! [
3943     \exp_after:wN \stex_capitalize:n \l_tmpa_str
3944     \l_stex_symname_post_str
3945   ] }
3946   \let\compemph@uri\compemph_uri_prev:
3947 }

```

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

#### sdefinition

```

3948
3949 \keys_define:nn {stex / sdefinition }{
3950   type      .str_set_x:N = \sdefinitiontype,
3951   id        .str_set_x:N = \sdefinitionid,
3952   name      .str_set_x:N = \sdefinitionname,
3953   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
3954   title     .tl_set:N     = \sdefinitiontitle
3955 }
3956 \cs_new_protected:Nn \_stex_statements_sdefinition_args:n {
3957   \str_clear:N \sdefinitiontype
3958   \str_clear:N \sdefinitionid
3959   \str_clear:N \sdefinitionname
3960   \clist_clear:N \l__stex_statements_sdefinition_for_clist

```

```

3961 \tl_clear:N \sdefinitiontitle
3962 \keys_set:nn { stex / sdefinition }{ #1 }
3963 }
3964
3965 \NewDocumentEnvironment{sdefinition}{0{}}{
3966   \__stex_statements_sdefinition_args:n{ #1 }
3967   \stex_reactivate_macro:N \definiendum
3968   \stex_reactivate_macro:N \definame
3969   \stex_reactivate_macro:N \Definame
3970   \stex_if_smsmode:F{
3971     \seq_clear:N \l_tmpa_seq
3972     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
3973       \str_if_eq:nnF{ ##1 }{ }{
3974         \stex_get_symbol:n { ##1 }
3975         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
3976           \l_stex_get_symbol_uri_str
3977         }
3978       }
3979     }
3980     \exp_args:Nnnx
3981     \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {,}}
3982     \str_if_empty:NF \sdefinitiontype {
3983       \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{ }
3984     }
3985     \clist_set:Nn \l_tmpa_clist \sdefinitiontype
3986     \tl_clear:N \l_tmpa_tl
3987     \clist_map_inline:Nn \l_tmpa_clist {
3988       \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
3989         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
3990       }
3991     }
3992     \tl_if_empty:NTF \l_tmpa_tl {
3993       \__stex_statements_sdefinition_start:
3994     }{
3995       \l_tmpa_tl
3996     }
3997   }
3998   \stex_ref_new_doc_target:n \sdefinitionid
3999   \stex_smsmode_do:
4000 }{
4001   \str_if_empty:NF \sdefinitionname { \symdecl*{\sdefinitionname} }
4002   \stex_if_smsmode:F {
4003     \clist_set:Nn \l_tmpa_clist \sdefinitiontype
4004     \tl_clear:N \l_tmpa_tl
4005     \clist_map_inline:Nn \l_tmpa_clist {
4006       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
4007         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
4008       }
4009     }
4010     \tl_if_empty:NTF \l_tmpa_tl {
4011       \__stex_statements_sdefinition_end:
4012     }{
4013       \l_tmpa_tl
4014     }

```



```

4015     \end{stex_annotate_env}
4016   }
4017 }

```

\stexpatchdefinition

```

4018 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
4019   \par\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
4020     ~(\sdefinitiontitle)
4021   }~}
4022 }
4023 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\par\medskip}
4024
4025 \newcommand\stexpatchdefinition[3] [] {
4026   \str_set:Nx \l_tmpa_str{ #1 }
4027   \str_if_empty:NTF \l_tmpa_str {
4028     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
4029     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
4030   }{
4031     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
4032     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
4033   }
4034 }

```

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

\inlinedef inline:

```

4035 \keys_define:nn {stex / inlinedef }{
4036   type      .str_set_x:N = \sdefinitiontype,
4037   id        .str_set_x:N = \sdefinitionid,
4038   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4039   name      .str_set_x:N = \sdefinitionname
4040 }
4041 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
4042   \str_clear:N \sdefinitiontype
4043   \str_clear:N \sdefinitionid
4044   \str_clear:N \sdefinitionname
4045   \clist_clear:N \l__stex_statements_sdefinition_for_clist
4046   \keys_set:nn { stex / inlinedef }{ #1 }
4047 }
4048 \NewDocumentCommand \inlinedef { 0{} m } {
4049   \begingroup
4050   \__stex_statements_inlinedef_args:n{ #1 }
4051   \stex_ref_new_doc_target:n \sdefinitionid
4052   \stex_reactivate_macro:N \definiendum
4053   \stex_reactivate_macro:N \definame
4054   \stex_reactivate_macro:N \Definame
4055   \stex_if_smsmode:TF{
4056     \str_if_empty:NF \sdefinitionname { \symdecl*{\sdefinitionname} }
4057   }{
4058     \seq_clear:N \l_tmpa_seq
4059     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4060       \str_if_eq:nnF{ ##1 }{}{
4061         \stex_get_symbol:n { ##1 }
4062         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {

```

```

4063         \l_stex_get_symbol_uri_str
4064     }
4065 }
4066 }
4067 \exp_args:Nnx
4068 \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
4069     \str_if_empty:NF \sdefinitiontype {
4070         \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{ }
4071     }
4072     #2
4073     \str_if_empty:NF \sdefinitionname { \symdecl*{\sdefinitionname} }
4074 }
4075 }
4076 \endgroup
4077 \stex_smsmode_do:
4078 }

```

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

## 33.2 Assertions

**sassertion**

```

4079 \keys_define:nn {stex / sassertion }{
4080     type      .str_set_x:N = \sassertiontype,
4081     id        .str_set_x:N = \sassertionid,
4082     title     .tl_set:N    = \sassertiontitle ,
4083     for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4084     name      .str_set_x:N = \sassertionname
4085 }
4086 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
4087     \str_clear:N \sassertiontype
4088     \str_clear:N \sassertionid
4089     \str_clear:N \sassertionname
4090     \clist_clear:N \l__stex_statements_sassertion_for_clist
4091     \tl_clear:N \sassertiontitle
4092     \keys_set:nn { stex / sassertion }{ #1 }
4093 }
4094
4095
4096 %\tl_new:N \g__stex_statements_aftergroup_tl
4097
4098 \NewDocumentEnvironment{sassertion}{0{}}{
4099     \__stex_statements_sassertion_args:n{ #1 }
4100     \stex_if_smsmode:F {
4101         \seq_clear:N \l_tmpa_seq
4102         \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4103             \str_if_eq:nnF{ ##1 }{ }{
4104                 \stex_get_symbol:n { ##1 }
4105                 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4106                     \l_stex_get_symbol_uri_str
4107                 }
4108             }
4109         }

```

```

4110 \exp_args:Nnnx
4111 \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}
4112 \str_if_empty:NF \sassertiontype {
4113   \stex_annotate_invisible:nnn{type}{\sassertiontype}{ }
4114 }
4115 \clist_set:No \l_tmpa_clist \sassertiontype
4116 \tl_clear:N \l_tmpa_tl
4117 \clist_map_inline:Nn \l_tmpa_clist {
4118   \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
4119     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
4120   }
4121 }
4122 \tl_if_empty:NTF \l_tmpa_tl {
4123   \__stex_statements_sassertion_start:
4124 }{
4125   \l_tmpa_tl
4126 }
4127 }
4128 \stex_ref_new_doc_target:n \sassertionid
4129 \stex_smsmode_do:
4130 ){
4131   \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
4132   \stex_if_smsmode:F {
4133     \clist_set:No \l_tmpa_clist \sassertiontype
4134     \tl_clear:N \l_tmpa_tl
4135     \clist_map_inline:Nn \l_tmpa_clist {
4136       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
4137         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
4138       }
4139     }
4140     \tl_if_empty:NTF \l_tmpa_tl {
4141       \__stex_statements_sassertion_end:
4142     }{
4143       \l_tmpa_tl
4144     }
4145     \end{stex_annotate_env}
4146   }
4147 }

```

\stexpatchassertion

```

4148
4149 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
4150   \par\noindent\titllemph{Assertion~\tl_if_empty:NF \sassertiontitle {
4151     (\sassertiontitle)
4152   }~}
4153 }
4154 \cs_new_protected:Nn \__stex_statements_sassertion_end: { \par\medskip}
4155
4156 \newcommand\stexpatchassertion[3] [] {
4157   \str_set:Nx \l_tmpa_str{ #1 }
4158   \str_if_empty:NTF \l_tmpa_str {
4159     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
4160     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
4161   }{

```

```

4162     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
4163     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
4164   }
4165 }

```

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

`\inlineass` inline:

```

4166 \keys_define:nn {stex / inlineass }{
4167   type      .str_set_x:N = \sassertiontype,
4168   id        .str_set_x:N = \sassertionid,
4169   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4170   name      .str_set_x:N = \sassertionname
4171 }
4172 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
4173   \str_clear:N \sassertiontype
4174   \str_clear:N \sassertionid
4175   \str_clear:N \sassertionname
4176   \clist_clear:N \l__stex_statements_sassertion_for_clist
4177   \keys_set:nn { stex / inlineass }{ #1 }
4178 }
4179 \NewDocumentCommand \inlineass { 0{} m } {
4180   \begingroup
4181   \__stex_statements_inlineass_args:n{ #1 }
4182   \stex_ref_new_doc_target:n \sassertionid
4183   \stex_if_smsmode:TF{
4184     \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
4185   }{
4186     \seq_clear:N \l_tmpa_seq
4187     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4188       \str_if_eq:nnF{ ##1 }{ }{
4189         \stex_get_symbol:n { ##1 }
4190         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4191           \l_stex_get_symbol_uri_str
4192         }
4193       }
4194     }
4195     \exp_args:Nnx
4196     \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{
4197       \str_if_empty:NF \sassertiontype {
4198         \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
4199       }
4200       #2
4201       \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
4202     }
4203   }
4204   \endgroup
4205   \stex_smsmode_do:
4206 }

```

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

## 33.3 Examples

sexample

```

4207
4208 \keys_define:nn {stex / sexample }{
4209   type      .str_set_x:N = \exampletype,
4210   id        .str_set_x:N = \sexampleid,
4211   title     .tl_set:N     = \sexampletitle,
4212   for       .clist_set:N  = \l__stex_statements_sexample_for_clist,
4213 }
4214 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
4215   \str_clear:N \sexampletype
4216   \str_clear:N \sexampleid
4217   \tl_clear:N \sexampletitle
4218   \clist_clear:N \l__stex_statements_sexample_for_clist
4219   \keys_set:nn { stex / sexample }{ #1 }
4220 }
4221
4222 \NewDocumentEnvironment{sexample}{0{}}{
4223   \__stex_statements_sexample_args:n{ #1 }
4224   \stex_if_smsmode:F {
4225     \seq_clear:N \l_tmpa_seq
4226     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4227       \str_if_eq:nnF{ ##1 }{}{
4228         \stex_get_symbol:n { ##1 }
4229         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4230           \l_stex_get_symbol_uri_str
4231         }
4232       }
4233     }
4234     \exp_args:Nnnx
4235     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
4236     \str_if_empty:NF \sexampletype {
4237       \stex_annotate_invisible:nnn{type}{\sexampletype}{}
4238     }
4239     \clist_set:Nn \l_tmpa_clist \sexampletype
4240     \tl_clear:N \l_tmpa_tl
4241     \clist_map_inline:Nn \l_tmpa_clist {
4242       \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
4243         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
4244       }
4245     }
4246     \tl_if_empty:NTF \l_tmpa_tl {
4247       \__stex_statements_sexample_start:
4248     }{
4249       \l_tmpa_tl
4250     }
4251   }
4252   \stex_ref_new_doc_target:n \sexampleid
4253   \stex_smsmode_do:
4254 }{
4255   \str_if_empty:NF \sexamplename { \symdecl*{\sexamplename} }
4256   \stex_if_smsmode:F {
4257     \clist_set:Nn \l_tmpa_clist \sexampletype

```

```

4258 \tl_clear:N \l_tmpa_tl
4259 \clist_map_inline:Nn \l_tmpa_clist {
4260   \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
4261     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
4262   }
4263 }
4264 \tl_if_empty:NTF \l_tmpa_tl {
4265   \__stex_statements_sexample_end:
4266 }{
4267   \l_tmpa_tl
4268 }
4269 \end{stex_annotate_env}
4270 }
4271 }

```

`\stexpatchexample`

```

4272
4273 \cs_new_protected:Nn \__stex_statements_sexample_start: {
4274   \par\noindent\titllemph{Example~\tl_if_empty:NF \sexampltitle {
4275     (\sexampltitle)
4276   }~}
4277 }
4278 \cs_new_protected:Nn \__stex_statements_sexample_end: { \par\medskip}
4279
4280 \newcommand\stexpatchexample[3] [] {
4281   \str_set:Nx \l_tmpa_str{ #1 }
4282   \str_if_empty:NTF \l_tmpa_str {
4283     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
4284     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
4285   }{
4286     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
4287     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
4288   }
4289 }

```

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

`\inlineex` inline:

```

4290 \keys_define:nn {stex / inlineex }{
4291   type      .str_set_x:N = \sexampltype,
4292   id        .str_set_x:N = \sexampleid,
4293   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
4294   name      .str_set_x:N = \sexamplname
4295 }
4296 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
4297   \str_clear:N \sexampltype
4298   \str_clear:N \sexampleid
4299   \str_clear:N \sexamplname
4300   \clist_clear:N \l__stex_statements_sexample_for_clist
4301   \keys_set:nn { stex / inlineex }{ #1 }
4302 }
4303 \NewDocumentCommand \inlineex { 0{} m } {
4304   \begingroup
4305   \__stex_statements_inlineex_args:n{ #1 }

```

```

4306 \stex_ref_new_doc_target:n \sexampleid
4307 \stex_if_smsmode:TF{
4308   \str_if_empty:NF \sexamplename { \symdecl*{\examplename} }
4309 }{
4310   \seq_clear:N \l_tmpa_seq
4311   \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4312     \str_if_eq:nnF{ ##1 }{ }{
4313       \stex_get_symbol:n { ##1 }
4314       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4315         \l_stex_get_symbol_uri_str
4316       }
4317     }
4318   }
4319   \exp_args:Nnx
4320   \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpa_seq {},}{ }{
4321     \str_if_empty:NF \sexamplotype {
4322       \stex_annotate_invisible:nnn{type}{\sexamplotype}{ }
4323     }
4324     #2
4325     \str_if_empty:NF \sexamplename { \symdecl*{\sexamplename} }
4326   }
4327 }
4328 \endgroup
4329 \stex_smsmode_do:
4330 }

```

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

## 33.4 Logical Paragraphs

sparagraph

```

4331 \keys_define:nn { stex / sparagraph } {
4332   id      .str_set:N = \sparagraphid ,
4333   title   .tl_set:N  = \l_stex_sparagraph_title_tl ,
4334   type    .str_set:N  = \sparagraphtype ,
4335   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
4336   from    .tl_set:N   = \sparagraphfrom ,
4337   to      .tl_set:N   = \sparagraphto ,
4338   start   .tl_set:N   = \l_stex_sparagraph_start_tl ,
4339   name    .str_set:N   = \sparagraphname
4340 }
4341
4342 \cs_new_protected:Nn \stex_sparagraph_args:n {
4343   \tl_clear:N \l_stex_sparagraph_title_tl
4344   \tl_clear:N \sparagraphfrom
4345   \tl_clear:N \sparagraphto
4346   \tl_clear:N \l_stex_sparagraph_start_tl
4347   \str_clear:N \sparagraphid
4348   \str_clear:N \sparagraphtype
4349   \clist_clear:N \l__stex_statements_sparagraph_for_clist
4350   \str_clear:N \sparagraphname
4351   \keys_set:nn { stex / sparagraph }{ #1 }
4352 }

```

```

4353 \newif\if@in@omtext\@in@omtextfalse
4354
4355 \NewDocumentEnvironment {sparagraph} { 0{} } {
4356   \stex_sparagraph_args:n { #1 }
4357   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4358     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
4359   }{
4360     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
4361   }
4362   \@in@omtexttrue
4363   \stex_if_smsmode:F {
4364     \seq_clear:N \l_tmpa_seq
4365     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
4366       \str_if_eq:nnF{ ##1 }{}{
4367         \stex_get_symbol:n { ##1 }
4368         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4369           \l_stex_get_symbol_uri_str
4370         }
4371       }
4372     }
4373     \exp_args:Nnnx
4374     \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
4375     \str_if_empty:NF \sparagraphtype {
4376       \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}
4377     }
4378     \str_if_empty:NF \sparagraphfrom {
4379       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
4380     }
4381     \str_if_empty:NF \sparagraphto {
4382       \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
4383     }
4384     \clist_set:No \l_tmpa_clist \sparagraphtype
4385     \tl_clear:N \l_tmpa_tl
4386     \clist_map_inline:Nn \sparagraphtype {
4387       \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
4388         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
4389       }
4390     }
4391     \tl_if_empty:NTF \l_tmpa_tl {
4392       \__stex_statements_sparagraph_start:
4393     }{
4394       \l_tmpa_tl
4395     }
4396   }
4397   \stex_ref_new_doc_target:n \sparagraphid
4398   \stex_smsmode_do:
4399   \ignorespacesandpars
4400 }{
4401   \stex_if_smsmode:F {
4402     \clist_set:No \l_tmpa_clist \sparagraphtype
4403     \tl_clear:N \l_tmpa_tl
4404     \clist_map_inline:Nn \l_tmpa_clist {
4405       \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
4406         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}

```



```

4407     }
4408   }
4409   \str_if_empty:NF \sparagraphname { \symdecl*{\sparagraphname} }
4410   \tl_if_empty:NTF \l_tmpa_tl {
4411     \__stex_statements_sparagraph_end:
4412   }{
4413     \l_tmpa_tl
4414   }
4415   \end{stex_annotate_env}
4416 }
4417 }

```

# \stexpatchparagraph

```

4418
4419 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
4420   \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4421     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
4422       \titleemph{\l_stex_sparagraph_title_tl}:~
4423     }
4424   }{
4425     \titleemph{\l_stex_sparagraph_start_tl}~
4426   }
4427 }
4428 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
4429
4430 \newcommand\stexpatchparagraph[3] [] {
4431   \str_set:Nx \l_tmpa_str{ #1 }
4432   \str_if_empty:NTF \l_tmpa_str {
4433     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
4434     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
4435   }{
4436     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2 }
4437     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
4438   }
4439 }
4440
4441 \keys_define:nn { stex / inlinepara } {
4442   id      .str_set_x:N = \sparagraphid ,
4443   type    .str_set_x:N = \sparagraphtype ,
4444   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
4445   from    .tl_set:N    = \sparagraphfrom ,
4446   to      .tl_set:N    = \sparagraphto ,
4447   name    .str_set:N   = \sparagraphname
4448 }
4449 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
4450   \tl_clear:N \sparagraphfrom
4451   \tl_clear:N \sparagraphto
4452   \str_clear:N \sparagraphid
4453   \str_clear:N \sparagraphtype
4454   \clist_clear:N \l__stex_statements_sparagraph_for_clist
4455   \str_clear:N \sparagraphname
4456   \keys_set:nn { stex / inlinepara }{ #1 }
4457 }
4458 \NewDocumentCommand \inlinepara { 0{} m } {

```

```

4459 \begingroup
4460 \__stex_statements_inlinepara_args:n{ #1 }
4461 \stex_ref_new_doc_target:n \sparagraphid
4462 \stex_if_smsmode:TF{
4463   \str_if_empty:NF \sparagraphname { \symdecl*\sparagraphname } }
4464 }{
4465   \seq_clear:N \l_tmpa_seq
4466   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
4467     \str_if_eq:nnF{ ##1 }{}{
4468       \stex_get_symbol:n { ##1 }
4469       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4470         \l_stex_get_symbol_uri_str
4471       }
4472     }
4473   }
4474   \exp_args:Nnx
4475   \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
4476     \str_if_empty:NF \sparagraphtype {
4477       \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}
4478     }
4479     \str_if_empty:NF \sparagraphfrom {
4480       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
4481     }
4482     \str_if_empty:NF \sparagraphto {
4483       \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
4484     }
4485     #2
4486     \str_if_empty:NF \sparagraphname { \symdecl*\sparagraphname } }
4487   }
4488 }
4489 \endgroup
4490 \stex_smsmode_do:
4491 }
4492

```

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

symboldoc

```

4493 \NewDocumentEnvironment{symboldoc}{ m }{
4494   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
4495   \seq_clear:N \l_tmpb_seq
4496   \seq_map_inline:Nn \l_tmpa_seq {
4497     \str_if_eq:nnF{ ##1 }{}{
4498       \stex_get_symbol:n { ##1 }
4499       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
4500         \l_stex_get_symbol_uri_str
4501       }
4502     }
4503   }
4504   \par
4505   \exp_args:Nnnx
4506   \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
4507 }{
4508   \end{stex_annotate_env}
4509 }

```

4510 </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>

```
4511 <*package>
4512 <@@=stex_sproof>
4513
4514 %%%%%%%%%%% sproof.dtx %%%%%%%%%%%
4515
```

### 34.2 Proofs

We first define some keys for the proof environment.

```
4516 \keys_define:nn { stex / spf } {
4517   id          .str_set:N = \l__stex_sproof_spf_id_str,
4518   display     .tl_set:N  = \l__stex_sproof_spf_display_tl,
4519   for         .tl_set:N  = \l__stex_sproof_spf_for_tl ,
4520   from        .tl_set:N  = \l__stex_sproof_spf_from_tl ,
4521   proofend    .tl_set:N  = \l__stex_sproof_spf_proofend_tl,
4522   type        .tl_set:N  = \l__stex_sproof_spf_type_tl,
4523   title       .tl_set:N  = \l__stex_sproof_spf_title_tl,
4524   continues   .tl_set:N  = \l__stex_sproof_spf_continues_tl,
4525   functions   .tl_set:N  = \l__stex_sproof_spf_functions_tl,
4526   method      .tl_set:N  = \l__stex_sproof_spf_method_tl
4527 }
4528 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
4529   \str_clear:N \l__stex_sproof_spf_id_str
4530   \tl_clear:N \l__stex_sproof_spf_display_tl
4531   \tl_clear:N \l__stex_sproof_spf_for_tl
4532   \tl_clear:N \l__stex_sproof_spf_from_tl
4533   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
4534   \tl_clear:N \l__stex_sproof_spf_type_tl
4535   \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

```

4536 \tl_clear:N \l__stex_sproof_spf_continues_tl
4537 \tl_clear:N \l__stex_sproof_spf_functions_tl
4538 \tl_clear:N \l__stex_sproof_spf_method_tl
4539 \keys_set:nn { stex / spf }{ #1 }
4540 }

```

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

```

4541 \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.

```

4542 \newcount\count_ten
4543 \newenvironment{pst@with@label}[1]{
4544   \edef\pst@label{#1}
4545   \advance\count_ten by 1\relax
4546   \count_ten=1
4547 }{
4548   \advance\count_ten by -1\relax
4549 }

```

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

```

4550 \def\the@pst@label{
4551   \pst@make@label\pst@label{\number\count_ten}\l__stex_sproof_pstlabel_postfix_tl
4552 }

```

(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.

```

4553 \keys_define:nn { stex / pstlabel }{
4554   prefix      .tl_set:N   = \l__stex_sproof_pstlabel_prefix_tl,
4555   delimiter   .tl_set:N   = \l__stex_sproof_pstlabel_delimiter_tl,
4556   postfix     .tl_set:N   = \l__stex_sproof_pstlabel_postfix_tl
4557 }
4558 \cs_new_protected:Nn \__stex_sproof_pstlabel_args:n {

```

---

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

```

4559 \tl_set:Nn \l__stex_sproof_pstlabel_prefix_tl {P}
4560 \tl_set:Nn \l__stex_sproof_pstlabel_delimiter_tl {.}
4561 \tl_clear:N \l__stex_sproof_pstlabel_postfix_tl
4562 }
4563 \__stex_sproof_pstlabel_args:n {}
4564 \newcommand\setpstlabelstyle[1]{
4565   \__stex_sproof_pstlabel_args:n {#1}
4566 }
4567 \newcommand\setpstlabelstyledefault{%
4568   \__stex_sproof_pstlabel_args:n{prefix=P,delimiter=.,postfix={}}
4569 }

```

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

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

```

4570 \ExplSyntaxOff
4571 \def\pst@make@label@long#1#2{\@for\@I:=#1\do{\expandafter\expandafter\expandafter\@I\csname
4572 \def\pst@make@label@angles#1#2{\ensuremath{\@for\@I:=#1\do{\rangle}}#2}
4573 \def\pst@make@label@short#1#2{#2}
4574 \def\pst@make@label@empty#1#2{}
4575 \ExplSyntaxOn
4576 \def\pstlabelstyle#1{%
4577   \def\pst@make@label{\use:c{pst@make@label@#1}}%
4578 }%
4579 \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.

```

4580 \def\next@pst@label{%
4581   \global\advance\count\count10 by 1%
4582 }%

```

(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

```

4583 \def\sproof@box{
4584   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
4585 }
4586 \def\spf@proofend{\sproof@box}
4587 \def\sproofend{
4588   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
4589     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
4590   }
4591 }
4592 \def\sProofEndSymbol#1{\def\sproof@box{#1}}

```

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

**spf@\*@kw**

```

4593 \def\spf@proofsketch@kw{Proof Sketch}
4594 \def\spf@proof@kw{Proof}
4595 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

4596 \AddToHook{begindocument}{
4597   \ltx@ifpackageloaded{babel}{
4598     \makeatletter
4599     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4600     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4601       \input{sproof-ngerman.ldf}
4602     }
4603     \clist_if_in:NnT \l_tmpa_clist {finnish}{
4604       \input{sproof-finnish.ldf}
4605     }
4606     \clist_if_in:NnT \l_tmpa_clist {french}{
4607       \input{sproof-french.ldf}
4608     }
4609     \clist_if_in:NnT \l_tmpa_clist {russian}{
4610       \input{sproof-russian.ldf}
4611     }
4612     \makeatother
4613   }{}
4614 }

```

`spfsketch`

```

4615 \newcommand\spfsketch[2][]{
4616   \__stex_sproof_spf_args:n{#1}
4617   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4618     \titleemph{
4619       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
4620         \spf@proofsketch@kw
4621       }{
4622         \l__stex_sproof_spf_type_tl
4623       }
4624     }:
4625   }
4626   {~#2}
4627   %\sref@label@id{this \ifx\spf@type\@empty\spf@proofsketch@kw\else\spf@type\fi}
4628   \sproofend
4629 }

```

(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>

```

4630 \newenvironment{spfeq}[2][]{
4631   \__stex_sproof_spf_args:n{#1}
4632   %\sref@target
4633   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4634     \titleemph{
4635       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
4636         \spf@proof@kw
4637       }{

```

<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

```

4638     \l__stex_sproof_spf_type_tl
4639   }
4640   }:
4641 }
4642 {-#2}
4643 \begin{displaymath}\begin{array}{rcll}
4644 }{
4645   \end{array}\end{displaymath}
4646 }

```

(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.

```

4647 \newenvironment{spf@proof}[2] []{
4648   \l__stex_sproof_spf_args:n{#1}
4649   %\sref@target
4650   \count_ten=10
4651   \par\noindent
4652   \tl_if_eq:NNTF \l__stex_sproof_spf_display_tl\spf@flow{
4653     \titleemph{
4654       \tl_if_empty:NNTF \l__stex_sproof_spf_type_tl {
4655         \spf@proof@kw
4656       }{
4657         \l__stex_sproof_spf_type_tl
4658       }
4659     }:
4660   }
4661   {-#2}
4662   %\sref@label{id{this \ifx\spf@type\empty\spf@proof@kw\else\spf@type\fi}
4663   \def\pst@label{
4664     \newcount\pst@count% initialize the labeling mechanism
4665     \begin{description}\begin{pst@with@label}{\l__stex_sproof_pstlabel_prefix_tl}
4666   }{
4667     \end{pst@with@label}\end{description}
4668   }
4669   \newenvironment{sproof}[2] []{\begin{spf@proof}[#1]{#2}}{\sproofend\end{spf@proof}}
4670   \newenvironment{sProof}[2] []{\begin{spf@proof}[#1]{#2}}{\end{spf@proof}}

```

**\spfidea**

```

4671 \newcommand\spfidea[2] []{
4672   \l__stex_sproof_spf_args:n{#1}
4673   \titleemph{
4674     \tl_if_empty:NNTF \l__stex_sproof_spf_type_tl {Proof~Idea}{
4675       \l__stex_sproof_spf_type_tl
4676     }:
4677   }-#2
4678   \sproofend
4679 }

```

(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
4680 \newenvironment{spfstep}[1][]{
4681   \_stex_sproof_spf_args:n{#1}
4682   \@in@omtexttrue
4683   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4684     \item[\the@pst@label]
4685   }
4686   \tl_if_empty:NF \l__stex_sproof_spf_title_tl {
4687     {(\titleemph{\l__stex_sproof_spf_title_tl})\enspace}
4688   }
4689   %\sref@label{id{\pst@label}
4690   \ignorespacesandpars
4691 }{
4692   \next@pst@label\ignorespacesandpars
4693 }

```

**sproofcomment**

```

4694 \newenvironment{sproofcomment}[1][]{
4695   \_stex_sproof_spf_args:n{#1}
4696   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4697     \item[\the@pst@label]
4698   }
4699 }{
4700   \next@pst@label
4701 }

```

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.

```

4702 \newenvironment{subproof}[2][]{
4703   \_stex_sproof_spf_args:n{#1}
4704   \def\@test{#2}
4705   \ifx\@test\empty\else
4706     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4707       \item[\the@pst@label]
4708     }{#2}
4709   \fi
4710   \begin{pst@with@label}{\pst@label,\number\count_ten}
4711 }{
4712   \end{pst@with@label}\next@pst@label
4713 }

```

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

```

4714 \newenvironment{spfcases}[2][]{
4715   \def\@test{#1}
4716   \ifx\@test\empty
4717     \begin{subproof}[method=by-cases]{#2}

```

---

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

```

4718 \else
4719   \begin{subproof}[#1,method=by-cases]{#2}
4720 \fi
4721 }{
4722   \end{subproof}
4723 }

```

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

```

4724 \newenvironment{spfcase}[2] [] {
4725   \__stex_sproof_spf_args:n{#1}
4726   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4727     \item[\the@pst@label]
4728   }
4729   \def\@test{#2}
4730   \ifx\@test\@empty
4731     \else
4732       {\titleemph{#2}:~}
4733     \fi
4734     \begin{pst@with@label}{\pst@label,\number\count_ten}
4735   }{
4736     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4737       \sproofend
4738     }
4739     \end{pst@with@label}
4740     \next@pst@label
4741   }

```

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

```

4742 \newcommand\spfcasesketch[3] [] {
4743   \__stex_sproof_spf_args:n{#1}
4744   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4745     \item[\the@pst@label]
4746   }
4747   \def\@test{#2}
4748   \ifx\@test\@empty
4749     \else
4750       {\titleemph{#2}:~}
4751     \fi#3
4752     \next@pst@label
4753 }%

```

### 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.

```

4754 \keys_define:nn { stex / just }{
4755   id      .str_set_x:N = \l__stex_sproof_just_id_str,
4756   method  .tl_set:N   = \l__stex_sproof_just_method_tl,
4757   premises .tl_set:N   = \l__stex_sproof_just_premises_tl,
4758   args    .tl_set:N   = \l__stex_sproof_just_args_tl
4759 }

```

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`

4760 `\newenvironment{justification}[1] [] {}{}`

`\premise`

4761 `\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.

4762 `\newcommand\justarg[2] [] {#2}`

4763 `\</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

```
4764 <*package>
4765
4766 %%%%%%%%%% others.dtx %%%%%%%%%%
4767
4768 <@@=stex_others>
      Warnings and error messages
4769 % None

\MSC Math subject classifier

4770 \NewDocumentCommand \MSC {m} {
4771 % TODO
4772 }

(End definition for \MSC. This function is documented on page 21.)
      Patching tikzinput, if loaded
4773 \@ifpackageloaded{tikzinput}{
4774 \RequirePackage{stex-tikzinput}
4775 }{}
4776 </package>
```

## Chapter 36

# STEX -Metatheory Implementation

```
4777 <*package>
4778 <@@=stex_modules>
4779
4780 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
4781
4782 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
4783 \begingroup
4784 \stex_module_setup:nn{
4785   ns=\c_stex_metatheory_ns_str,
4786   meta=NONE
4787 }{Metatheory}
4788 \stex_reactivate_macro:N \symdecl
4789 \stex_reactivate_macro:N \notation
4790 \stex_reactivate_macro:N \symdef
4791 \ExplSyntaxOff
4792 \csname stex_suppress_html:n\endcsname{
4793   % is-a (a:A, a \in A, a is an A, etc.)
4794   \symdecl[args=ai]{isa}
4795   \notation[typed]{isa}{#1 \comp{:} #2}{#1 \comp, #2}
4796   \notation[in]{isa}{#1 \comp\in #2}{#1 \comp, #2}
4797   \notation[pred]{isa}{#2\comp(#1 \comp)}{#1 \comp, #2}
4798
4799   % bind (\forall, \Pi, \lambda etc.)
4800   \symdecl[args=Bi]{bind}
4801   \notation[forall]{bind}{\comp\forall #1.\;#2}{#1 \comp, #2}
4802   \notation[\Pi]{bind}{\comp\prod_{#1}#2}{#1 \comp, #2}
4803   \notation[deffun]{bind}{\comp( #1 \comp)\; \to\;}{#1 \comp, #2}
4804
4805   % dummy variable
4806   \symdecl{dummyvar}
4807   \notation[underscore]{dummyvar}{\comp\_}
4808   \notation[dot]{dummyvar}{\comp\cdot}
4809   \notation[dash]{dummyvar}{\comp{\rm --}}
4810
4811   %fromto (function space, Hom-set, implication etc.)
```

```

4812 \symdecl[args=ai]{fromto}
4813 \notation[xarrow]{fromto}{#1 \comp\to #2}{#1 \comp\times #2}
4814 \notation[arrow]{fromto}{#1 \comp\to #2}{#1 \comp\to #2}
4815
4816 % mapto (lambda etc.)
4817 %\symdecl[args=Bi]{mapto}
4818 %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
4819 %\notation[lambda]{mapto}{\comp\lambda #1 \comp. \; #2}{#1 \comp, #2}
4820 %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp. \; #2}{#1 \comp, #2}
4821
4822 % function/operator application
4823 \symdecl[args=ia]{apply}
4824 \notation[prec=0;0x\infpres,parens]{apply}{#1 \comp( #2 \comp)}{#1 \comp, #2}
4825 \notation[prec=0;0x\infpres,lambda]{apply}{#1 \; #2 }{#1 \; #2}
4826
4827 % ‘type’ of all collections (sets, classes, types, kinds)
4828 \symdecl{collection}
4829 \notation[U]{collection}{\comp{\mathcal{U}}}
4830 \notation[set]{collection}{\comp{\textsf{Set}}}
4831
4832 % sequences
4833 \symdecl[args=1]{seqtype}
4834 \notation[kleene]{seqtype}{#1^{\comp\ast}}
4835
4836 \symdef[args=2,li,prec=nobrackets]{sequence-index}{#1}_{#2}
4837 \notation[ui,prec=nobrackets]{sequence-index}{#1}^{#2}
4838
4839 %\symdef[args=3,li]{sequence-from-to}{#1}_{#2}\comp{\,\ellipses,}#1_{#3}
4840 %\notation[ui]{sequence-from-to}{#1}^{#2}\comp{\,\ellipses,}#1^{#3}
4841 % ^ superceded by \aseqfromto and \livar/\uivar
4842
4843 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{\,\ellipses}}{#1\comp,#2}
4844 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{\,\ellipses,}#2}{#1\comp,#2}
4845 \symdef[args=aui,prec=nobrackets]{aseqfromtovia}{#1\comp{\,\ellipses,}#2\comp{\,\ellipses,}#3}
4846
4847 % letin (‘let’, local definitions, variable substitution)
4848 \symdecl[args=bii]{letin}
4849 \notation[let]{letin}{\comp{\rm let}}{\;#1\comp{=}\;#2\; \comp{\rm in}}{\;#3}
4850 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
4851 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
4852
4853 % structures
4854 \symdecl*[args=1]{module-type}
4855 \notation{module-type}{\mathtt{MOD} #1}
4856 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
4857 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{#1 \comp, #2}
4858
4859 }
4860 \ExplSyntaxOn
4861 \stex_add_to_current_module:n{
4862   \let\nappa\apply
4863   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
4864   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
4865   \def\livar{\csname sequence-index\endcsname[li]}

```

```

4866 \def\uivar{\csname sequence-index\endcsname[ui]}
4867 \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
4868 \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
4869 \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
4870 }
4871 \__stex_modules_end_module:
4872 \endgroup
4873 \</package>

```

## Chapter 37

# Tikzinput Implementation

```
4874 <*package>
4875
4876 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
4877
4878 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
4879 \RequirePackage{l3keys2e}
4880
4881 \keys_define:nn { tikzinput } {
4882   image .bool_set:N = \c_tikzinput_image_bool,
4883   image .default:n = false ,
4884   unknown .code:n = {}
4885 }
4886
4887 \ProcessKeysOptions { tikzinput }
4888
4889 \bool_if:NTF \c_tikzinput_image_bool {
4890   \RequirePackage{graphicx}
4891
4892   \providecommand\usetikzlibrary[]{}
4893   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
4894 }{
4895   \RequirePackage{tikz}
4896   \RequirePackage{standalone}
4897
4898   \newcommand \tikzinput [2] [] {
4899     \setkeys{Gin}{#1}
4900     \ifx \Gin@ewidth \Gin@exclamation
4901       \ifx \Gin@eheight \Gin@exclamation
4902         \input { #2 }
4903       \else
4904         \resizebox{!}{ \Gin@eheight }{
4905           \input { #2 }
4906         }
4907       \fi
4908     \else
4909       \ifx \Gin@eheight \Gin@exclamation
4910         \resizebox{ \Gin@ewidth }{!}{
4911           \input { #2 }
```



```

4912     }
4913     \else
4914         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
4915             \input { #2 }
4916         }
4917     \fi
4918 \fi
4919 }
4920 }
4921
4922 \newcommand \ctikzinput [2] [] {
4923     \begin{center}
4924         \tikzinput [1] {#2}
4925     \end{center}
4926 }
4927
4928 \@ifpackageloaded{stex}{
4929     \RequirePackage{stex-tikzinput}
4930 }{}
4931
4932 </package>
4933 <*stex>
4934 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
4935 \RequirePackage{stex}
4936 \RequirePackage{tikzinput}
4937
4938 \newcommand\mhtikzinput [2] [] {%
4939     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
4940     \stex_in_repository:nn\Gin@mhrepos{
4941         \tikzinput [1]{\mhpath{##1}{#2}}
4942     }
4943 }
4944 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [1] {#2}\end{center}}
4945 </stex>

```

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

## Chapter 38

# document-structure.sty Implementation

### 38.1 The document-structure Class

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

```
4946 \*cls)
4947 \@@=document_structure)
4948 \ProvidesExplClass{document-structure}{2022/02/10}{3.0}{Modular Document Structure Class}
4949 \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`

```
4950 \keys_define:nn{ document-structure / pkg }{
4951   class      .str_set_x:N = \c_document_structure_class_str,
4952   minimal    .bool_set:N = \c_document_structure_minimal_bool,
4953   report     .code:n      = {
4954     \ClassWarning{document-structure}{the option 'report' is deprecated, use 'class=report',
4955     \str_set:Nn \c_document_structure_class_str {report}
4956   },
4957   book       .code:n      = {
4958     \ClassWarning{document-structure}{the option 'book' is deprecated, use 'class=book', ins
4959     \str_set:Nn \c_document_structure_class_str {book}
4960   },
4961   bookpart   .code:n      = {
4962     \ClassWarning{document-structure}{the option 'bookpart' is deprecated, use 'class=book,t
4963     \str_set:Nn \c_document_structure_class_str {book}
4964     \str_set:Nn \c_document_structure_topsect_str {chapter}
4965   },
```

```

4966 docopt      .str_set_x:N = \c_document_structure_docopt_str,
4967 unknown     .code:n      = {
4968   \PassOptionsToPackage{ \CurrentOption }{ document-structure }
4969 }
4970 }
4971 \ProcessKeysOptions{ document-structure / pkg }
4972 \str_if_empty:NT \c_document_structure_class_str {
4973   \str_set:Nn \c_document_structure_class_str {article}
4974 }
4975 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
4976   {\c_document_structure_class_str}
4977

```

### 38.3 Beefing up the document environment

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

```

4978 \RequirePackage{document-structure}
4979 \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>

```

4980 \keys_define:nn { document-structure / document }{
4981   id .str_set_x:N = \c_document_structure_document_id_str
4982 }
4983 \let\__document_structure_orig_document=\document
4984 \renewcommand{\document}[1][]{
4985   \keys_set:nn{ document-structure / document }{ #1 }
4986   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
4987   \__document_structure_orig_document
4988 }

```

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

```

4989 }
4990 \</cls>

```

### 38.4 Implementation: document-structure Package

```

4991 \<*package>
4992 \ProvidesExplPackage{document-structure}{2022/02/10}{3.0}{Modular Document Structure}
4993 \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

```

4994
4995 \keys_define:nn{ document-structure / pkg }{
4996   class      .str_set_x:N = \c_document_structure_class_str,
4997   topsect    .str_set_x:N = \c_document_structure_topsect_str,
4998   % showignores .bool_set:N = \c_document_structure_showignores_bool,
4999 }
5000 \ProcessKeysOptions{ document-structure / pkg }
5001 \str_if_empty:NT \c_document_structure_class_str {
5002   \str_set:Nn \c_document_structure_class_str {article}
5003 }
5004 \str_if_empty:NT \c_document_structure_topsect_str {
5005   \str_set:Nn \c_document_structure_topsect_str {section}
5006 }

```

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

```

5007 \RequirePackage{xspace}
5008 \RequirePackage{comment}
5009 \AddToHook{begindocument}{
5010   \ltx@ifpackageloaded{babel}{
5011     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5012     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5013       \makeatletter\input{omdoc-ngerman.ldf}\makeatother
5014     }
5015   }{}
5016 }

```

`\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.

```

5017 \int_new:N \l_document_structure_section_level_int
5018 \str_case:VnF \c_document_structure_topsect_str {
5019   {part}}{
5020     \int_set:Nn \l_document_structure_section_level_int {0}
5021   }
5022   {chapter}{
5023     \int_set:Nn \l_document_structure_section_level_int {1}
5024   }
5025 }{
5026   \str_case:VnF \c_document_structure_class_str {
5027     {book}}{
5028       \int_set:Nn \l_document_structure_section_level_int {0}
5029     }
5030     {report}}{
5031       \int_set:Nn \l_document_structure_section_level_int {0}
5032     }
5033   }{
5034     \int_set:Nn \l_document_structure_section_level_int {2}
5035   }
5036 }

```

## 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

```
5037 \def\current@section@level{document}%
5038 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
5039 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
5040 \cs_new_protected:Npn \skipomgroup {
5041   \ifcase\l_document_structure_section_level_int
5042   \or\stepcounter{part}
5043   \or\stepcounter{chapter}
5044   \or\stepcounter{section}
5045   \or\stepcounter{subsection}
5046   \or\stepcounter{subsubsection}
5047   \or\stepcounter{paragraph}
5048   \or\stepcounter{subparagraph}
5049   \fi
5050 }
```

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

`blindomgroup`

```
5051 \newcommand\at@begin@blindomgroup[1]{%
5052 \newenvironment{blindomgroup}
5053 {
5054   \int_incr:N\l_document_structure_section_level_int
5055   \at@begin@blindomgroup\l_document_structure_section_level_int
5056 }{}}
```

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

```
5057 \newcommand\omgroup@nonum[2]{
5058   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
5059   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
5060 }
```

*(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.

```
5061 \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.

```

5062 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
5063   \@nameuse{#1}{#2}
5064 }{
5065   \cs_if_exist:NTF\rdfmata@sectioning{
5066     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
5067   }{
5068     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
5069   }
5070 }
5071 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
5072 }

```

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

omgroup

```

5073 \keys_define:nn { document-structure / omgroup }{
5074   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
5075   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
5076   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
5077   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
5078   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
5079   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
5080   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
5081   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
5082   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
5083   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
5084 }
5085 \cs_new_protected:Nn \__document_structure_omgroup_args:n {
5086   \str_clear:N \l__document_structure_omgroup_id_str
5087   \str_clear:N \l__document_structure_omgroup_date_str
5088   \clist_clear:N \l__document_structure_omgroup_creators_clist
5089   \clist_clear:N \l__document_structure_omgroup_contributors_clist
5090   \tl_clear:N \l__document_structure_omgroup_srccite_tl
5091   \tl_clear:N \l__document_structure_omgroup_type_tl
5092   \tl_clear:N \l__document_structure_omgroup_short_tl
5093   \tl_clear:N \l__document_structure_omgroup_display_tl
5094   \tl_clear:N \l__document_structure_omgroup_intro_tl
5095   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
5096   \keys_set:nn { document-structure / omgroup } { #1 }
5097 }

```

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.

```

5098 \newif\if@mainmatter\@mainmattertrue
5099 \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.

```

5100 \keys_define:nn { document-structure / sectioning }{
5101   name .str_set_x:N = \l__document_structure_sect_name_str ,
5102   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
5103   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
5104   num .bool_set:N = \l__document_structure_sect_num_bool ,
5105 }

```

```

5106 \cs_new_protected:Nn \__document_structure_sect_args:n {
5107   \str_clear:N \l__document_structure_sect_name_str
5108   \str_clear:N \l__document_structure_sect_ref_str
5109   \bool_set_false:N \l__document_structure_sect_clear_bool
5110   \bool_set_false:N \l__document_structure_sect_num_bool
5111   \keys_set:nn { document-structure / sectioning } { #1 }
5112 }
5113 \newcommand\omdoc@sectioning[3][]{
5114   \__document_structure_sect_args:n {#1}
5115   \let\omdoc@sect@name\l__document_structure_sect_name_str
5116   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
5117   \if@mainmatter% numbering not overridden by frontmatter, etc.
5118     \bool_if:NTF \l__document_structure_sect_num_bool {
5119       \omgroup@num{#2}{#3}
5120     }{
5121       \omgroup@nonum{#2}{#3}
5122     }
5123   \def\current@section@level{\omdoc@sect@name}
5124   \else
5125     \omgroup@nonum{#2}{#3}
5126   \fi
5127 }% 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.

```

5128 \newcommand\omgroup@redefine@addtocontents[1]{%
5129 %\edef\__document_structureimport{#1}%
5130 %\@for\@I:=\__document_structureimport\do{%
5131 %\edef\@path{\csname module@\@I @path\endcsname}%
5132 %\@ifundefined{tf@toc}\relax%
5133 %   {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
5134 %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
5135 %\def\addcontentsline##1##2##3{%
5136 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
5137 %\else% hyperref.sty not loaded
5138 %\def\addcontentsline##1##2##3{%
5139 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
5140 %\fi
5141 }% 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.

```

5142 \int_new:N \l_document_structure_omgroup_level_int
5143 \newenvironment{omgroup}[2][]{% keys, title
5144 {
5145   \__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.

```

5146 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
5147   \omgroup@redefine@addtocontents{
5148     %\@ifundefined{module@id}\used@modules%
5149     %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}

```

```

5150     }
5151 }

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

5152 \int_incr:N \l_document_structure_omgroup_level_int
5153 \int_incr:N \l_document_structure_section_level_int
5154 \ifcase\l_document_structure_section_level_int
5155   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
5156   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
5157   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
5158   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
5159   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
5160   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
5161   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
5162 \fi
5163 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
5164 \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
5165 }% for customization
5166 {}

```

and finally, we localize the sections

```

5167 \newcommand\omdoc@part@kw{Part}
5168 \newcommand\omdoc@chapter@kw{Chapter}
5169 \newcommand\omdoc@section@kw{Section}
5170 \newcommand\omdoc@subsection@kw{Subsection}
5171 \newcommand\omdoc@subsubsection@kw{Subsubsection}
5172 \newcommand\omdoc@paragraph@kw{paragraph}
5173 \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`

```

5174 \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).

```

5175 \cs_if_exist:NTF\frontmatter{
5176   \let\__document_structure_orig_frontmatter\frontmatter
5177   \let\frontmatter\relax
5178 }{
5179   \tl_set:Nn\__document_structure_orig_frontmatter{
5180     \clearpage
5181     \@mainmatterfalse
5182     \pagenumbering{roman}
5183   }
5184 }

```



```

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

```

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.

```

5195 \newenvironment{frontmatter}{
5196   \__document_structure_orig_frontmatter
5197 }{
5198   \cs_if_exist:NTF\mainmatter{
5199     \mainmatter
5200   }{
5201     \clearpage
5202     \@mainmattertrue
5203     \pagenumbering{arabic}
5204   }
5205 }

```

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

```

5206 \newenvironment{backmatter}{
5207   \__document_structure_orig_backmatter
5208 }{
5209   \cs_if_exist:NTF\mainmatter{
5210     \mainmatter
5211   }{
5212     \clearpage
5213     \@mainmattertrue
5214     \pagenumbering{arabic}
5215   }
5216 }

```

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

5217 \@mainmattertrue\pagenumbering{arabic}

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

```

5218 \def \c__document_structure_document_str{document}
5219 \newcommand\afterprematurestop{}
5220 \def\prematurestop@endomgroup{
5221   \unless\ifx\@currentvir\c__document_structure_document_str
5222     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
5223       \expandafter\prematurestop@endomgroup
5224   \fi
5225 }

```

```

5226 \providecommand\prematurestop{
5227   \message{Stopping~sTeX~processing~prematurely}
5228   \prematurestop@endomgroup
5229   \afterprematurestop
5230   \end{document}
5231 }

```

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

## 38.8 Global Variables

**\setSGvar** set a global variable

```

5232 \RequirePackage{etoolbox}
5233 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

**\useSGvar** use a global variable

```

5234 \newrobustcmd\useSGvar[1]{%
5235   \@ifundefined{sTeX@Gvar@#1}
5236   {\PackageError{document-structure}
5237    {The sTeX Global variable #1 is undefined}
5238    {set it with \protect\setSGvar}}
5239   \@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.

```

5240 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
5241   \@ifundefined{sTeX@Gvar@#1}
5242   {\PackageError{document-structure}
5243    {The sTeX Global variable #1 is undefined}
5244    {set it with \protect\setSGvar}}
5245   {\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.

```
5246 \*cls)
5247 \@@=notesslides)
5248 \ProvidesExplClass{notesslides}{2022/02/10}{3.0}{notesslides Class}
5249 \RequirePackage{l3keys2e,expl-keystr-compatible}
5250
5251 \keys_define:nn{notesslides / cls}{
5252   class .code:n = {
5253     \PassOptionsToClass{\CurrentOption}{omdoc}
5254     \str_if_eq:nnT{#1}{book}{
5255       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5256     }
5257     \str_if_eq:nnT{#1}{report}{
5258       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5259     }
5260   },
5261   notes .bool_set:N = \c__notesslides_notes_bool ,
5262   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
5263   unknown .code:n = {
5264     \PassOptionsToClass{\CurrentOption}{omdoc}
5265     \PassOptionsToClass{\CurrentOption}{beamer}
5266     \PassOptionsToPackage{\CurrentOption}{notesslides}
5267   }
5268 }
5269 \ProcessKeysOptions{ notesslides / cls }
5270 \bool_if:NTF \c__notesslides_notes_bool {
5271   \PassOptionsToPackage{notes=true}{notesslides}
5272 }{
5273   \PassOptionsToPackage{notes=false}{notesslides}
5274 }
5275 \</cls)
```

now we do the same for the notesslides package.

```

5276 <*package>
5277 \ProvidesExplPackage{notesslides}{2022/02/10}{3.0}{notesslides Package}
5278 \RequirePackage{l3keys2e,expl-keystr-compat}
5279
5280 \keys_define:nn{notesslides / pkg}{
5281   topsect      .str_set_x:N = \c__notesslides_topsect_str,
5282   defaulttopsect .str_set_x:N = \c__notesslides_defaulttopsec_str,
5283   notes        .bool_set:N = \c__notesslides_notes_bool ,
5284   slides       .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
5285   sectocframes .bool_set:N = \c__notesslides_sectocframes_bool ,
5286   frameimages  .bool_set:N = \c__notesslides_frameimages_bool ,
5287   fiboxed      .bool_set:N = \c__notesslides_fiboxed_bool ,
5288   nopproblems  .bool_set:N = \c__notesslides_nopproblems_bool,
5289   unknown      .code:n      = {
5290     \PassOptionsToClass{\CurrentOption}{stex}
5291     \PassOptionsToClass{\CurrentOption}{tikzinput}
5292   }
5293 }
5294 \ProcessKeysOptions{ notesslides / pkg }
5295 \newif\ifnotes
5296 \bool_if:NTF \c__notesslides_notes_bool {
5297   \notesttrue
5298 }{
5299   \notesfalse
5300 }
5301

```

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

```

5302 \str_if_empty:NTF \c__notesslides_topsect_str {
5303   \str_set_eq:NN \__notesslides_topsect \c__notesslides_defaulttopsec_str
5304 }{
5305   \str_set_eq:NN \__notesslides_topsect \c__notesslides_topsect_str
5306 }
5307 </package>

```

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

```

5308 <*cls>
5309 \bool_if:NTF \c__notesslides_notes_bool {
5310   \LoadClass{document-structure}
5311 }{
5312   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
5313   \newcounter{Item}
5314   \newcounter{paragraph}
5315   \newcounter{subparagraph}
5316   \newcounter{Hfootnote}
5317   \RequirePackage{document-structure}
5318 }

```

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

```

5319 \RequirePackage{notesslides}
5320 </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.

```

5321 \*package>
5322 \bool_if:NT \c__notesslides_notes_bool {
5323   \RequirePackage{a4wide}
5324   \RequirePackage{marginnote}
5325   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
5326   \RequirePackage{mdframed}
5327   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
5328   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
5329 }
5330 \RequirePackage{stex-tikzinput}
5331 \RequirePackage{etoolbox}
5332 \RequirePackage{amssymb}
5333 \RequirePackage{amsmath}
5334 \RequirePackage{comment}
5335 \RequirePackage{textcomp}
5336 \RequirePackage{url}
5337 \RequirePackage{graphicx}
5338 \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>

```

5339 \bool_if:NT \c__notesslides_notes_bool {
5340   \renewcommand\usetheme[2][\]{\usepackage[#1]{beamernotestheme#2}}
5341 }

```

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

```

5342 \newcounter{slide}
5343 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
5344 \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.

```

5345 \bool_if:NTF \c__notesslides_notes_bool {
5346   \renewenvironment{note}{\ignorespaces}{\ignorespaces}
5347 }{
5348   \excludcomment{note}
5349 }

```

---

<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.

```
5350 \bool_if:NT \c__notesslides_notes_bool {
5351   \newlength{\slideframewidth}
5352   \setlength{\slideframewidth}{1.5pt}
```

**frame** We first define the keys.

```
5353 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
5354   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
5355     \bool_set_true:N #1
5356   }{
5357     \bool_set_false:N #1
5358   }
5359 }
5360 \keys_define:nn{notesslides / frame}{
5361   label .str_set_x:N = \l__notesslides_frame_label_str,
5362   allowframebreaks .code:n = {
5363     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
5364   },
5365   allowdisplaybreaks .code:n = {
5366     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
5367   },
5368   fragile .code:n = {
5369     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
5370   },
5371   shrink .code:n = {
5372     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
5373   },
5374   squeeze .code:n = {
5375     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
5376   },
5377   t .code:n = {
5378     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
5379   },
5380 }
5381 \cs_new_protected:Nn \__notesslides_frame_args:n {
5382   \str_clear:N \l__notesslides_frame_label_str
5383   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
5384   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
5385   \bool_set_true:N \l__notesslides_frame_fragile_bool
5386   \bool_set_true:N \l__notesslides_frame_shrink_bool
5387   \bool_set_true:N \l__notesslides_frame_squeeze_bool
5388   \bool_set_true:N \l__notesslides_frame_t_bool
5389   \keys_set:nn { notesslides / frame }{ #1 }
5390 }
```

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

```
5391 \renewenvironment{frame}[1][]{
5392   \__notesslides_frame_args:n{#1}
5393   \sffamily
5394   \stepcounter{slide}
5395   \def\@currentlabel{\theslide}
5396   \str_if_empty:NF \l__notesslides_frame_label_str {
5397     \label{\l__notesslides_frame_label_str}
```

5398 }  
5399

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

5399 \def\itemize@level{outer}  
5400 \def\itemize@outer{outer}  
5401 \def\itemize@inner{inner}  
5402 \renewcommand\newpage{\addtocounter{framenum}{1}}  
5403 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}  
5404 \renewenvironment{itemize}{  
5405 \ifx\itemize@level\itemize@outer  
5406 \def\itemize@label{\\$ \rhd\$}  
5407 \fi  
5408 \ifx\itemize@level\itemize@inner  
5409 \def\itemize@label{\\$ \scriptstyle \rhd\$}  
5410 \fi  
5411 \begin{list}  
5412 {\itemize@label}  
5413 {\setlength{\labelsep}{.3em}  
5414 \setlength{\labelwidth}{.5em}  
5415 \setlength{\leftmargin}{1.5em}  
5416 }  
5417 \edef\itemize@level{\itemize@inner}  
5418 }{  
5419 \end{list}  
5420 }

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

5421 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth=1ex]  
5422 }{  
5423 \medskip\miko@slidelabel\end{mdframed}  
5424 }

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

\frametitle

5425 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip  
5426 }

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

EdN:21

\pause 21

5427 \bool\_if:NT \c\_\_notesslides\_notes\_bool {  
5428 \newcommand\pause{  
5429 }

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

nparagraph

5430 \bool\_if:NTF \c\_\_notesslides\_notes\_bool {  
5431 \newenvironment{nparagraph}[1][\begin{sparagraph}[#1]}{\end{sparagraph}}  
5432 }{  
5433 \excludecomment{nparagraph}  
5434 }

---

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

```

nomgroup
5435 \bool_if:NTF \c__notesslides_notes_bool {
5436   \newenvironment{nomgroup}[2] [] {\begin{omgroup}[#1]{#2}}{\end{omgroup}}
5437 }{
5438   \excludecomment{nomgroup}
5439 }

ndefinition
5440 \bool_if:NTF \c__notesslides_notes_bool {
5441   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}{\end{sdefinition}}
5442 }{
5443   \excludecomment{ndefinition}
5444 }

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

nsproof
5450 \bool_if:NTF \c__notesslides_notes_bool {
5451   \newenvironment{nproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
5452 }{
5453   \excludecomment{nproof}
5454 }

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

\inputref@*skip We customize the hooks for in \inputref.
5460 \def\inputref@preskip{\smallskip}
5461 \def\inputref@postskip{\medskip}

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

\inputref*
5462 \let\orig@inputref\inputref
5463 \def\inputref{\@ifstar\ninputref\orig@inputref}
5464 \newcommand\ninputref[2] [] {
5465   \bool_if:NT \c__notesslides_notes_bool {
5466     \orig@inputref[#1]{#2}
5467   }
5468 }

(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>}`.

```

5469 \newlength{\slidelogoheight}
5470
5471 \bool_if:NTF \c__notesslides_notes_bool {
5472   \setlength{\slidelogoheight}{.4cm}
5473 }{
5474   \setlength{\slidelogoheight}{1cm}
5475 }
5476 \newsavebox{\slidelogo}
5477 \sbox{\slidelogo}{\TeX}
5478 \newrobustcmd{\setslidelogo}[1]{
5479   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
5480 }
```

(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.

```

5481 \def\source{Michael Kohlhase}% customize locally
5482 \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.

```

5483 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
5484 \newsavebox{\cclogo}
5485 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
5486 \newif\ifcchref\cchreffalse
5487 \AtBeginDocument{
5488   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
5489 }
5490 \def\licensing{
5491   \ifcchref
5492     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
5493   \else
5494     {\usebox{\cclogo}}
5495   \fi
5496 }
5497 \newrobustcmd{\setlicensing}[2][]{
5498   \def@url{#1}
5499   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
5500   \ifx@url@empty
5501     \def\licensing{{\usebox{\cclogo}}}
5502   \else
5503     \def\licensing{
```

```

5504     \ifcchref
5505     \href{#1}{\usebox{\cclogo}}
5506     \else
5507     {\usebox{\cclogo}}
5508     \fi
5509   }
5510 \fi
5511 }

```

(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>

```

5512 \newrobustcmd\miko@slidelabel{
5513   \vbox to \slidelogoheight{
5514     \vss\hbox to \slidewidth
5515     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
5516   }
5517 }

```

(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.

```

5518 \def\Gin@mhrepos{}
5519 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
5520 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
5521 \newrobustcmd\frameimage[2][{}]{
5522   \stepcounter{slide}
5523   \bool_if:NT \c__notesslides_frameimages_bool {
5524     \def\Gin@ewidth{}\setkeys{Gin}{#1}
5525     \bool_if:NF \c__notesslides_notes_bool { \vfill }
5526     \begin{center}
5527       \bool_if:NTF \c__notesslides_fiboxed_bool {
5528         \fbox{
5529           \ifx\Gin@ewidth\@empty
5530             \ifx\Gin@mhrepos\@empty
5531               \mhgraphics[width=\slidewidth,#1]{#2}
5532             \else
5533               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5534             \fi
5535           \else% Gin@ewidth empty
5536             \ifx\Gin@mhrepos\@empty
5537               \mhgraphics[#1]{#2}
5538             \else
5539               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5540             \fi
5541           \fi% Gin@ewidth empty
5542         }
5543       }{
5544         \ifx\Gin@ewidth\@empty

```

---

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

```

5545         \ifx\Gin@mhrepos\@empty
5546             \mhgraphics[width=\slidewidth,#1]{#2}
5547         \else
5548             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5549         \fi
5550         \ifx\Gin@mhrepos\@empty
5551             \mhgraphics[#1]{#2}
5552         \else
5553             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5554         \fi
5555     \fi% Gin@ewidth empty
5556 }
5557 \end{center}
5558 \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
5559 \bool_if:NF \c__notesslides_notes_bool { \vfill }
5560 }
5561 } % 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.

```

5562 \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

```

5563 \AddToHook{begindocument}{
5564     \definecolor{green}{rgb}{0,.5,0}
5565     \definecolor{purple}{cmyk}{.3,1,0,.17}
5566 }

```

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`.

```

5567 % \def\STpresent#1{\textcolor{blue}{#1}}
5568 \def\defemph#1{\textcolor{magenta}{#1}}
5569 \def\symrefemph#1{\textcolor{cyan}{#1}}
5570 \def\compemph#1{\textcolor{blue}{#1}}
5571 \def\titleemph#1{\textcolor{blue}{#1}}
5572 \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.

```

5573 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
5574 \def\smalltextwarning{
5575     \pgfuseimage{miko@small@dbend}
5576     \xspace
5577 }
5578 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}

```

```

5579 \newrobustcmd\textwarning{
5580   \raisebox{-0.05cm}{\pgfuseimage{miko@dbend}}
5581   \xspace
5582 }
5583 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}
5584 \newrobustcmd\bigtextwarning{
5585   \raisebox{-0.05cm}{\pgfuseimage{miko@big@dbend}}
5586   \xspace
5587 }

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

5588 \newrobustcmd\putgraphicsat[3]{
5589   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
5590 }
5591 \newrobustcmd\putat[2]{
5592   \begin{picture}(0,0)\put(#1){#2}\end{picture}
5593 }

```

## 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`.

```

5594 \bool_if:NT \c__notesslides_sectocframes_bool {
5595   \str_if_eq:VnTF \__notesslidesstopsect{part}{
5596     \newcounter{chapter}\counterwithin*{section}{chapter}
5597   }{
5598     \str_if_eq:VnT\__notesslidesstopsect{chapter}{
5599       \newcounter{chapter}\counterwithin*{section}{chapter}
5600     }
5601   }
5602 }

```

`\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
5603 \def\part@prefix{}
5604 \@ifpackageloaded{document-structure}{
5605   \str_case:VnF \__notesslidesstopsect {
5606     {part}{
5607       \int_set:Nn \l_document_structure_section_level_int {0}
5608       \def\thesection{\arabic{chapter}.\arabic{section}}
5609       \def\part@prefix{\arabic{chapter}.}
5610     }
5611     {chapter}{
5612       \int_set:Nn \l_document_structure_section_level_int {1}
5613       \def\thesection{\arabic{chapter}.\arabic{section}}
5614       \def\part@prefix{\arabic{chapter}.}
5615     }
5616   }{
5617     \int_set:Nn \l_document_structure_section_level_int {2}
5618     \def\part@prefix{}

```

```

5619 }
5620 }
5621
5622 \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

```

5623 \renewenvironment{omgroup}[2][]{
5624   \__document_structure_omgroup_args:n { #1 }
5625   \int_incr:N \l_document_structure_omgroup_level_int
5626   \int_incr:N \l_document_structure_section_level_int
5627   \bool_if:NT \c__notesslides_sectocframes_bool {
5628     \stepcounter{slide}
5629     \begin{frame}[noframenumbering]
5630     \vfill\Large\centering
5631     \red{
5632       \ifcase\l_document_structure_section_level_int\or
5633         \stepcounter{part}
5634         \def\__notesslideslabel{\omdoc@part@kw~\Roman{part}}
5635         \def\currentsectionlevel{\omdoc@part@kw}
5636       \or
5637         \stepcounter{chapter}
5638         \def\__notesslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
5639         \def\currentsectionlevel{\omdoc@chapter@kw}
5640       \or
5641         \stepcounter{section}
5642         \def\__notesslideslabel{\part@prefix\arabic{section}}
5643         \def\currentsectionlevel{\omdoc@section@kw}
5644       \or
5645         \stepcounter{subsection}
5646         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
5647         \def\currentsectionlevel{\omdoc@subsection@kw}
5648       \or
5649         \stepcounter{subsubsection}
5650         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
5651         \def\currentsectionlevel{\omdoc@subsubsection@kw}
5652       \or
5653         \stepcounter{paragraph}
5654         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
5655         \def\currentsectionlevel{\omdoc@paragraph@kw}
5656       \else
5657         \def\__notesslideslabel{}
5658         \def\currentsectionlevel{\omdoc@paragraph@kw}
5659       \fi% end ifcase
5660       \__notesslideslabel%\sref@label@id\__notesslideslabel
5661       \quad #2%
5662     }%
5663     \vfill%
5664     \end{frame}%
5665   }
5666   \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str%

```

```

5667 }{}
5668 }

```

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

```

5669 \def\inserttheorembodyfont{\normalfont}
5670 %\bool_if:NF \c__notesslides_notes_bool {
5671 % \defbeamertemplate{theorem begin}{miko}
5672 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
5673 % \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
5674 % \inserttheorempunctuation\inserttheorembodyfont\hspace}
5675 % \defbeamertemplate{theorem end}{miko}{}}

```

and we set it as the default one.

```

5676 % \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.

```

5677 % \expandafter\def\csname Parent2\endcsname{}
5678 %}
5679
5680 \AddToHook{begindocument}{% this does not work for some reasons
5681 \setbeamertemplate{theorems}[ams style]
5682 }
5683 \bool_if:NT \c__notesslides_notes_bool {
5684 \renewenvironment{columns}[1][{}]{%
5685 \par\noindent%
5686 \begin{minipage}%
5687 \slidewidth\centering\leavevmode%
5688 }{}%
5689 \end{minipage}\par\noindent%
5690 }%
5691 \newsavebox\columnbox%
5692 \renewenvironment<>{column}[2][{}]{%
5693 \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
5694 }{}%
5695 \end{minipage}\end{lrbox}\usebox\columnbox%
5696 }%
5697 }
5698 \bool_if:NTF \c__notesslides_noproblems_bool {
5699 \newenvironment{problems}{}{}
5700 }{
5701 \excludecomment{problems}
5702 }

```

## 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.

```

5703 \gdef\printexcursions{}
5704 \newcommand\excursionref[2]{% label, text
5705 \bool_if:NT \c__notesslides_notes_bool {

```

```

5706     \begin{sparagraph}[title=Excursion]
5707     #2 \sref[fallback=the appendix]{#1}.
5708     \end{sparagraph}
5709   }
5710 }
5711 \newcommand\activate@excursion[2][]{
5712   \gappto\printexcursions{\inputref{#1}{#2}}
5713 }
5714 \newcommand\excursion[4][]{% repos, label, path, text
5715   \bool_if:NT \c__notesslides_notes_bool {
5716     \activate@excursion[1]{#3}\excursionref{#2}{#4}
5717   }
5718 }

```

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

\excursiongroup

```

5719 \keys_define:nn{notesslides / excursiongroup }{
5720   id          .str_set_x:N = \l__notesslides_excursion_id_str,
5721   intro       .tl_set:N   = \l__notesslides_excursion_intro_tl,
5722   mhrepos     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
5723 }
5724 \cs_new_protected:Nn \__notesslides_excursion_args:n {
5725   \tl_clear:N \l__notesslides_excursion_intro_tl
5726   \str_clear:N \l__notesslides_excursion_id_str
5727   \str_clear:N \l__notesslides_excursion_mhrepos_str
5728   \keys_set:nn {notesslides / excursiongroup }{ #1 }
5729 }
5730 \newcommand\excursiongroup[1][]{
5731   \__notesslides_excursion_args:n{ #1 }
5732   \ifdefempty\printexcursions{}% only if there are excursions
5733   {\begin{note}
5734     \begin{omgroup}[1]{Excursions}%
5735     \ifdefempty\l__notesslides_excursion_intro_tl{\{
5736       \inputref[\l__notesslides_excursion_mhrepos_str]{
5737         \l__notesslides_excursion_intro_tl
5738       }
5739     }
5740     \printexcursions%
5741     \end{omgroup}
5742   }\end{note}}
5743 }
5744 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{\fi
5745 \</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.

```
5746 <*package>
5747 <@@=problems>
5748 \ProvidesExplPackage{problem}{2019/03/20}{1.3}{Semantic Markup for Problems}
5749 \RequirePackage{l3keys2e,expl-keystr-compat}
5750
5751 \keys_define:nn { problem / pkg }{
5752   notes      .default:n    = { true },
5753   notes      .bool_set:N   = \c__problems_notes_bool,
5754   gnotes     .default:n    = { true },
5755   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
5756   hints      .default:n    = { true },
5757   hints      .bool_set:N   = \c__problems_hints_bool,
5758   solutions  .default:n    = { true },
5759   solutions  .bool_set:N   = \c__problems_solutions_bool,
5760   pts        .default:n    = { true },
5761   pts        .bool_set:N   = \c__problems_pts_bool,
5762   min        .default:n    = { true },
5763   min        .bool_set:N   = \c__problems_min_bool,
5764   boxed      .default:n    = { true },
5765   boxed      .bool_set:N   = \c__problems_boxed_bool,
5766   unknown    .code:n       = {}
5767 }
5768 \newif\ifsolutions
5769
5770 \ProcessKeysOptions{ problem / pkg }
5771 \bool_if:NTF \c__problems_solutions_bool {
5772   \solutionstrue
5773 }{
5774   \solutionsfalse
5775 }
```

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



```
5776 \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.

```
5777 \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.

```
5778 \def\prob@problem@kw{Problem}
5779 \def\prob@solution@kw{Solution}
5780 \def\prob@hint@kw{Hint}
5781 \def\prob@note@kw{Note}
5782 \def\prob@gnote@kw{Grading}
5783 \def\prob@pt@kw{pt}
5784 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
5785 \AddToHook{begindocument}{
5786   \ltx@ifpackageloaded{babel}{
5787     \makeatletter
5788     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5789     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5790       \input{problem-ngerman.ldf}
5791     }
5792     \clist_if_in:NnT \l_tmpa_clist {finnish}{
5793       \input{problem-finnish.ldf}
5794     }
5795     \clist_if_in:NnT \l_tmpa_clist {french}{
5796       \input{problem-french.ldf}
5797     }
5798     \clist_if_in:NnT \l_tmpa_clist {russian}{
5799       \input{problem-russian.ldf}
5800     }
5801     \makeatother
5802   }{}
5803 }
```

## 40.2 Problems and Solutions

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

```
5804 \keys_define:nn{ problem / problem }{
5805   id      .str_set:x:N = \l__problems_prob_id_str,
5806   pts     .tl_set:N    = \l__problems_prob_pts_tl,
5807   min     .tl_set:N    = \l__problems_prob_min_tl,
5808   title   .tl_set:N    = \l__problems_prob_title_tl,
5809   type    .tl_set:N    = \l__problems_prob_type_tl,
5810   refnum  .int_set:N   = \l__problems_prob_refnum_int
5811 }
5812 \cs_new_protected:Nn \__problems_prob_args:n {
```

```

5813 \str_clear:N \l__problems_prob_id_str
5814 \tl_clear:N \l__problems_prob_pts_tl
5815 \tl_clear:N \l__problems_prob_min_tl
5816 \tl_clear:N \l__problems_prob_title_tl
5817 \tl_clear:N \l__problems_prob_type_tl
5818 \int_zero_new:N \l__problems_prob_refnum_int
5819 \keys_set:nn { problem / problem }{ #1 }
5820 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
5821   \let\l__problems_prob_refnum_int\undefined
5822 }
5823 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

5824 \newcounter{problem}
5825 \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.

```

5826 \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

```

5827 \newcommand\prob@number{
5828   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
5829     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
5830   }{
5831     \int_if_exist:NTF \l__problems_prob_refnum_int {
5832       \prob@label{\int_use:N \l__problems_prob_refnum_int }
5833     }{
5834       \prob@label\theproblem
5835     }
5836   }
5837 }

```

(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.

```

5838 \newcommand\prob@title[3]{%
5839   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
5840     #2 \l__problems_inclprob_title_tl #3
5841   }{
5842     \tl_if_exist:NTF \l__problems_prob_title_tl {
5843       #2 \l__problems_prob_title_tl #3
5844     }{
5845       #1
5846     }
5847   }
5848 }

```

(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.

```
5849 \def\prob@heading{
5850   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
5851   %\sref@label{id}\prob@problem@kw~\prob@number}{~}
5852 }
```

(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`

```
5853 \newenvironment{sproblem}[1][]{
5854   \__problems_prob_args:n{#1}%\sref@target%
5855   \@in@omtexttrue% we are in a statement (for inline definitions)
5856   \stepcounter{problem}\record@problem
5857   \def\current@section@level{\prob@problem@kw}
5858   \tl_if_exist:NTF \l__problems_inclprob_type_tl {
5859     \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
5860   }{
5861     \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
5862   }
5863   \str_if_exist:NTF \l__problems_inclprob_id_str {
5864     \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
5865   }{
5866     \str_set_eq:NN \sproblemid \l__problems_prob_id_str
5867   }
5868
5869
5870   \clist_set:No \l_tmpa_clist \sproblemtype
5871   \tl_clear:N \l_tmpa_tl
5872   \clist_map_inline:Nn \l_tmpa_clist {
5873     \tl_if_exist:cT {\__problems_sproblem_##1_start:}{
5874       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_start:}}
5875     }
5876   }
5877   \tl_if_empty:NTF \l_tmpa_tl {
5878     \__problems_sproblem_start:
5879   }{
5880     \l_tmpa_tl
5881   }
5882   \stex_ref_new_doc_target:n \sproblemid
5883 }{
5884   \clist_set:No \l_tmpa_clist \sproblemtype
5885   \tl_clear:N \l_tmpa_tl
5886   \clist_map_inline:Nn \l_tmpa_clist {
5887     \tl_if_exist:cT {\__problems_sproblem_##1_end:}{
5888       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_end:}}
5889     }
5890   }
```

```

5890 }
5891 \tl_if_empty:NTF \l_tmpa_tl {
5892   \__problems_sproblem_end:
5893 }{
5894   \l_tmpa_tl
5895 }
5896
5897
5898 \smallskip
5899 }
5900
5901
5902 \cs_new_protected:Nn \__problems_sproblem_start: {
5903   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
5904 }
5905 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
5906
5907 \newcommand\stexpatchproblem[3][] {
5908   \str_set:Nx \l_tmpa_str{ #1 }
5909   \str_if_empty:NTF \l_tmpa_str {
5910     \tl_set:Nn \__problems_sproblem_start: { #2 }
5911     \tl_set:Nn \__problems_sproblem_end: { #3 }
5912   }{
5913     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
5914     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
5915   }
5916 }
5917
5918
5919 \bool_if:NT \c__problems_boxed_bool {
5920   \surroundwithmdframed{problem}
5921 }

```

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

```

5922 \def\record@problem{
5923   \protected@write\@auxout{}
5924   {
5925     \string\@problem{\prob@number}
5926     {
5927       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
5928         \l__problems_inclprob_pts_tl
5929       }{
5930         \l__problems_prob_pts_tl
5931       }
5932     }%
5933     {
5934       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
5935         \l__problems_inclprob_min_tl
5936       }{
5937         \l__problems_prob_min_tl
5938       }
5939     }
5940   }
5941 }

```

(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).

```
5942 \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.

```
5943 \keys_define:nn { problem / solution }{
5944   id                .str_set_x:N = \l__problems_solution_id_str ,
5945   for               .tl_set:N   = \l__problems_solution_for_tl ,
5946   height            .dim_set:N   = \l__problems_solution_height_dim ,
5947   creators          .clist_set:N = \l__problems_solution_creators_clist ,
5948   contributors      .clist_set:N = \l__problems_solution_contributors_clist ,
5949   srccite           .tl_set:N    = \l__problems_solution_srccite_tl
5950 }
5951 \cs_new_protected:Nn \__problems_solution_args:n {
5952   \str_clear:N \l__problems_solution_id_str
5953   \tl_clear:N \l__problems_solution_for_tl
5954   \tl_clear:N \l__problems_solution_srccite_tl
5955   \clist_clear:N \l__problems_solution_creators_clist
5956   \clist_clear:N \l__problems_solution_contributors_clist
5957   \dim_zero:N \l__problems_solution_height_dim
5958   \keys_set:nn { problem / solution }{ #1 }
5959 }
```

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

```
5960 \newcommand\@startsolution[1][{}]{
5961   \__problems_solution_args:n { #1 }
5962   \@in@omtexttrue% we are in a statement.
5963   \bool_if:NF \c__problems_boxed_bool { \hrule }
5964   \smallskip\noindent
5965   {\textbf\prob@solution@kw : \enspace}
5966   \begin{small}
5967   \def\current@section@level{\prob@solution@kw}
5968   \ignorespacesandpars
5969 }
```

**\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.

```
5970 \newcommand\startsolutions{
5971   \specialcomment{solution}{\@startsolution}{
5972     \bool_if:NF \c__problems_boxed_bool {
5973       \hrule\medskip
5974     }
5975     \end{small}%
5976   }
5977   \bool_if:NT \c__problems_boxed_bool {
5978     \surroundwithmdframed{solution}
5979   }
5980 }
```

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

\stopsolutions

```
5981 \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.

```
5982 \ifsolutions
5983 \startsolutions
5984 \else
5985 \stopsolutions
5986 \fi
```

exnote

```
5987 \bool_if:NTF \c__problems_notes_bool {
5988 \newenvironment{exnote}[1][]{
5989 \par\smallskip\hrule\smallskip
5990 \noindent\textbf{\prob@note@kw : }\small
5991 }{
5992 \smallskip\hrule
5993 }
5994 }{
5995 \excludecomment{exnote}
5996 }
```

hint

```
5997 \bool_if:NTF \c__problems_notes_bool {
5998 \newenvironment{hint}[1][]{
5999 \par\smallskip\hrule\smallskip
6000 \noindent\textbf{\prob@hint@kw :~ }\small
6001 }{
6002 \smallskip\hrule
6003 }
6004 \newenvironment{exhint}[1][]{
6005 \par\smallskip\hrule\smallskip
6006 \noindent\textbf{\prob@hint@kw :~ }\small
6007 }{
6008 \smallskip\hrule
6009 }
6010 }{
6011 \excludecomment{hint}
6012 \excludecomment{exhint}
6013 }
```

gnote

```
6014 \bool_if:NTF \c__problems_notes_bool {
6015 \newenvironment{gnote}[1][]{
6016 \par\smallskip\hrule\smallskip
6017 \noindent\textbf{\prob@gnote@kw : }\small
6018 }{
6019 \smallskip\hrule
6020 }
6021 }{
6022 \excludecomment{gnote}
6023 }
```

## 40.3 Multiple Choice Blocks

```

6024 \newenvironment{mcb}{
6025   \begin{enumerate}
6026 }{
6027   \end{enumerate}
6028 }

```

we define the keys for the mcc macro

```

6029 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
6030   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
6031     \bool_set_true:N #1
6032   }{
6033     \bool_set_false:N #1
6034   }
6035 }
6036 \keys_define:nn { problem / mcc }{
6037   id          .str_set_x:N = \l__problems_mcc_id_str ,
6038   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
6039   T           .default:n   = { true } ,
6040   T           .bool_set:N   = \l__problems_mcc_t_bool ,
6041   F           .default:n   = { true } ,
6042   F           .bool_set:N   = \l__problems_mcc_f_bool ,
6043   Ttext       .code:n      = {
6044     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
6045   } ,
6046   Ftext       .code:n      = {
6047     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
6048   }
6049 }
6050 \cs_new_protected:Nn \l__problems_mcc_args:n {
6051   \str_clear:N \l__problems_mcc_id_str
6052   \tl_clear:N \l__problems_mcc_feedback_tl
6053   \bool_set_true:N \l__problems_mcc_t_bool
6054   \bool_set_true:N \l__problems_mcc_f_bool
6055   \bool_set_true:N \l__problems_mcc_Ttext_bool
6056   \bool_set_false:N \l__problems_mcc_Ftext_bool
6057   \keys_set:nn { problem / mcc }{ #1 }
6058 }

```

\mcc

```

6059 \newcommand\mcc[2][] {
6060   \l__problems_mcc_args:n{ #1 }
6061   \item #2
6062   \ifsolutions
6063     \\\
6064     \bool_if:NT \l__problems_mcc_t_bool {
6065       % TODO!
6066       % \ifcsstring{mcc@T}{T}{\mcc@Ttext}%
6067     }
6068     \bool_if:NT \l__problems_mcc_f_bool {

```

---

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

```

6069      % TODO!
6070      % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
6071    }
6072    \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
6073      !
6074    }{
6075      \l__problems_mcc_feedback_tl
6076    }
6077    \fi
6078  } %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.

```

6079
6080 \keys_define:nn{ problem / inclproblem }{
6081   id      .str_set_x:N = \l__problems_inclprob_id_str,
6082   pts     .tl_set:N    = \l__problems_inclprob_pts_tl,
6083   min     .tl_set:N    = \l__problems_inclprob_min_tl,
6084   title   .tl_set:N    = \l__problems_inclprob_title_tl,
6085   refnum  .int_set:N    = \l__problems_inclprob_refnum_int,
6086   type    .tl_set:N    = \l__problems_inclprob_type_tl,
6087   mhrepos .str_set_x:N = \l__problems_inclprob_mhrepos_str
6088 }
6089 \cs_new_protected:Nn \l__problems_inclprob_args:n {
6090   \str_clear:N \l__problems_prob_id_str
6091   \tl_clear:N \l__problems_inclprob_pts_tl
6092   \tl_clear:N \l__problems_inclprob_min_tl
6093   \tl_clear:N \l__problems_inclprob_title_tl
6094   \tl_clear:N \l__problems_inclprob_type_tl
6095   \int_zero_new:N \l__problems_inclprob_refnum_int
6096   \str_clear:N \l__problems_inclprob_mhrepos_str
6097   \keys_set:nn { problem / inclproblem }{ #1 }
6098   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
6099     \let\l__problems_inclprob_pts_tl\undefined
6100   }
6101   \tl_if_empty:NT \l__problems_inclprob_min_tl {
6102     \let\l__problems_inclprob_min_tl\undefined
6103   }
6104   \tl_if_empty:NT \l__problems_inclprob_title_tl {
6105     \let\l__problems_inclprob_title_tl\undefined
6106   }
6107   \tl_if_empty:NT \l__problems_inclprob_type_tl {
6108     \let\l__problems_inclprob_type_tl\undefined
6109   }
6110   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
6111     \let\l__problems_inclprob_refnum_int\undefined
6112   }
6113 }

```



```

6114
6115 \cs_new_protected:Nn \__problems_inclprob_clear: {
6116   \let\l__problems_inclprob_id_str\undefined
6117   \let\l__problems_inclprob_pts_tl\undefined
6118   \let\l__problems_inclprob_min_tl\undefined
6119   \let\l__problems_inclprob_title_tl\undefined
6120   \let\l__problems_inclprob_type_tl\undefined
6121   \let\l__problems_inclprob_refnum_int\undefined
6122   \let\l__problems_inclprob_mhrepos_str\undefined
6123 }
6124 \__problems_inclprob_clear:
6125
6126 \newcommand\includeproblem[2][ ]{
6127   \__problems_inclprob_args:n{ #1 }
6128   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
6129     \input{#2}
6130   }{
6131     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
6132       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
6133     }
6134   }
6135   \__problems_inclprob_clear:
6136 }

```

(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.

```

6137 \AddToHook{enddocument}{
6138   \bool_if:NT \c__problems_pts_bool {
6139     \message{Total:~\arabic{pts}~points}
6140   }
6141   \bool_if:NT \c__problems_min_bool {
6142     \message{Total:~\arabic{min}~minutes}
6143   }
6144 }

```

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

```

6145 \def\pts#1{
6146   \bool_if:NT \c__problems_pts_bool {
6147     \marginpar{#1~\prob@pt@kw}
6148   }
6149 }
6150 \def\min#1{
6151   \bool_if:NT \c__problems_min_bool {
6152     \marginpar{#1~\prob@min@kw}
6153   }
6154 }

```

`\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.

```

6155 \newcounter{pts}
6156 \def\show@pts{
6157   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
6158     \bool_if:NT \c__problems_pts_bool {
6159       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
6160       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
6161     }
6162   }{
6163     \tl_if_exist:NT \l__problems_prob_pts_tl {
6164       \bool_if:NT \c__problems_pts_bool {
6165         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
6166         \addtocounter{pts}{\l__problems_prob_pts_tl}
6167       }
6168     }
6169   }
6170 }

```

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

`\show@min`

```

6171 \newcounter{min}
6172 \def\show@min{
6173   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
6174     \bool_if:NT \c__problems_min_bool {
6175       \marginpar{\l__problems_inclprob_min_tl\ min}
6176       \addtocounter{min}{\l__problems_inclprob_min_tl}
6177     }
6178   }{
6179     \tl_if_exist:NT \l__problems_prob_min_tl {
6180       \bool_if:NT \c__problems_min_bool {
6181         \marginpar{\l__problems_prob_min_tl\ min}
6182         \addtocounter{min}{\l__problems_prob_min_tl}
6183       }
6184     }
6185   }
6186 }
6187 \</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.

```
6188 <@@=hwexam>
6189 <*cls>
6190 \ProvidesExplClass{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
6191 \RequirePackage{l3keys2e,expl-keystr-compatible}
6192 \DeclareOption*{
6193   \PassOptionsToClass{\CurrentOption}{document-structure}
6194   \PassOptionsToPackage{\CurrentOption}{stex}
6195   \PassOptionsToPackage{\CurrentOption}{hwexam}
6196   \PassOptionsToPackage{\CurrentOption}{tikzinput}
6197 }
6198 \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.

```
6199 \LoadClass{document-structure}
6200 \RequirePackage{stex}
6201 \RequirePackage{hwexam}
6202 \RequirePackage{tikzinput}
6203 \RequirePackage{graphicx}
6204 \RequirePackage{a4wide}
6205 \RequirePackage{amssymb}
6206 \RequirePackage{amstext}
6207 \RequirePackage{amsmath}
```

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

```

6208 \newcommand\assig@default@type{\hwexam@assignment@kw}
6209 \def\document@hwexamtype{\assig@default@type}
6210 <@@=document_structure>
6211 \keys_define:nn { document-structure / document }{
6212 id .str_set_x:N = \c_document_structure_document_id_str,
6213 hwexamtype .tl_set:N = \document@hwexamtype
6214 }
6215 <@@=hwexam>
6216 </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.

```
6217 \*package>
6218 \ProvidesExplPackage{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
6219 \RequirePackage{l3keys2e,expl-keystr-compat}
6220
6221 \newif\iftest\testfalse
6222 \DeclareOption{test}{\testtrue}
6223 \newif\ifmultiple\multiplefalse
6224 \DeclareOption{multiple}{\multipletrue}
6225 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
6226 \ProcessOptions
```

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

```
6227 \RequirePackage{keyval}[1997/11/10]
6228 \RequirePackage{problem}
```

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

```
6229 \newcommand\hwexam@assignment@kw{Assignment}
6230 \newcommand\hwexam@given@kw{Given}
6231 \newcommand\hwexam@due@kw{Due}
6232 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
6233 blank~for~extra~space}
6234 \def\hwexam@minutes@kw{minutes}
6235 \newcommand\correction@probs@kw{prob.}
6236 \newcommand\correction@pts@kw{total}
6237 \newcommand\correction@reached@kw{reached}
6238 \newcommand\correction@sum@kw{Sum}
6239 \newcommand\correction@grade@kw{grade}
6240 \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

```

6241 \AddToHook{begindocument}{
6242 \ltx@ifpackageloaded{babel}{
6243 \makeatletter
6244 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6245 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6246 \input{hwexam-ngerman.ldf}
6247 }
6248 \clist_if_in:NnT \l_tmpa_clist {finnish}{
6249 \input{hwexam-finnish.ldf}
6250 }
6251 \clist_if_in:NnT \l_tmpa_clist {french}{
6252 \input{hwexam-french.ldf}
6253 }
6254 \clist_if_in:NnT \l_tmpa_clist {russian}{
6255 \input{hwexam-russian.ldf}
6256 }
6257 \makeatother
6258 }{}
6259 }
6260

```

## 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.

```

6261 \newcounter{assignment}
6262 \numberproblemsin{assignment}
6263 \renewcommand\prob@label[1]{\assignment@number.#1}

```

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

```

6264 \keys_define:nn { hwexam / assignment } {
6265 id .str_set:N = \l__hwexam_assign_id_str,
6266 number .int_set:N = \l__hwexam_assign_number_int,
6267 title .tl_set:N = \l__hwexam_assign_title_tl,
6268 type .tl_set:N = \l__hwexam_assign_type_tl,
6269 given .tl_set:N = \l__hwexam_assign_given_tl,
6270 due .tl_set:N = \l__hwexam_assign_due_tl,
6271 loadmodules .code:n = {
6272 \bool_set_true:N \l__hwexam_assign_loadmodules_bool
6273 }
6274 }
6275 \cs_new_protected:Nn \__hwexam_assignment_args:n {
6276 \str_clear:N \l__hwexam_assign_id_str
6277 \int_set:Nn \l__hwexam_assign_number_int {-1}
6278 \tl_clear:N \l__hwexam_assign_title_tl
6279 \tl_clear:N \l__hwexam_assign_type_tl
6280 \tl_clear:N \l__hwexam_assign_given_tl
6281 \tl_clear:N \l__hwexam_assign_due_tl
6282 \bool_set_false:N \l__hwexam_assign_loadmodules_bool

```

```

6283 \keys_set:nn { hwexam / assignment }{ #1 }
6284 }

```

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.

```

6285 \newcommand\given@due[2]{
6286 \bool_lazy_all:nF {
6287 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
6288 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
6289 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
6290 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
6291 }{ #1 }
6292
6293 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
6294 \tl_if_empty:NF \l__hwexam_assign_given_tl {
6295 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
6296 }
6297 }{
6298 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
6299 }
6300
6301 \bool_lazy_or:nnF {
6302 \bool_lazy_and_p:nn {
6303 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
6304 }{
6305 \tl_if_empty_p:V \l__hwexam_assign_due_tl
6306 }
6307 }{
6308 \bool_lazy_and_p:nn {
6309 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
6310 }{
6311 \tl_if_empty_p:V \l__hwexam_assign_due_tl
6312 }
6313 }{ ,~ }
6314
6315 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
6316 \tl_if_empty:NF \l__hwexam_assign_due_tl {
6317 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
6318 }
6319 }{
6320 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
6321 }
6322
6323 \bool_lazy_all:nF {
6324 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
6325 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
6326 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
6327 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
6328 }{ #2 }
6329 }

```

`\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.

```

6330 \newcommand\assignment@title[3]{
6331 \tl_if_empty:NTF \l__hwexam_inclasssign_title_tl {
6332 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
6333 #1
6334 }{
6335 #2\l__hwexam_assign_title_tl#3
6336 }
6337 }{
6338 #2\l__hwexam_inclasssign_title_tl#3
6339 }
6340 }

```

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

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

```

6341 \newcommand\assignment@number{
6342 \int_compare:nNnTF \l__hwexam_inclasssign_number_int = {-1} {
6343 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
6344 \arabic{assignment}
6345 } {
6346 \int_use:N \l__hwexam_assign_number_int
6347 }
6348 }{
6349 \int_use:N \l__hwexam_inclasssign_number_int
6350 }
6351 }

```

(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.

```

6352 \newenvironment{assignment}[1][ ]{
6353 \__hwexam_assignment_args:n { #1 }
6354 %\sref@target
6355 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
6356 \global\stepcounter{assignment}
6357 }{
6358 \global\setcounter{assignment}{\int_use:N\l__hwexam_assign_number_int}
6359 }
6360 \setcounter{problem}{0}
6361 \def\current@section@level{\document@hwexamtype}
6362 %\sref@label@id{\document@hwexamtype \thesection}
6363 \begin{@assignment}
6364 }{
6365 \end{@assignment}
6366 }

```



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

```

6367 \def\ass@title{
6368 \protect\document@hwexamtype~\arabic{assignment}
6369 \assignment@title{}\{;\}\{;\} -- \given@due{}\{;\}
6370 }
6371 \ifmultiple
6372 \newenvironment{@assignment}{
6373 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
6374 \begin{omgroup}[loadmodules]{\ass@title}
6375 }{
6376 \begin{omgroup}{\ass@title}
6377 }
6378 }{
6379 \end{omgroup}
6380 }

```

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

```

6381 \else
6382 \newenvironment{@assignment}{
6383 \begin{center}\bf
6384 \Large@title\strut\
6385 \document@hwexamtype~\arabic{assignment}\assignment@title{}\{;\}\{;\}\{;\}
6386 \large\given@due{--;\}\{;\}--}
6387 \end{center}
6388 }{}
6389 \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.

```

6390 \keys_define:nn { hwexam / inclassignment } {
6391 %id .str_set_x:N = \l__hwexam_assign_id_str,
6392 number .int_set:N = \l__hwexam_inclassign_number_int,
6393 title .tl_set:N = \l__hwexam_inclassign_title_tl,
6394 type .tl_set:N = \l__hwexam_inclassign_type_tl,
6395 given .tl_set:N = \l__hwexam_inclassign_given_tl,
6396 due .tl_set:N = \l__hwexam_inclassign_due_tl,
6397 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
6398 }
6399 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
6400 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
6401 \tl_clear:N \l__hwexam_inclassign_title_tl
6402 \tl_clear:N \l__hwexam_inclassign_type_tl
6403 \tl_clear:N \l__hwexam_inclassign_given_tl
6404 \tl_clear:N \l__hwexam_inclassign_due_tl
6405 \str_clear:N \l__hwexam_inclassign_mhrepos_str
6406 \keys_set:nn { hwexam / inclassignment }{ #1 }
6407 }
6408 \__hwexam_inclassignment_args:n {}
6409
6410 \newcommand\inputassignment[2][{}]{

```

```

6411 \_hwexam_inclassnment_args:n { #1 }
6412 \str_if_empty:NTF \l__hwexam_inclassn_mhrepos_str {
6413   \input{#2}
6414 }{
6415   \stex_in_repository:nn{\l__hwexam_inclassn_mhrepos_str}{
6416     \input{\mhp{path}\l__hwexam_inclassn_mhrepos_str}{#2}}
6417   }
6418 }
6419 \_hwexam_inclassnment_args:n {}
6420 }
6421 \newcommand\includeassignment[2][ ]{
6422   \newpage
6423   \inputassignment[#1]{#2}
6424 }

```

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

## 42.4 Typesetting Exams

\quizheading

```

6425 \ExplSyntaxOff
6426 \newcommand\quizheading[1]{%
6427   \def\@tas{#1}%
6428   \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
6429   \ifx\@tas\@empty\else%
6430     \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
6431   \fi%
6432 }
6433 \ExplSyntaxOn

```

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

\testheading

```

6434
6435 \def\hwexamheader{\input{hwexam-default.header}}
6436
6437 \def\hwexamminutes{
6438   \tl_if_empty:NTF \testheading@duration {
6439     {\testheading@min}~\hwexam@minutes@kw
6440   }{
6441     \testheading@duration
6442   }
6443 }
6444
6445 \keys_define:nn { hwexam / testheading } {
6446   min .tl_set:N = \testheading@min,
6447   duration .tl_set:N = \testheading@duration,
6448   reqpts .tl_set:N = \testheading@reqpts,
6449   tools .tl_set:N = \testheading@tools
6450 }
6451 \cs_new_protected:Nn \_hwexam_testheading_args:n {
6452   \tl_clear:N \testheading@min
6453   \tl_clear:N \testheading@duration

```

```

6454 \tl_clear:N \testheading@reqpts
6455 \tl_clear:N \testheading@tools
6456 \keys_set:nn { hwexam / testheading }{ #1 }
6457 }
6458 \newenvironment{testheading}[1][]{
6459   \_hwexam_testheading_args:n{ #1 }
6460   \newcount\check@time\check@time=\testheading@min
6461   \advance\check@time by -\theassignment@totalmin
6462   \newif\if@bonuspoints
6463   \tl_if_empty:NTF \testheading@reqpts {
6464     \@bonuspointsfalse
6465   }{
6466     \newcount\bonus@pts
6467     \bonus@pts=\theassignment@totalpts
6468     \advance\bonus@pts by -\testheading@reqpts
6469     \edef\bonus@pts{\the\bonus@pts}
6470     \@bonuspointstrue
6471   }
6472   \edef\check@time{\the\check@time}
6473
6474   \makeatletter\hwexamheader\makeatother
6475 }{
6476   \newpage
6477 }

```

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

\testspace

```

6478 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

6479 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

6480 \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.

```

6481 <@=problems>
6482 \renewcommand\@problem[3]{
6483   \stepcounter{assignment@probs}
6484   \def\__problemspts{#2}
6485   \ifx\__problemspts\@empty\else
6486     \addtocounter{assignment@totalpts}{#2}
6487   \fi
6488   \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
6489   \xdef\correction@probs{\correction@probs & #1}%
6490   \xdef\correction@pts{\correction@pts & #2}
6491   \xdef\correction@reached{\correction@reached &}

```

```

6492 }
6493 <@@=hwexam>

```

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

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

```

6494 \newcounter{assignment@probs}
6495 \newcounter{assignment@totalpts}
6496 \newcounter{assignment@totalmin}
6497 \def\correction@probs{\correction@probs@kw}
6498 \def\correction@pts{\correction@pts@kw}
6499 \def\correction@reached{\correction@reached@kw}
6500 \stepcounter{assignment@probs}
6501 \newcommand\correction@table{
6502 \resizebox{\textwidth}{!}{%
6503 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
6504 &\multicolumn{\theassignment@probs}{c|}|%|
6505 {\footnotesize\correction@forgrading@kw} &\\ \hline
6506 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
6507 \correction@pts & \theassignment@totalpts & \\ \hline
6508 \correction@reached & & \[.7cm]\hline
6509 \end{tabular}}
6510 </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}

```