

# The sTeX3 Package \*

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

2022-02-16

## Abstract

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

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

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

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

---

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

# Contents

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

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

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

<b>25</b>	<b>STeX-Basics Implementation</b>	<b>72</b>
25.1	The STeXDocument Class . . . . .	72
25.2	Preliminaries . . . . .	72
25.3	Messages and logging . . . . .	73
25.4	Persistence . . . . .	74
25.5	HTML Annotations . . . . .	74
25.6	Languages . . . . .	77
25.7	Activating/Deactivating Macros . . . . .	78
<b>26</b>	<b>STeX-MathHub Implementation</b>	<b>80</b>
26.1	Generic Path Handling . . . . .	80
26.2	PWD and kpsewhich . . . . .	82
26.3	File Hooks and Tracking . . . . .	83
26.4	MathHub Repositories . . . . .	84
<b>27</b>	<b>STeX-References Implementation</b>	<b>92</b>
27.1	Document URIs and URLs . . . . .	92
27.2	Setting Reference Targets . . . . .	94
27.3	Using References . . . . .	95
<b>28</b>	<b>STeX-Modules Implementation</b>	<b>99</b>
28.1	The module environment . . . . .	102
28.2	Invoking modules . . . . .	108
<b>29</b>	<b>STeX-Module Inheritance Implementation</b>	<b>110</b>
29.1	SMS Mode . . . . .	110
29.2	Inheritance . . . . .	113
<b>30</b>	<b>STeX-Symbols Implementation</b>	<b>118</b>
30.1	Symbol Declarations . . . . .	118
30.2	Notations . . . . .	125
<b>31</b>	<b>STeX-Terms Implementation</b>	<b>135</b>
31.1	Symbol Invocations . . . . .	135
31.2	Terms . . . . .	138
31.3	Notation Components . . . . .	145
<b>32</b>	<b>STeX-Structural Features Implementation</b>	<b>148</b>
32.1	Imports with modification . . . . .	148
32.2	The feature environment . . . . .	155
32.3	Features . . . . .	157
<b>33</b>	<b>STeX-Statements Implementation</b>	<b>162</b>
33.1	Definitions . . . . .	162
33.2	Assertions . . . . .	167
33.3	Examples . . . . .	170
33.4	Logical Paragraphs . . . . .	172

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

**Part I**  
**Manual**

# Chapter 1

## What is sTeX?

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

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

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

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

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

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



# Chapter 2

## Quickstart

### 2.1 Setup

#### 2.1.1 The sTeX IDE

TODO: VSCode Plugin

#### 2.1.2 Manual Setup

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

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

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

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

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

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

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

---

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

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

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

## 2.2 A First $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$ Document

Having set everything up, we can write a first  $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$  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}  $\sum_{n=1}^{\infty} \frac{1}{2^n}$ 
\realdive[\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  $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$  *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}`),  $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$  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}`.

$\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$  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. `\realdive` and `\realpower`.

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

---

`\symref`  
`\symname`

---

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

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

---

`\importmodule`

---

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

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

## Chapter 3

# Using Semantic Macros

TODO

## Chapter 4

# TeX Archives

### 4.1 The Local MathHub-Directory

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

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

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

### 4.2 The Structure of TeX Archives

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

Each such archive needs two subdirectories:

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

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

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

### 4.3 MANIFEST.MF-Files

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

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

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

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

`source-base` or

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

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

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

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

## Chapter 5

# Creating New Modules and Symbols

TODO

### Example 1

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

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

## 5.1 Advanced Structuring Mechanisms

Given modules:

### Example 2

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

Module 2:  
Module 3:  
Module 4:

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

### Example 3

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

Module 5:

Test:  $a \circ a$

TODO: explain donotclone

### Example 4

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

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

Module 6:

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



## Chapter 6

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

## Chapter 7

# Additional Packages

**7.1** Modular Document Structuring

**7.2** Slides and Course Notes

**7.3** Homework, Problems and Exams

# Chapter 8

# Stuff

## 8.1 Modules

---

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

---

### 8.1.1 Semantic Macros and Notations

Semantic macros invoke a formally declared symbol.

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

**Module 7:** For example, to introduce binary multiplication, we can do `\symdecl[args=2]{mult}`. We can then supply the semantic macro with arbitrarily many notations, such as `\notation{mult}{#1 #2}`.

#### Example 5

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

$ab$

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

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

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

### Example 6

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

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

.

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

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

### Example 7

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

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

.

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

### Example 8

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

Multiplying again by  $b$  yields...

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

### Example 9

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

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

<sup>4</sup>EdNOTE: TODO

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

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

### Example 10

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

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

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

### Example 11

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

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

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

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

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

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

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

## Other Argument Types

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

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

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

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

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

**Module 8:** b-type arguments are indistinguishable from i-type arguments within  $\text{\TeX}$ , but are treated very differently in OMDoc and by MMT. More interesting *within*  $\text{\TeX}$  are a-type arguments, which represent (associative) arguments of flexible arity, which are provided as comma-separated lists. This allows e.g. better representing the `\mult`-macro above:

### Example 12

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

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

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

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

### Example 13

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

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

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

5 6

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

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

## Precedences

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

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

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

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

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

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

For example:

**Module 9:**

### Example 14

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

---

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

---

---

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

---

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

---

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

---

---

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

---

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

---

<code>\stex_annotate:nnn</code>	<code>\stex_annotate:nnn {&lt;property&gt;} {&lt;resource&gt;} {&lt;content&gt;}</code>
<code>\stex_annotate_invisible:nnn</code>	
<code>\stex_annotate_invisible:n</code>	

---

Annotates the HTML generated by `<content>` with

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

---

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

---

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

---

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

---

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

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

---

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

---

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

# Chapter 10

## sTEX-MathHub

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

### 10.1 Macros and Environments

---

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

---

#### 10.1.1 Files, Paths, URIs

---

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

---

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

---

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

---

---

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

---

---

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

---

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

---

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

---

---

`\g_stex_currentfile_seq`

---

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

**Test 1**

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

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

**10.1.2 MathHub Archives**

---

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

---

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

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

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

---

`\l_stex_current_repository_prop`

---

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

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

## Test 2

```

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

```

```

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

```

## Chapter 11

# sTeX-References

Code related to links and cross-references

### 11.1 Macros and Environments

# Chapter 12

## sTeX-Modules

Code related to Modules

### 12.1 Macros and Environments

---

---

`\l_stex_current_module_str`

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

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

Additionally, it stores:

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

---

---

`\l_stex_all_modules_seq`

Stores full URIs for all modules currently in scope.



---

```
\g_stex_module_files_prop
\g_stex_modules_in_file_seq
```

---

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

---

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

---



---

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

---

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

---

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

---

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

---

```
\stex_add_constant_to_current_module:n
```

---

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

---

```
\stex_add_import_to_current_module:n
```

---

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

---

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

---

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

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

---

```
\stex_modules_current_namespace:
```

---

Computes the current namespace

### Test 3

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

```

Namespace 1:
file://stextest
Faking a repository:
Namespace 2:
http://mathhub.info/tests/Foo/Bar/test/stextest

```

### 12.1.1 The module-environment

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

---

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

---

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

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

### Test 4

```

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

```

```

Module 10:  Module path: http://mathhub.info/tests/Foo/Bar?Foo
Language:
Signature:
Metatheory:

```

## Test 5

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

**Module 11: FooBar** Module path: <http://mathhub.info/tests/Foo/Bar/Foo?Bar>  
Language:  
Signature:  
Metatheory:

---

**\STEXModule**    \STEXModule {<fragment>}

---

Attempts to find a module whose URI ends with <fragment> in the current scope and passes the full URI on to \stex\_invoke\_module:n.

---

**\stex\_invoke\_module:n**

---

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

## Test 6

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

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

---

`\stex_activate_module:n`

---

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

# Chapter 13

## STEX-Module Inheritance

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

### 13.1 Macros and Environments

#### 13.1.1 SMS Mode

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

---

`\g_stex_smsmode_allowedmacros_tl`

---

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

---

`\g_stex_smsmode_allowedmacros_escape_tl`

---

Macros that are executed with the category codes restored.

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

---

`\g_stex_smsmode_allowedenvs_seq`

---

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

---

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

---

Tests whether SMS mode is currently active.

---

`\stex_smsmode_set_codes:`

---

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

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

---

**`\stex_in_smsmode:nn`**

---

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

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

#### Test 7

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

### 13.1.2 Imports and Inheritance

---

**`\importmodule`**

---

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

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

#### Test 8

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

```
Module 15:    Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?Foo?foo}<
              Meaning: >macro:->\protect \bar <
Module 16:    Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?Foo?foo}<
Module 17:    Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?Foo?foo}<
```

---

**`\usemodule`**

---

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

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

## Test 9

```

\begin{smodule}{UseTest1}
\symdecl{foo}
\end{smodule}
\begin{smodule}{UseTest2}
\usemodule{UseTest1}
\symdecl{bar}
Meaning:- \present\foo\\
\end{smodule}
\begin{smodule}{UseTest3}
\importmodule{UseTest2}
Meaning:- \present\foo\\
Meaning:- \present\bar\\

All modules: \ExplSyntaxOn
\seq_use:Nn \l_stex_all_modules_seq {,-} \\
All-symbols:-
\seq_use:Nn \l_stex_all_symbols_seq {,-}
\ExplSyntaxOff
\end{smodule}

```

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

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

## Test 10

```

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

```

Circular dependencies:  
Module 21:      >macro:->\stex\_invoke\_symbol:n {http://mathhub.info/tests/Foo/Bar/circular1?Circular1?fooA}<  
>macro:->\stex\_invoke\_symbol:n {http://mathhub.info/tests/Bar/Foo/circular2?Circular2?fooB}<

---

---

`\stex_import_module_uri:nn`

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

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

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

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

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

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

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

2. Otherwise:

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

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

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

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

---

---

`\stex_import_require_module:nnnn`

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

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

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



# Chapter 14

## TEX-Symbols

Code related to symbol declarations and notations

### 14.1 Macros and Environments

---

<u><code>\symdecl</code></u>	<code>\symdecl[<math>\langle args \rangle</math>]{<math>\langle macroname \rangle</math>}</code>
------------------------------	--

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

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

---

---

`\stex_symdecl_do:n`

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

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

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

### Test 11

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

```
Module 22:      Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?SymdeclTest?foo}<
Result: file://stextest?SymdeclTest?foo
Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?SymdeclTest?bardef}<
```

---

---

`\l_stex_all_symbols_seq`

Stores full URIs for all modules currently in scope.

---

---

`\stex_get_symbol:n`

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

---

---

`\notation`

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

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

---

---

`\stex_notation_do:nn`

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

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

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

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

### Test 12

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

Module 23:

---

---

`\symdef`

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

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

### Test 13

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

Module 24:  $a + b + c$

# Chapter 15

## STEX-Terms

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

### 15.1 Macros and Environments

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

---

`\infprec`  
`\neginfprec`

---

Maximal and minimal notation precedences.

---

`\dobrackets`

---

`\dobrackets {<body>}`

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

---

`\withbrackets`

---

`\withbrackets <left> <right> {<body>}`

Temporarily (i.e. within  $\langle body \rangle$ ) sets the brackets used by  $\text{\TeX}$  for automated bracketing (by default ( and )) to  $\langle left \rangle$  and  $\langle right \rangle$ .

Note that  $\langle left \rangle$  and  $\langle right \rangle$  need to be allowed after `\left` and `\right` in display-mode.

### Test 14

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

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

### Test 15

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

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

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

$a + (b \cdot c)$  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 27:**  
some a and some b and also some c here.  
some *a* and some *b* and also some *c* here.  
bar  
or just some c  
bar  
or first b, then c, and finally a

---

`\stex_highlight_term:nn`

---

`\stex_highlight_term:nn{<URI>}{<args>}`

Establishes a context for `\comp`. Stores the URI in a variable so that `\comp` knows which symbol governs the current notation.

---

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

---

`\comp{<args>}`

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

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

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

---

`\STEXinvisible`

---

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

---

`\ellipses`

---

TODO

## Chapter 16

# TeX-Structural Features

Code related to structural features

### 16.1 Macros and Environments

#### 16.1.1 Structures

`mathstructure` TODO

## Chapter 17

# TeX-Statements

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

### 17.1 Macros and Environments

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



## Chapter 18

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

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

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

## Contents

## 18.1 Introduction

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

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

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

Example 1: A very explicit proof, marked up semantically

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

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

## 18.2 The User Interface

### 18.2.1 Package Options

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

### 18.2.2 Proofs and Proof steps

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

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

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

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

### 18.2.3 Justifications

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

#### 18.2.4 Proof Structure

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

### 18.2.5 Proof End Markers

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

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

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
sproof	\spf@proof@kw	Proof
sketchproof	\spf@sketchproof@kw	ProofSketch

Figure 1: Configuration Hooks for Semantic Proof Markup

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

style	example	configuration macro
long	0.8.1.5	<code>\def\pst@make@label@long#1#2{\@for\@I:=#1\do{\@I.}#2}</code>
angles	$\ggg 5$	<code>\def\pst@make@label@angles#1#2{\ensurermath{\@for\@I:=#1\do{\@I\angle}}#2}</code>
short	5	<code>\def\pst@make@label@short#1#2{#2}</code>
empty		<code>\def\pst@make@label@empty#1#2{}</code>

Figure 2: Configuration Proof Step Label Styles

### 18.3 Limitations

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

---

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

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

## Chapter 19

# sTeX-Metatheory

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

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

### 19.1 Symbols



**Part III**  
**Extensions**

## Chapter 20

# Tikzinput

### 20.1 Macros and Environments

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

## Chapter 21

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

The `document-structure` package is part of the  $\S\TeX$  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 $\TeX$ packages

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

### 21.2.2 Document Structure

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

```
\begin{smodule}{foo}
\symdef{bar}{B^a_r}
...
\begin{omgroup}[id=sec.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.  
`\STRcopy`  
`\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

# STEX -Basics Implementation

### 25.1 The STEXDocument Class

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

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

### 25.2 Preliminaries

```
15 <*package>
16
17 %%%%%%%%% basics.dtx %%%%%%%%%
18
19 \RequirePackage{expl3,l3keys2e,ltxcmds}
20 \ProvidesExplPackage{stex}{2021/08/01}{1.9}{bla}
21 \RequirePackage{expl-keystr-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:nxx{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
810 \str_new:N \l_stex_current_docns_str
811
812 \cs_new_protected:Nn \stex_get_document_uri: {
813   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
814   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
815   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
816   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
817   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
818 }
819
```

```

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

```

```

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

```

## 27.2 Setting Reference Targets

```

895 \str_const:Nn \c__stex_refs_url_str{URL}
896 \str_const:Nn \c__stex_refs_ref_str{REF}
897 \str_new:N \l__stex_refs_curr_label_str
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   \str_clear:N \l__stex_refs_curr_label_str
906   \str_set:Nx \l_tmpa_str { #1 }
907   \str_if_empty:NF \l_tmpa_str {
908     \stex_get_document_uri:
909     \str_set:Nx \l__stex_refs_curr_label_str {
910       \l_stex_current_docns_str?#1
911     }
912     \seq_if_exist:cF{g__stex_refs_labels_#1_seq}{
913       \seq_new:c {g__stex_refs_labels_#1_seq}
914     }
915     \seq_if_in:coF{g__stex_refs_labels_#1_seq}\l__stex_refs_curr_label_str {
916       \seq_gput_right:co{g__stex_refs_labels_#1_seq}\l__stex_refs_curr_label_str
917     }
918     \stex_if_smsmode:TF {
919       \stex_get_document_url:
920       \str_gset_eq:cN {sref_url_}\l__stex_refs_curr_label_str _str}\l_stex_current_docurl_str
921       \str_gset_eq:cN {sref_}\l__stex_refs_curr_label_str _type}\c__stex_refs_url_str
922     }{

```

```

923     \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~~~\expandafter\unexpanded\expandafter
924     \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
925     \immediate\write\@auxout{\stexauxadddocref{\l_stex_current_docns_str}{#1}}
926     \str_gset:cx {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_ref_str
927   }
928 }
929 }
930
931 \cs_new_protected:Npn \stexauxadddocref #1 #2 {
932   \str_set:Nn \l_tmpa_str {#1?#2}
933   \str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
934   \seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
935     \seq_new:c {g__stex_refs_labels_#2_seq}
936   }
937   \seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
938     \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
939   }
940 }
941
942 \AtEndDocument{
943   \def\stexauxadddocref#1{}
944 }
945
946 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
947   \stex_if_smsmode:TF {
948     \str_if_exist:cF{sref_sym_#1_type}{
949       \stex_get_document_url:
950       \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str
951       \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
952     }
953   }{
954     \str_if_empty:NF \l__stex_refs_curr_label_str {
955       \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
956       \immediate\write\@auxout{
957         \exp_not:N\expandafter\def\exp_not:N\csname sref_sym_#1_label_str\exp_not:N\endcsname
958         \l__stex_refs_curr_label_str
959       }
960     }
961   }
962 }
963 }

```

## 27.3 Using References

```

964 \str_new:N \l__stex_refs_indocument_str
965 \keys_define:nn { stex / sref } {
966   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
967   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
968   pre           .tl_set:N = \l__stex_refs_pre_tl ,
969   post          .tl_set:N = \l__stex_refs_post_tl ,
970   %indoc        .str_set_x:N = \l__stex_refs_repo_str ,
971 }
972
973

```

```

974
975 \cs_new_protected:Nn \__stex_refs_args:n {
976   \tl_clear:N \l__stex_refs_linktext_tl
977   \tl_clear:N \l__stex_refs_fallback_tl
978   \tl_clear:N \l__stex_refs_pre_tl
979   \tl_clear:N \l__stex_refs_post_tl
980   \str_clear:N \l__stex_refs_repo_str
981   \keys_set:nn { stex / sref } { #1 }
982 }
983
984 \NewDocumentCommand \sref { 0{} m}{
985   \__stex_refs_args:n { #1 }
986   \str_if_empty:NTF \l__stex_refs_indocument_str {
987     \str_set:Nx \l_tmpa_str { #2 }
988     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
989     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
990       \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
991         \seq_get_left:cnF {g__stex_refs_labels_\l_tmpa_str _seq} \l_tmpa_str {
992           \str_clear:N \l_tmpa_str
993         }
994       }{
995         \str_clear:N \l_tmpa_str
996       }
997     }{
998       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
999       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1000       \int_set:Nn \l_tmpa_int { \exp_args:Nx \str_count:n {\l_tmpb_str?\l_tmpa_str} }
1001       \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
1002         \str_set_eq:NN \l_tmpc_str \l_tmpa_str
1003         \str_clear:N \l_tmpa_str
1004         \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str _seq} {
1005           \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
1006             \str_range:nnn { ##1 }{ -\l_tmpa_int}{ -1 }
1007           }{
1008             \seq_map_break:n {
1009               \str_set:Nn \l_tmpa_str { ##1 }
1010             }
1011           }
1012         }
1013       }{
1014         \str_clear:N \l_tmpa_str
1015       }
1016     }
1017     \str_if_empty:NTF \l_tmpa_str {
1018       \l__stex_refs_fallback_tl
1019     }{
1020       \str_if_eq:cNTF {sref_\l_tmpa_str _type} \c__stex_refs_ref_str {
1021         \cs_if_exist:cTF{autoref}{
1022           \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1023         }{
1024           \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1025         }
1026       }{
1027         \ltx@ifpackageloaded{hyperref}{

```

```

1028         \exp_args:Nx \href{\use:c{sref_url_\l_tmpa_str _str}}{\l__stex_refs_fallback_tl}
1029     }{
1030         \l__stex_refs_fallback_tl
1031     }
1032 }
1033 }
1034 }{
1035     % TODO
1036 }
1037 }
1038
1039 \NewDocumentCommand \srefsym { 0{} m}{
1040     \stex_get_symbol:n { #2 }
1041     \str_if_exist:cTF {sref_sym_\l_stex_get_symbol_uri_str _label_str }{
1042         \sref[#1]{\use:c{sref_sym_\l_stex_get_symbol_uri_str _label_str}}
1043     }{
1044         \__stex_refs_args:n { #1 }
1045         \str_if_empty:NTF \l__stex_refs_indocument_str {
1046             \tl_set:Nn \l_tmpa_tl {
1047                 \l__stex_refs_fallback_tl
1048             }
1049             \tl_if_exist:cT{sref_sym_\l_stex_get_symbol_uri_str _type}{
1050                 \tl_set:Nn \l_tmpa_tl {
1051                     % doc uri in \l_tmpb_str
1052                     \str_set:Nx \l_tmpa_str {\use:c{sref_sym_\l_stex_get_symbol_uri_str _type}}
1053                     \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1054                         % reference
1055                         \cs_if_exist:cTF{autoref}{
1056                             \l__stex_refs_pre_tl\autoref{sref_sym_\l_stex_get_symbol_uri_str}\l__stex_refs_pos
1057                         }{
1058                             \l__stex_refs_pre_tl\ref{sref_sym_\l_stex_get_symbol_uri_str}\l__stex_refs_pos
1059                         }
1060                     }{
1061                         % URL
1062                         \ltx@ifpackageloaded{hyperref}{
1063                             \exp_args:Nx \href{\use:c{sref_sym_url_\l_stex_get_symbol_uri_str _str}}{\l__s
1064                         }{
1065                             \l__stex_refs_fallback_tl
1066                         }
1067                     }
1068                 }
1069             }
1070             \l_tmpa_tl
1071         }{
1072             % TODO
1073         }
1074     }
1075 }
1076
1077 \cs_new_protected:Npn \srefsymuri #1 #2 {
1078     \str_if_exist:cTF {sref_sym_#1 _label_str }{
1079         \exp_args:Nx \hyperref{\use:c{sref_sym_\l_stex_get_symbol_uri_str _label_str}}{#2}
1080     }{
1081         \hyperref[sref_sym_#1]{#2}

```

```
1082     }  
1083   }  
1084  
1085 </package>
```



## Chapter 28

# STEX -Modules Implementation

```
1086 <*package>
1087
1088 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1089
1090 <@@=stex_modules>
1091
1092     Warnings and error messages
1093 \msg_new:nnn{stex}{error/unknownmodule}{
1094     No~module~#1~found
1095 }
1096 \msg_new:nnn{stex}{error/syntax}{
1097     Syntax~error:~#1
1098 }
1099 \msg_new:nnn{stex}{error/siglanguage}{
1100     Module~#1~declares~signature~#2,~but~does~not~
1101     declare~its~language
1102 }
1103 \msg_new:nnn{stex}{error/conflictingmodules}{
1104     Conflicting~imports~for~module~#1
1105 }
1106
1107 \l_stex_current_module_str The current module:
1108 \str_new:N \l_stex_current_module_str
1109
1110 (End definition for \l_stex_current_module_str. This variable is documented on page 26.)
1111
1112 \l_stex_all_modules_seq Stores all available modules
1113 \seq_new:N \l_stex_all_modules_seq
1114
1115 (End definition for \l_stex_all_modules_seq. This variable is documented on page 26.)
1116
1117 \stex_if_in_module_p:
1118 \stex_if_in_module:TF
1119 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1120     \str_if_empty:NTF \l_stex_current_module_str
1121     \prg_return_false: \prg_return_true:
1122 }
```

(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`

```
1111 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1112   \prop_if_exist:cTF { c_stex_module_#1_prop }
1113   \prg_return_true: \prg_return_false:
1114 }
```

(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:

```
1115 \cs_new_protected:Nn \stex_add_to_current_module:n {
1116   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1117 }
1118 \cs_new_protected:Npn \STEXexport {
1119   \beginngroup
1120   \newlinechar=-1\relax
1121   \endlinechar=-1\relax
1122   %\catcode'\ = 9\relax
1123   \expandafter\endgroup\STEXexport:n
1124 }
1125 \cs_new_protected:Nn \STEXexport:n {
1126   \ignorespaces #1
1127   \stex_add_to_current_module:n { \ignorespaces #1 }
1128   \stex_smsmode_do:
1129 }
1130 \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`

```
1131 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1132   \str_set:Nx \l_tmpa_str { #1 }
1133   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1134 }
1135
1136 %\cs_new_protected:Nn \stex_add_field_to_current_module:n {
1137 % \str_set:Nx \l_tmpa_str { #1 }
1138 % \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _fields} { \l_tmpa_str }
1139 %}
```

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

`\stex_collect_imports:n`

```
1140 \cs_new_protected:Nn \stex_collect_imports:n {
1141   \seq_clear:N \l_stex_collect_imports_seq
1142   \__stex_modules_collect_imports:n {#1}
1143 }
1144 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1145   \seq_map_inline:cn {c_stex_module_#1_imports} {
1146     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1147       \__stex_modules_collect_imports:n { ##1 }
1148     }
1149 }
```

```

1149 }
1150 \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1151   \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1152 }
1153 }

```

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

`\stex_add_import_to_current_module:n`

```

1154 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1155   \str_set:Nx \l_tmpa_str { #1 }
1156   \exp_args:Nno
1157   \seq_if_in:cnF{c_stex_module\l_stex_current_module_str_imports}\l_tmpa_str{
1158     \seq_gput_right:co{c_stex_module\l_stex_current_module_str_imports}\l_tmpa_str
1159   }
1160 }

```

(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).

```

1161 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
1162   \str_set:Nx \l_tmpa_str { #1 }
1163   \seq_set_eq:NN \l_tmpa_seq #2
1164   % split off file extension
1165   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1166   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1167   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1168   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1169
1170   \bool_set_true:N \l_tmpa_bool
1171   \bool_while_do:Nn \l_tmpa_bool {
1172     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1173     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1174       {source} { \bool_set_false:N \l_tmpa_bool }
1175     }{}{
1176       \seq_if_empty:NT \l_tmpa_seq {
1177         \bool_set_false:N \l_tmpa_bool
1178       }
1179     }
1180   }
1181
1182   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
1183   \str_if_empty:NTF \l_stex_modules_subpath_str {
1184     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
1185   }{
1186     \str_set:Nx \l_stex_modules_ns_str {
1187       \l_tmpa_str/\l_stex_modules_subpath_str
1188     }
1189   }
1190 }

```

(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
1191 \str_new:N \l_stex_modules_ns_str
1192 \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.

```

1193 \cs_new_protected:Nn \stex_modules_current_namespace: {
1194   \str_clear:N \l_stex_modules_subpath_str
1195   \prop_if_exist:NTF \l_stex_current_repository_prop {
1196     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1197     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1198   }{
1199     % split off file extension
1200     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1201     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1202     \exp_args:Nnno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1203     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1204     \seq_put_right:Nn \l_tmpa_seq \l_tmpb_str
1205     \str_set:Nx \l_stex_modules_ns_str {
1206       file:/\stex_path_to_string:N \l_tmpa_seq
1207     }
1208   }
1209 }

```

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

## 28.1 The module environment

module arguments:

```

1210 \keys_define:nn { stex / module } {
1211   title      .tl_set:N      = \smodulename ,
1212   type       .str_set_x:N   = \smodulename ,
1213   id         .str_set_x:N   = \smodulename ,
1214   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1215   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1216   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1217   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1218   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1219   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1220   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1221 }
1222
1223 \cs_new_protected:Nn \__stex_modules_args:n {
1224   \str_clear:N \smodulename
1225   \str_clear:N \smodulename
1226   \str_clear:N \smodulename
1227   \str_clear:N \l_stex_module_ns_str
1228   \str_clear:N \l_stex_module_lang_str
1229   \str_clear:N \l_stex_module_sig_str
1230   \str_clear:N \l_stex_module_creators_str

```

```

1231 \str_clear:N \l_stex_module_contributors_str
1232 \str_clear:N \l_stex_module_meta_str
1233 \str_clear:N \l_stex_module_srccite_str
1234 \keys_set:nn { stex / module } { #1 }
1235 }
1236
1237 % module parameters here? In the body?
1238

```

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

```

1239 \cs_new_protected:Nn \stex_module_setup:nn {
1240   \str_set:Nx \l_stex_module_name_str { #2 }
1241   \__stex_modules_args:n { #1 }
1242
1243   First, we set up the name and namespace of the module.
1244   Are we in a nested module?
1245
1246   \stex_if_in_module:TF {
1247     % Nested module
1248     \prop_get:cnN {c_stex_module\_l_stex_current_module_str _prop}
1249       { ns } \l_stex_module_ns_str
1250     \str_set:Nx \l_stex_module_name_str {
1251       \prop_item:cn {c_stex_module\_l_stex_current_module_str _prop}
1252       { name } / \l_stex_module_name_str
1253     }
1254   }{
1255     % not nested:
1256     \str_if_empty:NT \l_stex_module_ns_str {
1257       \stex_modules_current_namespace:
1258       \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1259       \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1260         / { \l_stex_module_ns_str }
1261       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1262       \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1263         \str_set:Nx \l_stex_module_ns_str {
1264           \stex_path_to_string:N \l_tmpa_seq
1265         }
1266       }
1267     }
1268   }
1269 }

```

Next, we determine the language of the module:

```

1265 \str_if_empty:NT \l_stex_module_lang_str {
1266   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1267   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1268   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1269   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1270   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1271     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1272       inferred~from~file~name}
1273     \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1274   }
1275 }
1276
1277 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {

```

```

1278 \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1279 \l_tmpa_str {
1280   \ltx@ifpackageloaded{babel}{
1281     \exp_args:Nx \selectlanguage { \l_tmpa_str }
1282   }{}
1283 } {
1284   \msg_error:nxx{stex}{error/unknownlanguage}{\l_tmpa_str}
1285 }
1286 }}

```

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

```

1287 \str_if_empty:NTF \l_stex_module_sig_str {
1288   \exp_args:Nnx \prop_gset_from_keyval:cn {
1289     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1290   } {
1291     name      = \l_stex_module_name_str ,
1292     ns        = \l_stex_module_ns_str ,
1293     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1294     lang      = \l_stex_module_lang_str ,
1295     sig       = \l_stex_module_sig_str ,
1296     meta      = \l_stex_module_meta_str
1297   }
1298   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1299   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _fields}
1300   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1301   \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1302   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1303 \str_if_empty:NT \l_stex_module_meta_str {
1304   \str_set:Nx \l_stex_module_meta_str {
1305     \c_stex_metatheory_ns_str ? Metatheory
1306   }
1307 }
1308 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1309   \bool_set_true:N \l_stex_in_meta_bool
1310   \exp_args:Nx \stex_add_to_current_module:n {
1311     \bool_set_true:N \l_stex_in_meta_bool
1312     \stex_activate_module:n {\l_stex_module_meta_str}
1313     \bool_set_false:N \l_stex_in_meta_bool
1314   }
1315   \stex_activate_module:n {\l_stex_module_meta_str}
1316   \bool_set_false:N \l_stex_in_meta_bool
1317 }
1318 }{
1319   \str_if_empty:NT \l_stex_module_lang_str {
1320     \msg_error:nxxx{stex}{error/siglanguage}{
1321       \l_stex_module_ns_str?\l_stex_module_name_str
1322     }{\l_stex_module_sig_str}
1323   }
1324
1325   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1326   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str

```

```

1327 \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1328 \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1329 \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1330 \str_set:Nx \l_tmpa_str {
1331   \stex_path_to_string:N \l_tmpa_seq /
1332   \l_tmpa_str . \l_stex_module_sig_str .tex
1333 }
1334 \IfFileExists \l_tmpa_str {
1335   \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1336     \str_clear:N \l_stex_current_module_str
1337     \seq_clear:N \l_stex_all_modules_seq
1338     \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1339   }
1340 }{
1341   \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1342 }
1343 \stex_if_smsmode:F {
1344   \stex_activate_module:n {
1345     \l_stex_module_ns_str ? \l_stex_module_name_str
1346   }
1347 }
1348 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1349 }
1350 }

```

(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}

1351 \int_new:N \l_stex_module_group_depth_int
1352 \cs_new_protected:Nn \__stex_modules_begin_module: {
1353   \stex_reactivate_macro:N \STEXexport
1354   \stex_reactivate_macro:N \importmodule
1355   \stex_reactivate_macro:N \symdecl
1356   \stex_reactivate_macro:N \notation
1357   \stex_reactivate_macro:N \symdef
1358
1359   \stex_debug:nn{modules}{
1360     New~module:\\
1361     Namespace:~\l_stex_module_ns_str\\
1362     Name:~\l_stex_module_name_str\\
1363     Language:~\l_stex_module_lang_str\\
1364     Signature:~\l_stex_module_sig_str\\
1365     Metatheory:~\l_stex_module_meta_str\\
1366     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1367   }
1368
1369   \seq_put_right:Nx \l_stex_all_modules_seq {
1370     \l_stex_module_ns_str ? \l_stex_module_name_str
1371   }
1372
1373   % \seq_gput_right:Nx \g_stex_modules_in_file_seq
1374   % { \l_stex_module_ns_str ? \l_stex_module_name_str }

```

```

1375
1376
1377 \stex_if_smsmode:F{
1378   \begin{stex_annotate_env} {theory} {
1379     \l_stex_module_ns_str ? \l_stex_module_name_str
1380   }
1381
1382   \stex_annotate_invisible:nnn{header}{} {
1383     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1384     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1385     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1386       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1387     }
1388     \str_if_empty:NF \smoduletype {
1389       \stex_annotate:nnn{type}{\smoduletype}{}
1390     }
1391   }
1392 }
1393 \int_set:Nn \l_stex_module_group_depth_int {\currentgrouplevel}
1394 % TODO: Inherit metatheory for nested modules?
1395 }
1396 \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}

1397 \cs_new_protected:Nn \_stex_modules_end_module: {
1398   % \str_set:Nx \l_tmpa_str {
1399   %   c_stex_module_
1400   %   \prop_item:Nn \l_stex_current_module_prop { ns } ?
1401   %   \prop_item:Nn \l_stex_current_module_prop { name }
1402   %   _prop
1403   % }
1404   %^^A \prop_new:c { \l_tmpa_str }
1405   % \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
1406   \stex_debug:nn{modules}{Closing module~\prop_item:cn {c_stex_module_\l_stex_current_module_
1407   }

```

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

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

```

1408 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1409 \NewDocumentEnvironment { smodule } { 0{} m } {
1410   \stex_module_setup:nn{#1}{#2}
1411   \par
1412   \stex_if_smsmode:F{
1413     \tl_clear:N \l_tmpa_tl
1414     \clist_map_inline:Nn \smoduletype {
1415       \tl_if_exist:cT {\_stex_modules_smodule_##1_start:}{
1416         \tl_set:Nn \l_tmpa_tl {\use:c{\_stex_modules_smodule_##1_start:}}
1417       }
1418     }
1419     \tl_if_empty:NTF \l_tmpa_tl {
1420       \_stex_modules_smodule_start:

```



```

1421     }{
1422     \l_tmpa_tl
1423     }
1424   }
1425   \__stex_modules_begin_module:
1426   \stex_ref_new_doc_target:n \smoduleid
1427   \stex_smsmode_do:
1428 } {
1429   \__stex_modules_end_module:
1430   \stex_if_smsmode:TF {
1431   %   \exp_args:Nx \stex_add_to_sms:n {
1432   %     \prop_gset_from_keyval:cn {
1433   %       c_stex_module_
1434   %       \prop_item:Nn \l_stex_current_module_prop { ns } ?
1435   %       \prop_item:Nn \l_stex_current_module_prop { name }
1436   %       _prop
1437   %     } {
1438   %       name      = \prop_item:cn { \l_tmpa_str } { name } ,
1439   %       ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
1440   %       file      = \prop_item:cn { \l_tmpa_str } { file } ,
1441   %       lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
1442   %       sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
1443   %       meta      = \prop_item:cn { \l_tmpa_str } { meta }
1444   %     }
1445   %   }
1446   }{
1447     \end{stex_annotate_env}
1448     \clist_set:No \l_tmpa_clist \smoduletype
1449     \tl_clear:N \l_tmpa_tl
1450     \clist_map_inline:Nn \l_tmpa_clist {
1451       \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1452         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1453       }
1454     }
1455     \tl_if_empty:NTF \l_tmpa_tl {
1456       \__stex_modules_smodule_end:
1457     }{
1458       \l_tmpa_tl
1459     }
1460   }
1461 }
1462
1463 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1464 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1465
1466 \newcommand\stexpatchmodule[3] [] {
1467   \str_set:Nx \l_tmpa_str{ #1 }
1468   \str_if_empty:NTF \l_tmpa_str {
1469     \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1470     \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1471   }{
1472     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1473     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1474   }

```

```

1475 }
1476

```

## 28.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n

```

```

1477 \NewDocumentCommand \STEXModule { m } {
1478   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1479   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1480   \tl_set:Nn \l_tmpa_tl {
1481     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1482   }
1483   \seq_map_inline:Nn \l_stex_all_modules_seq {
1484     \str_set:Nn \l_tmpb_str { ##1 }
1485     \str_if_eq:eeT { \l_tmpa_str } {
1486       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1487     } {
1488       \seq_map_break:n {
1489         \tl_set:Nn \l_tmpa_tl {
1490           \stex_invoke_module:n { ##1 }
1491         }
1492       }
1493     }
1494   }
1495   \l_tmpa_tl
1496 }
1497
1498 \cs_new_protected:Nn \stex_invoke_module:n {
1499   \stex_debug:nn{modules}{Invoking~module~#1}
1500   \peek_charcode_remove:NTF ! {
1501     \__stex_modules_invoke_uri:nN { #1 }
1502   } {
1503     \peek_charcode_remove:NTF ? {
1504       \__stex_modules_invoke_symbol:nn { #1 }
1505     } {
1506       \msg_error:nnx{stex}{error/syntax}{
1507         ?~or~!~expected~after~
1508         \c_backslash_str STEXModule{#1}
1509       }
1510     }
1511   }
1512 }
1513
1514 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1515   \str_set:Nn #2 { #1 }
1516 }
1517
1518 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1519   \stex_invoke_symbol:n{#1?#2}
1520 }

```

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

`\stex_activate_module:n`

```
1521 \bool_new:N \l_stex_in_meta_bool
1522 \bool_set_false:N \l_stex_in_meta_bool
1523 \cs_new_protected:Nn \stex_activate_module:n {
1524   \stex_debug:nn{modules}{Activating~module~#1}
1525   \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1526     \msg_error:nnn{stex}{error/conflictingmodules}{ #1 }
1527   }
1528   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1529     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1530     \use:c{ c_stex_module_#1_code }
1531   }
1532 }
```

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

```
1533 </package>
```

## Chapter 29

# STEX -Module Inheritance Implementation

```
1534 <*package>
1535
1536 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1537
```

### 29.1 SMS Mode

```
1538 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1539 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1540 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1541 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1542
1543 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1544   \makeatletter
1545   \makeatother
1546   \ExplSyntaxOn
1547   \ExplSyntaxOff
1548   \rustexBREAK
1549 }
1550
1551 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1552   \symdef
1553   \importmodule
1554   \notation
1555   \symdecl
1556   \STEXexport
1557   \inlineass
1558   \inlinedef
1559   \inlineex
1560   \endinput
1561   \setnotation
```

```

1562 \copynotation
1563 }
1564
1565 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1566   \tl_to_str:n {
1567     smodule,
1568     copymodule,
1569     interpretmodule
1570     sdefinition,
1571     sexample,
1572     sassertion,
1573     sparagraph
1574   }
1575 }

```

(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`

```

1576 \bool_new:N \g__stex_smsmode_bool
1577 \bool_set_false:N \g__stex_smsmode_bool
1578 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1579   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1580 }

```

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

`\stex_in_smsmode:nn`

```

1581 \cs_new_protected:Nn \stex_in_smsmode:nn {
1582   \vbox_set:Nn \l_tmpa_box {
1583     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1584     \bool_gset_true:N \g__stex_smsmode_bool
1585     #2
1586     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1587   }
1588   \box_clear:N \l_tmpa_box
1589 }
1590
1591 \quark_new:N \q__stex_smsmode_break
1592
1593 %\ior_new:N \c__stex_smsmode_ior
1594 %\tl_new:N \l__stex_smsmode_filecontent_tl
1595 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1596   % \tl_clear:N \l__stex_smsmode_filecontent_tl
1597   % \ior_open:Nn \c__stex_smsmode_ior {#1}
1598   % \ior_map_inline:Nn \c__stex_smsmode_ior {
1599     % \tl_put_right:Nn \l__stex_smsmode_filecontent_tl { ##1 }
1600   % }
1601   % \ior_close:N \c__stex_smsmode_ior
1602   \stex_filestack_push:n{#1}
1603   \stex_in_smsmode:nn{#1} {
1604     #2
1605     \everyeof{\q__stex_smsmode_break\noexpand}
1606     \expandafter\expandafter\expandafter
1607     \stex_smsmode_do:

```

```

1608     \csname @ @ input\endcsname "#1"\relax
1609     %\expandafter \stex_smsmode_do: \l__stex_smsmode_filecontent_tl
1610   }
1611   \stex_filestack_pop:
1612 }

```

(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:

```

1613 \cs_new_protected:Npn \stex_smsmode_do: {
1614   \stex_if_smsmode:T {
1615     \__stex_smsmode_do:w
1616   }
1617 }
1618 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1619   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
1620     \expandafter\if\expandafter\relax\noexpand#1
1621     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1622   }else\expandafter\__stex_smsmode_do:w\fi
1623 }{
1624   \__stex_smsmode_do:w % #1
1625 }
1626 }
1627 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1628   \cs_if_eq:NNTF #1 \q__stex_smsmode_break {
1629     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1630       #1\__stex_smsmode_do:w
1631     }{
1632       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1633         #1
1634       }{
1635         \cs_if_eq:NNTF \begin #1 {
1636           \__stex_smsmode_check_begin:n
1637         }{
1638           \cs_if_eq:NNTF \end #1 {
1639             \__stex_smsmode_check_end:n
1640           }{
1641             \__stex_smsmode_do:w
1642           }
1643         }
1644       }
1645     }
1646   }
1647 }
1648
1649 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1650   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1651     \begin{#1}
1652   }{
1653     \__stex_smsmode_do:w
1654   }
1655 }
1656 \cs_new_protected:Nn \__stex_smsmode_check_end:n {

```

```

1657 \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1658   \end{#1}\__stex_smsmode_do:w
1659 }{
1660   \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1661 }
1662 }

```

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

## 29.2 Inheritance

```

1663 <@@=stex_importmodule>

```

`\stex_import_module_uri:nn`

```

1664 \cs_new_protected:Nn \stex_import_module_uri:nn {
1665   \str_set:Nx \l_stex_import_archive_str { #1 }
1666   \str_set:Nn \l_stex_import_path_str { #2 }
1667
1668   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1669   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1670   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1671
1672   \stex_modules_current_namespace:
1673   \bool_lazy_all:nTF {
1674     {\str_if_empty_p:N \l_stex_import_archive_str}
1675     {\str_if_empty_p:N \l_stex_import_path_str}
1676     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1677   }{
1678     \str_set_eq:NN \l_stex_import_path_str \l_stex_modules_subpath_str
1679     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1680   }{
1681     \str_if_empty:NT \l_stex_import_archive_str {
1682       \prop_if_exist:NT \l_stex_current_repository_prop {
1683         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1684       }
1685     }
1686     \str_if_empty:NTF \l_stex_import_archive_str {
1687       \str_if_empty:NF \l_stex_import_path_str {
1688         \str_set:Nx \l_stex_import_ns_str {
1689           \l_stex_module_ns_str / \l_stex_import_path_str
1690         }
1691       }
1692     }{
1693       \stex_require_repository:n \l_stex_import_archive_str
1694       \prop_get:cnN { c_stex_mathhub_\l_stex_import_archive_str_manifest_prop } { ns }
1695       \l_stex_import_ns_str
1696       \str_if_empty:NF \l_stex_import_path_str {
1697         \str_set:Nx \l_stex_import_ns_str {
1698           \l_stex_import_ns_str / \l_stex_import_path_str
1699         }
1700       }
1701     }
1702   }
1703 }

```

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

```

\l_stex_import_name_str Store the return values of \stex_import_module_uri:nn.
\l_stex_import_archive_str 1704 \str_new:N \l_stex_import_name_str
\l_stex_import_path_str 1705 \str_new:N \l_stex_import_archive_str
\l_stex_import_ns_str 1706 \str_new:N \l_stex_import_path_str
1707 \str_new:N \l_stex_import_ns_str

```

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

```

\stex_import_require_module:nmmn {<ns>} {<archive-ID>} {<path>} {<name>}
1708 \cs_new_protected:Nn \stex_import_require_module:nmmn {
1709   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1710
1711     % archive
1712     \str_set:Nx \l_tmpa_str { #2 }
1713     \str_if_empty:NTF \l_tmpa_str {
1714       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1715     } {
1716       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1717       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1718       \seq_put_right:Nn \l_tmpa_seq { source }
1719     }
1720
1721     % path
1722     \str_set:Nx \l_tmpb_str { #3 }
1723     \str_if_empty:NTF \l_tmpb_str {
1724       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1725
1726       \ltx@ifpackageloaded{babel} {
1727         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1728           { \languagename } \l_tmpb_str {
1729           \msg_error:nnx{stex}{error/unknownlanguage}{\languagename}
1730         }
1731       } {
1732         \str_clear:N \l_tmpb_str
1733       }
1734
1735       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1736       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1737         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1738       }{
1739         \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1740         \IfFileExists{ \l_tmpa_str.tex }{
1741           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1742         }{
1743           % try english as default
1744           \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1745           \IfFileExists{ \l_tmpa_str.en.tex }{
1746             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1747           }{
1748             \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1749           }
1750         }

```



```

1751     }
1752
1753   } {
1754     \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1755     \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1756
1757     \ltx@ifpackageloaded{babel} {
1758       \exp_args:NnX \prop_get:NnNF \c_stex_language_abbrevs_prop
1759         { \language } \l_tmpb_str {
1760         \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1761       }
1762     } {
1763       \str_clear:N \l_tmpb_str
1764     }
1765
1766     \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1767
1768     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1769     \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1770       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1771     }{
1772       \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1773       \IfFileExists{ \l_tmpa_str/#4.tex }{
1774         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1775       }{
1776         % try english as default
1777         \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1778         \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1779           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1780         }{
1781           \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1782           \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1783             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1784           }{
1785             \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1786             \IfFileExists{ \l_tmpa_str.tex }{
1787               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1788             }{
1789               % try english as default
1790               \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1791               \IfFileExists{ \l_tmpa_str.en.tex }{
1792                 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1793               }{
1794                 \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1795               }
1796             }
1797           }
1798         }
1799       }
1800     }
1801   }
1802
1803   \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
1804     \seq_clear:N \l_stex_all_modules_seq

```

```

1805     \str_clear:N \l_stex_current_module_str
1806     \str_set:Nx \l_tmpb_str { #2 }
1807     \str_if_empty:NF \l_tmpb_str {
1808       \stex_set_current_repository:n { #2 }
1809     }
1810     \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1811   }
1812
1813   \stex_if_module_exists:nF { #1 ? #4 } {
1814     \msg_error:nnx{stex}{error/unknownmodule}{
1815       #1?#4~(in~file~\g__stex_importmodule_file_str)
1816     }
1817   }
1818 }
1819 \stex_activate_module:n { #1 ? #4 }
1820 }

```

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

### `\importmodule`

```

1821 \NewDocumentCommand \importmodule { 0{} m } {
1822   \stex_import_module_uri:nn { #1 } { #2 }
1823   \stex_debug:nn{modules}{Importing~module:~
1824     \l_stex_import_ns_str ? \l_stex_import_name_str
1825   }
1826   \stex_if_smsmode:F {
1827     \stex_import_require_module:nnnn
1828     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1829     { \l_stex_import_path_str } { \l_stex_import_name_str }
1830     \stex_annotate_invisible:nnn
1831     {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1832   }
1833   \exp_args:Nx \stex_add_to_current_module:n {
1834     \stex_import_require_module:nnnn
1835     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1836     { \l_stex_import_path_str } { \l_stex_import_name_str }
1837   }
1838   \exp_args:Nx \stex_add_import_to_current_module:n {
1839     \l_stex_import_ns_str ? \l_stex_import_name_str
1840   }
1841   \stex_smsmode_do:
1842 }
1843 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

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

### `\usemodule`

```

1844 \NewDocumentCommand \usemodule { 0{} m } {
1845   \stex_if_smsmode:F {
1846     \stex_import_module_uri:nn { #1 } { #2 }
1847     \stex_import_require_module:nnnn
1848     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1849     { \l_stex_import_path_str } { \l_stex_import_name_str }
1850     \stex_annotate_invisible:nnn
1851     {usemodule} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}

```

```

1852 }
1853 \stex_smsmode_do:
1854 }

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

1855 </package>

```

## Chapter 30

# STEX -Symbols Implementation

```
1856 <*package>
1857
1858 %%%%%%%%%% symbols.dtx %%%%%%%%%%
1859
```

Warnings and error messages

```
1860
```

### 30.1 Symbol Declarations

```
1861 <@@=stex_symdecl>
```

`\l_stex_all_symbols_seq` Stores all available symbols

```
1862 \seq_new:N \l_stex_all_symbols_seq
```

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

`\STEXsymbol`

```
1863 \NewDocumentCommand \STEXsymbol { m } {
1864   \stex_get_symbol:n { #1 }
1865   \exp_args:No
1866   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1867 }
```

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

symdecl arguments:

```
1868 \keys_define:nn { stex / symdecl } {
1869   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1870   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1871   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1872   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
1873   align     .str_set:N    = \l_stex_symdecl_align_str , % TODO(?)
1874   gfc       .str_set:N    = \l_stex_symdecl_gfc_str , % TODO(?)
1875   specializes .str_set:N  = \l_stex_symdecl_specializes_str , % TODO(?)
1876   def       .tl_set:N    = \l_stex_symdecl_definiens_tl
1877 }
```

```

1878
1879 \bool_new:N \l_stex_symdecl_make_macro_bool
1880
1881 \cs_new_protected:Nn \__stex_symdecl_args:n {
1882   \str_clear:N \l_stex_symdecl_name_str
1883   \str_clear:N \l_stex_symdecl_args_str
1884   \bool_set_false:N \l_stex_symdecl_local_bool
1885   \tl_clear:N \l_stex_symdecl_type_tl
1886   \tl_clear:N \l_stex_symdecl_definiens_tl
1887
1888   \keys_set:nn { stex / symdecl } { #1 }
1889 }

```

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

```

1890
1891 \NewDocumentCommand \symdecl { s O{} m } {
1892   \__stex_symdecl_args:n { #2 }
1893   \IfBooleanTF #1 {
1894     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1895   } {
1896     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1897   }
1898   \stex_symdecl_do:n { #3 }
1899   \stex_smsmode_do:
1900 }
1901 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

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

```

1902 \cs_new_protected:Nn \stex_symdecl_do:n {
1903   \stex_if_in_module:F {
1904     % TODO throw error? some default namespace?
1905   }
1906
1907   \str_if_empty:NT \l_stex_symdecl_name_str {
1908     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1909   }
1910
1911   \prop_if_exist:cT { l_stex_symdecl_
1912     \l_stex_current_module_str ?
1913     \l_stex_symdecl_name_str
1914     _prop
1915   }{
1916     % TODO throw error (beware of circular dependencies)
1917   }
1918
1919   \prop_clear:N \l_tmpa_prop
1920   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
1921   \seq_clear:N \l_tmpa_seq
1922   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1923   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1924

```

```

1925 \exp_args:No \stex_add_constant_to_current_module:n {
1926   \l_stex_symdecl_name_str
1927 }
1928
1929 % arity/args
1930 \int_zero:N \l_tmpb_int
1931
1932 \bool_set_true:N \l_tmpa_bool
1933 \str_map_inline:Nn \l_stex_symdecl_args_str {
1934   \token_case_meaning:NnF ##1 {
1935     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1936     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1937     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1938     {\tl_to_str:n a} {
1939       \bool_set_false:N \l_tmpa_bool
1940       \int_incr:N \l_tmpb_int
1941     }
1942     {\tl_to_str:n B} {
1943       \bool_set_false:N \l_tmpa_bool
1944       \int_incr:N \l_tmpb_int
1945     }
1946   }{
1947     \msg_set:nnn{stex}{error/wrongargs}{
1948       args~value~in~symbol~declaration~for~
1949       \l_stex_current_module_str ?
1950       \l_stex_symdecl_name_str ~
1951       needs~to~be~
1952       i,~a,~b~or~B,~but~##1~given
1953     }
1954     \msg_error:nn{stex}{error/wrongargs}
1955   }
1956 }
1957 \bool_if:NTF \l_tmpa_bool {
1958   % possibly numeric
1959   \str_if_empty:NTF \l_stex_symdecl_args_str {
1960     \prop_put:Nnn \l_tmpa_prop { args } {}
1961     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1962   }{
1963     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1964     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1965     \str_clear:N \l_tmpa_str
1966     \int_step_inline:nn \l_tmpa_int {
1967       \str_put_right:Nn \l_tmpa_str i
1968     }
1969     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1970   }
1971 } {
1972   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1973   \prop_put:Nnx \l_tmpa_prop { arity }
1974     { \str_count:N \l_stex_symdecl_args_str }
1975 }
1976 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
1977
1978

```

```

1979 % semantic macro
1980
1981 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1982   \exp_args:Nx \stex_do_aftergroup:n {
1983     \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1984       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1985     }}
1986   }
1987
1988   \bool_if:NF \l_stex_symdecl_local_bool {
1989     \exp_args:Nx \stex_add_to_current_module:n {
1990       \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1991         \l_stex_current_module_str ? \l_stex_symdecl_name_str
1992       } }
1993     }
1994   }
1995 }
1996
1997 % add to all symbols
1998
1999 \bool_if:NF \l_stex_symdecl_local_bool {
2000   \exp_args:Nx \stex_add_to_current_module:n {
2001     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2002       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2003     }
2004   }
2005 %   \exp_args:Nx \stex_add_field_to_current_module:n {
2006 %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2007 %   }
2008 }
2009
2010 \stex_debug:nn{symbols}{New~symbol:~
2011   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2012   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2013   Args:~\prop_item:Nn \l_tmpa_prop { args }
2014 }
2015
2016 % circular dependencies require this:
2017
2018 \prop_if_exist:cF {
2019   l_stex_symdecl_
2020   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2021   _prop
2022 } {
2023   \prop_set_eq:cN {
2024     l_stex_symdecl_
2025     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2026     _prop
2027   } \l_tmpa_prop
2028 }
2029
2030 \seq_clear:c {
2031   l_stex_symdecl_
2032   \l_stex_current_module_str ? \l_stex_symdecl_name_str

```

```

2033   _notations
2034 }
2035
2036 \bool_if:NF \l_stex_symdecl_local_bool {
2037   \exp_args:Nx
2038   \stex_add_to_current_module:n {
2039     \seq_clear:c {
2040       l_stex_symdecl_
2041       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2042       _notations
2043     }
2044     \prop_set_from_keyval:cn {
2045       l_stex_symdecl_
2046       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2047       _prop
2048     } {
2049       name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2050       module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2051       type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2052       args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2053       arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2054       assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
2055     }
2056   }
2057 }
2058
2059 \stex_if_smsmode:TF {
2060   \bool_if:NF \l_stex_symdecl_local_bool {
2061     % \exp_args:Nx \stex_add_to_sms:n {
2062     %   \prop_set_from_keyval:cn {
2063     %     l_stex_symdecl_
2064     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2065     %     _prop
2066     %   } {
2067     %     name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2068     %     module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2069     %     local     = \prop_item:Nn \l_tmpa_prop { local }     ,
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     %   \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2076     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2077     %   }
2078     % }
2079   }
2080 }{
2081   \exp_args:Nx \stex_do_aftergroup:n {
2082     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2083       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2084     }
2085   }
2086   \stex_if_do_html:T {

```



```

2087 \stex_annotate_invisible:nnn {symdecl} {
2088   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2089 } {
2090   \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}{}}{ $\l_st
2091   \stex_annotate_invisible:nnn{args}{}{
2092     \prop_item:Nn \l_tmpa_prop { args }
2093   }
2094   \stex_annotate_invisible:nnn{macroname}{#1}{}
2095   \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2096     \stex_annotate_invisible:nnn{definiens}{}
2097     { $\l_stex_symdecl_definiens_tl$ }
2098   }
2099 }
2100 }
2101 }
2102 }

```

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

`\stex_get_symbol:n`

```

2103 \str_new:N \l_stex_get_symbol_uri_str
2104
2105 \cs_new_protected:Nn \stex_get_symbol:n {
2106   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2107     \__stex_symdecl_get_symbol_from_cs:n { #1 }
2108   }{
2109     % argument is a string
2110     % is it a command name?
2111     \cs_if_exist:cTF { #1 }{
2112       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2113       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2114       \str_if_empty:NNTF \l_tmpa_str {
2115         \exp_args:Nx \cs_if_eq:NNTF {
2116           \tl_head:N \l_tmpa_tl
2117         } \stex_invoke_symbol:n {
2118           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
2119         }{
2120           \__stex_symdecl_get_symbol_from_string:n { #1 }
2121         }
2122       } {
2123         \__stex_symdecl_get_symbol_from_string:n { #1 }
2124       }
2125     }{
2126       % argument is not a command name
2127       \__stex_symdecl_get_symbol_from_string:n { #1 }
2128       % \l_stex_all_symbols_seq
2129     }
2130   }
2131 }
2132
2133 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2134   \str_set:Nn \l_tmpa_str { #1 }
2135   \bool_set_false:N \l_tmpa_bool
2136   \stex_if_in_module:T {

```

```

2137 \exp_args:Nno \seq_if_in:cnT {c_stex_module_\l_stex_current_module_str _constants} { \l_
2138 \bool_set_true:N \l_tmpa_bool
2139 \str_set:Nx \l_stex_get_symbol_uri_str {
2140 \l_stex_current_module_str ? #1
2141 }
2142 }
2143 }
2144 \bool_if:NF \l_tmpa_bool {
2145 \tl_set:Nn \l_tmpa_tl {
2146 \msg_set:nnn{stex}{error/unknownsymbol}{
2147 No~symbol~#1~found!
2148 }
2149 \msg_error:nn{stex}{error/unknownsymbol}
2150 }
2151 \str_set:Nn \l_tmpa_str { #1 }
2152 \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2153 \seq_map_inline:Nn \l_stex_all_symbols_seq {
2154 \str_set:Nn \l_tmpb_str { ##1 }
2155 \str_if_eq:eeT { \l_tmpa_str } {
2156 \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
2157 } {
2158 \seq_map_break:n {
2159 \tl_set:Nn \l_tmpa_tl {
2160 \str_set:Nn \l_stex_get_symbol_uri_str {
2161 ##1
2162 }
2163 }
2164 }
2165 }
2166 }
2167 \l_tmpa_tl
2168 }
2169 }
2170
2171 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
2172 \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2173 { \tl_tail:N \l_tmpa_tl }
2174 \tl_if_single:NTF \l_tmpa_tl {
2175 \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2176 \exp_after:wN \str_set:Nn \exp_after:wN
2177 \l_stex_get_symbol_uri_str \l_tmpa_tl
2178 }{
2179 % TODO
2180 % tail is not a single group
2181 }
2182 }{
2183 % TODO
2184 % tail is not a single group
2185 }
2186 }

```

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

## 30.2 Notations

```

2187 <@@=stex_notation>

      notation arguments:
2188 \keys_define:nn { stex / notation } {
2189   lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2190   variant   .tl_set_x:N = \l__stex_notation_variant_str ,
2191   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2192   op        .tl_set:N   = \l__stex_notation_op_tl ,
2193   primary   .bool_set:N = \l__stex_notation_primary_bool ,
2194   primary   .default:n  = {true} ,
2195   unknown   .code:n     = \str_set:Nx
2196               \l__stex_notation_variant_str \l_keys_key_str
2197 }
2198
2199 \cs_new_protected:Nn \stex_notation_args:n {
2200   \str_clear:N \l__stex_notation_lang_str
2201   \str_clear:N \l__stex_notation_variant_str
2202   \str_clear:N \l__stex_notation_prec_str
2203   \tl_clear:N \l__stex_notation_op_tl
2204   \bool_set_false:N \l__stex_notation_primary_bool
2205
2206   \keys_set:nn { stex / notation } { #1 }
2207 }

```

**\notation**

```

2208 \NewDocumentCommand \notation { 0{ } m } {
2209   \stex_notation_args:n { #1 }
2210   \tl_clear:N \l_stex_symdecl_definiens_tl
2211   \stex_get_symbol:n { #2 }
2212   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
2213 }
2214 \stex_deactivate_macro:Nn \notation {module-environments}

```

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

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

```

2215 \seq_new:N \l__stex_notation_precedences_seq
2216 \tl_new:N \l__stex_notation_opprec_tl
2217 \int_new:N \l__stex_notation_currarg_int
2218
2219 \cs_new_protected:Nn \stex_notation_do:nn {
2220   \let\l_stex_current_symbol_str\relax
2221   \str_set:Nx \l__stex_notation_symbol_str { #1 }
2222   \seq_clear:N \l__stex_notation_precedences_seq
2223   \tl_clear:N \l__stex_notation_opprec_tl
2224   \prop_get:cnN {
2225     l_stex_symdecl_ #1 _prop
2226   } { args } \l__stex_notation_args_str
2227
2228   % precedences
2229   \prop_get:cnN {
2230     l_stex_symdecl_ #1 _prop
2231   } { arity } \l__stex_notation_arity_str

```

```

2232 \str_if_empty:NTF \l__stex_notation_prec_str {
2233   \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2234     \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2235   }{
2236     \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2237   }
2238 } {
2239   \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2240     \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2241     \int_step_inline:nn { \l__stex_notation_arity_str } {
2242       \exp_args:NNo
2243       \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2244     }
2245   }{
2246     \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2247     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2248       \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2249       \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2250         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2251         \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2252         \seq_map_inline:Nn \l_tmpa_seq {
2253           \seq_put_right:Nn \l_tmpb_seq { ##1 }
2254         }
2255       }
2256     }{
2257       \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2258         \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2259       }{
2260         \tl_set:No \l__stex_notation_opprec_tl { 0 }
2261       }
2262     }
2263   }
2264 }
2265
2266 \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2267 \int_step_inline:nn { \l__stex_notation_arity_str } {
2268   \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
2269     \exp_args:NNo
2270     \seq_put_right:No \l__stex_notation_precedences_seq {
2271       \l__stex_notation_opprec_tl
2272     }
2273   }
2274 }
2275
2276 \tl_clear:N \l__stex_notation_dummyargs_tl
2277
2278 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2279   \exp_args:NNe
2280   \cs_set:Npn \l__stex_notation_macrocode_cs {
2281     \stex_term_math_oms:nxxx { \l_stex_current_symbol_str }
2282     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2283     { \l__stex_notation_opprec_tl }
2284     { \exp_not:n { #2 } }
2285   }

```

```

2286   \__stex_notation_final:
2287 }{
2288   \str_if_in:NnTF \l__stex_notation_args_str b {
2289     \exp_args:Nne \use:nn
2290     {
2291       \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2292       \cs_set:Npn \l__stex_notation_arity_str } { {
2293         \stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2294         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2295         { \l__stex_notation_opprec_tl }
2296         { \exp_not:n { #2 } }
2297       }}
2298 }{
2299   \str_if_in:NnTF \l__stex_notation_args_str B {
2300     \exp_args:Nne \use:nn
2301     {
2302       \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2303       \cs_set:Npn \l__stex_notation_arity_str } { {
2304         \stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2305         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2306         { \l__stex_notation_opprec_tl }
2307         { \exp_not:n { #2 } }
2308       } }
2309 }{
2310   \exp_args:Nne \use:nn
2311   {
2312     \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2313     \cs_set:Npn \l__stex_notation_arity_str } { {
2314       \stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2315       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2316       { \l__stex_notation_opprec_tl }
2317       { \exp_not:n { #2 } }
2318     } }
2319   }
2320 }
2321
2322 \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2323 \int_zero:N \l__stex_notation_currarg_int
2324 \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2325 \__stex_notation_arguments:
2326 }
2327 }

```

(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

```

2328 \cs_new_protected:Nn \__stex_notation_arguments: {
2329   \int_incr:N \l__stex_notation_currarg_int
2330   \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2331     \__stex_notation_final:
2332   }{
2333     \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2334     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2335     \str_if_eq:VnTF \l_tmpa_str a {

```

```

2336     \_stex_notation_argument_assoc:n
2337   }{
2338     \str_if_eq:VnTF \l_tmpa_str B {
2339       \_stex_notation_argument_assoc:n
2340     }{
2341       \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2342       \tl_put_right:Nx \l__stex_notation_dummyargs_tl {
2343         { \_stex_term_math_arg:nnn
2344           { \int_use:N \l__stex_notation_currarg_int }
2345           { \l_tmpa_str }
2346           { #####\int_use:N \l__stex_notation_currarg_int }
2347         }
2348       }
2349       \_stex_notation_arguments:
2350     }
2351   }
2352 }
2353 }

```

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

\\_stex\_notation\_argument\_assoc:n

```

2354 \cs_new_protected:Nn \_stex_notation_argument_assoc:n {
2355
2356   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2357   {\l__stex_notation_arity_str}{
2358     #1
2359   }
2360   \int_zero:N \l_tmpa_int
2361   \tl_clear:N \l_tmpa_tl
2362   \str_map_inline:Nn \l__stex_notation_args_str {
2363     \int_incr:N \l_tmpa_int
2364     \tl_put_right:Nx \l_tmpa_tl {
2365       \str_if_eq:nnTF {##1}{a}{ {} }{
2366         \str_if_eq:nnTF {##1}{B}{ {} }{
2367           {##### \int_use:N \l_tmpa_int}
2368         }
2369       }
2370     }
2371   }
2372   \exp_after:wN\exp_after:wN\exp_after:wN \def
2373   \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2374   \exp_after:wN\exp_after:wN\exp_after:wN ##
2375   \exp_after:wN\exp_after:wN\exp_after:wN 1
2376   \exp_after:wN\exp_after:wN\exp_after:wN ##
2377   \exp_after:wN\exp_after:wN\exp_after:wN 2
2378   \exp_after:wN\exp_after:wN\exp_after:wN {
2379     \exp_after:wN \exp_after:wN \exp_after:wN
2380     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2381       \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2382     }
2383   }
2384
2385   \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str

```

```

2386 \tl_put_right:Nx \l__stex_notation_dummyargs_tl { {
2387   \stex_term_math_assoc_arg:nnnn
2388   { \int_use:N \l__stex_notation_currarg_int }
2389   { \l_tmpa_str }
2390   { ####\int_use:N \l__stex_notation_currarg_int }
2391   { \l_tmpa_cs {####1} {####2} }
2392 } }
2393 %\cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
2394 %\tl_put_right:Nx \l_tmpa_tl {
2395 % { \stex_term_math_assoc_arg:nnnn
2396 % { \int_use:N \l_tmpa_int }
2397 % { \l_tmpb_str }
2398 % \exp_args:No \exp_not:n
2399 % {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
2400 % { ####\int_use:N \l_tmpa_int }
2401 % }
2402 %}
2403 \__stex_notation_arguments:
2404 }

```

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

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

```

2405 \cs_new_protected:Nn \__stex_notation_final: {
2406   \exp_args:Nne \use:nn
2407   {
2408     \cs_generate_from_arg_count:cNnn {
2409       stex_notation_ \l__stex_notation_symbol_str \c_hash_str
2410       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2411       _cs
2412     }
2413     \cs_set:Npn \l__stex_notation_arity_str { { {
2414       \exp_after:wN \exp_after:wN \exp_after:wN
2415       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2416       { \exp_after:wN \l__stex_notation_macrocode_cs \l__stex_notation_dummyargs_tl }
2417     } } }
2418
2419     \tl_if_empty:NF \l__stex_notation_op_tl {
2420       \cs_set:cpx {
2421         stex_op_notation_ \l__stex_notation_symbol_str \c_hash_str
2422         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2423         _cs
2424       } {
2425         \stex_term_oms:nnn {
2426           \l__stex_notation_symbol_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2427           \l__stex_notation_lang_str
2428         }{
2429           \l__stex_notation_symbol_str
2430         }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } } }
2431       }
2432     }
2433
2434     \exp_args:Ne
2435     \stex_add_to_current_module:n {

```

```

2436 \cs_generate_from_arg_count:cNnn {
2437   stex_notation_ \l__stex_notation_symbol_str \c_hash_str
2438   \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2439   _cs
2440 } \cs_set:Npn {\l__stex_notation_arity_str} {
2441   \exp_after:wN \exp_after:wN \exp_after:wN
2442   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2443   { \exp_after:wN \l__stex_notation_macrocode_cs \l__stex_notation_dummyargs_tl }
2444 }
2445 \tl_if_empty:NF \l__stex_notation_op_tl {
2446   \cs_set:cpn {
2447     stex_op_notation_ \l__stex_notation_symbol_str \c_hash_str
2448     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2449     _cs
2450   } {
2451     \stex_term_oms:nnn {
2452       \l__stex_notation_symbol_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2453       \l__stex_notation_lang_str
2454     }{
2455       \l__stex_notation_symbol_str
2456     }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2457   }
2458 }
2459 }
2460 \exp_args:Nx
2461 % \stex_do_aftergroup:n {
2462   \seq_put_right:cx {
2463     l_stex_symdecl_ \l__stex_notation_symbol_str
2464     _notations
2465   } {
2466     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2467   }
2468 % }
2469
2470 \stex_debug:nn{symbols}{
2471   Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2472   ~for~\l__stex_notation_symbol_str^^J
2473   Operator~precedence:~\l__stex_notation_opprec_tl^^J
2474   Argument~precedences:~
2475   \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2476   Notation: \cs_meaning:c {
2477     stex_notation_ \l__stex_notation_symbol_str \c_hash_str
2478     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2479     _cs
2480   }
2481 }
2482
2483 %\prop_set_eq:cN {
2484 %   l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2485 %   \c_hash_str \l__stex_notation_lang_str _prop
2486 %} \l_tmpb_prop
2487
2488 \exp_args:Ne
2489 \stex_add_to_current_module:n {

```



```

2490 \seq_put_right:cn {
2491   l_stex_symdecl_ \l__stex_notation_symbol_str
2492   _notations
2493 } {
2494   \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2495 }
2496 %\prop_set_from_keyval:cn {
2497 % l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2498 %   \c_hash_str \l__stex_notation_lang_str _prop
2499 %} {
2500 % symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2501 % language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2502 % variant     = \prop_item:Nn \l_tmpb_prop { variant }      ,
2503 % opprec      = \prop_item:Nn \l_tmpb_prop { opprec }       ,
2504 % argprec     = \prop_item:Nn \l_tmpb_prop { argprec }      ,
2505 %}
2506 }
2507
2508 \stex_if_smsmode:TF {
2509 %   \exp_args:Nx \stex_add_to_sms:n {
2510 %     \prop_set_from_keyval:cn {
2511 %       l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2512 %       \c_hash_str \l__stex_notation_lang_str _prop
2513 %     } {
2514 %       symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2515 %       language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2516 %       variant     = \prop_item:Nn \l_tmpb_prop { variant }      ,
2517 %       opprec      = \prop_item:Nn \l_tmpb_prop { opprec }       ,
2518 %       argprec     = \prop_item:Nn \l_tmpb_prop { argprec }      ,
2519 %     }
2520 %   }
2521 %}
2522
2523 % HTML annotations
2524 \stex_if_do_html:T {
2525   \stex_annotate_invisible:nnn { notation }
2526   { \l__stex_notation_symbol_str } {
2527     \stex_annotate_invisible:nnn { notationfragment }
2528     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
2529     \stex_annotate_invisible:nnn { precedence }
2530     { \l__stex_notation_prec_str }{}
2531
2532     \int_zero:N \l_tmpa_int
2533     \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2534     \tl_clear:N \l_tmpa_tl
2535     \int_step_inline:nn { \l__stex_notation_arity_str }{
2536       \int_incr:N \l_tmpa_int
2537       \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2538       \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_r
2539       \str_if_eq:VnTF \l_tmpb_str a {
2540         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2541           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2542           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2543         } }

```

```

2544     }{
2545       \str_if_eq:VnTF \l_tmpb_str B {
2546         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2547           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2548           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2549         } }
2550       }{
2551         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2552           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2553         } }
2554       }
2555     }
2556   }
2557   \stex_annotate_invisible:nnn { notationcomp }{}{
2558     \str_set:Nx \l_stex_current_symbol_str { \l__stex_notation_symbol_str }
2559     $ \exp_args:Nno \use:nn { \use:c {
2560       stex_notation_ \l_stex_current_symbol_str
2561       \c_hash_str \l__stex_notation_variant_str
2562       \c_hash_str \l__stex_notation_lang_str _cs
2563     } } { \l_tmpa_tl } $
2564   }
2565 }
2566 }
2567 }
2568 \stex_smsmode_do:
2569 }

```

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

\setnotation

```

2570 \keys_define:nn { stex / setnotation } {
2571   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2572   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2573   unknown .code:n = \str_set:Nx
2574     \l__stex_notation_variant_str \l_keys_key_str
2575 }
2576
2577 \cs_new_protected:Nn \_stex_setnotation_args:n {
2578   \str_clear:N \l__stex_notation_lang_str
2579   \str_clear:N \l__stex_notation_variant_str
2580   \keys_set:nn { stex / setnotation } { #1 }
2581 }
2582
2583 \NewDocumentCommand \setnotation {m m} {
2584   \stex_get_symbol:n { #1 }
2585   \_stex_setnotation_args:n { #2 }
2586   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl\_l_stex_get_symbol_uri_str _notations }
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       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2590     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notation
2591       { \c_hash_str }
2592     \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notations
2593       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }

```

```

2594 \exp_args:Nx \stex_add_to_current_module:n {
2595   \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _notati
2596     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2597   \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _notation
2598     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2599   \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _notati
2600     { \c_hash_str }
2601 }
2602 \stex_debug:nn {notations}{
2603   Setting~default~notation~
2604   {\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str}~for~
2605   \l_stex_get_symbol_uri_str \
2606   \expandafter\meaning\csname
2607   l_stex_symdecl_\l_stex_get_symbol_uri_str _notations\endcsname
2608 }
2609 }{
2610   % todo throw error
2611 }
2612 \stex_smsmode_do:
2613 }
2614
2615 \cs_new_protected:Nn \stex_copy_notations:nn {
2616   \stex_debug:nn {notations}{
2617     Copying~notations~from~#2~to~#1\
2618     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
2619   }
2620   \tl_clear:N \l_tmpa_tl
2621   \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2622     \tl_put_right:Nn \l_tmpa_tl { {## #1} }
2623   }
2624   \seq_map_inline:cn {l_stex_symdecl_#2_notations}{
2625     \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2626     \edef \l_tmpa_tl {
2627       \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2628       \exp_after:wN\exp_after:wN\exp_after:wN {
2629         \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2630       }
2631     }
2632   \exp_args:Nx
2633   \stex_do_aftergroup:n {
2634     \seq_put_right:cn{l_stex_symdecl_#1_notations}{##1}
2635     \cs_generate_from_arg_count:cNnn {
2636       stex_notation_ #1 \c_hash_str ##1 _cs
2637     } \cs_set:Npn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }{
2638       \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl}
2639     }
2640   }
2641 }
2642 }
2643
2644 \NewDocumentCommand \copynotation {m m} {
2645   \stex_get_symbol:n { #1 }
2646   \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2647   \stex_get_symbol:n { #2 }

```

```

2648 \exp_args:Noo
2649 \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2650 \exp_args:Nx \stex_add_import_to_current_module:n{
2651   \stex_copy_notations:nn {\l_tmpa_str} {\l_stex_get_symbol_uri_str}
2652 }
2653 \stex_smsmode_do:
2654 }
2655

```

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

**\symdef**

```

2656 \keys_define:nn { stex / symdef } {
2657   name .str_set_x:N = \l_stex_symdecl_name_str ,
2658   local .bool_set:N = \l_stex_symdecl_local_bool ,
2659   args .str_set_x:N = \l_stex_symdecl_args_str ,
2660   type .tl_set:N = \l_stex_symdecl_type_tl ,
2661   def .tl_set:N = \l_stex_symdecl_definiens_tl ,
2662   op .tl_set:N = \l__stex_notation_op_tl ,
2663   lang .str_set_x:N = \l__stex_notation_lang_str ,
2664   variant .str_set_x:N = \l__stex_notation_variant_str ,
2665   prec .str_set_x:N = \l__stex_notation_prec_str ,
2666   unknown .code:n = \str_set:Nx
2667     \l__stex_notation_variant_str \l_keys_key_str
2668 }
2669
2670 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2671   \str_clear:N \l_stex_symdecl_name_str
2672   \str_clear:N \l_stex_symdecl_args_str
2673   \bool_set_false:N \l_stex_symdecl_local_bool
2674   \tl_clear:N \l_stex_symdecl_type_tl
2675   \tl_clear:N \l_stex_symdecl_definiens_tl
2676   \str_clear:N \l__stex_notation_lang_str
2677   \str_clear:N \l__stex_notation_variant_str
2678   \str_clear:N \l__stex_notation_prec_str
2679   \tl_clear:N \l__stex_notation_op_tl
2680
2681   \keys_set:nn { stex / symdef } { #1 }
2682 }
2683
2684 \NewDocumentCommand \symdef { 0{} m } {
2685   \__stex_notation_symdef_args:n { #1 }
2686   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2687   \stex_symdecl_do:n { #2 }
2688   \exp_args:Nx \stex_notation_do:nn {
2689     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2690   }
2691 }
2692 \stex_deactivate_macro:Nn \symdef {module~environments}

```

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

```

2693 \endpackage

```

## Chapter 31

# STEX -Terms Implementation

```
2694 <*package>
2695
2696 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2697
2698 <@@=stex_terms>
2699
2700 Warnings and error messages
2701 \msg_new:nnn{stex}{error/nonotation}{
2702   Symbol~#1~invoked,~but~has~no~notation~#2!
2703 }
2704 \msg_new:nnn{stex}{error/notationarg}{
2705   Error~in~parsing~notation~#1
2706 }
2707 \msg_new:nnn{stex}{error/noop}{
2708   Symbol~#1~has~no~operator~notation~for~notation~#2
2709 }
```

### 31.1 Symbol Invocations

Arguments:

```
2709 \keys_define:nn { stex / terms } {
2710   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2711   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2712   unknown .code:n = \str_set:Nx
2713     \l__stex_terms_variant_str \l_keys_key_str
2714 }
2715
2716 \cs_new_protected:Nn \__stex_terms_args:n {
2717   \str_clear:N \l__stex_terms_lang_str
2718   \str_clear:N \l__stex_terms_variant_str
2719   \str_clear:N \l__stex_terms_prec_str
2720   \tl_clear:N \l__stex_terms_op_tl
2721
2722   \keys_set:nn { stex / terms } { #1 }
```

2723 }

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

```
2724 \cs_new_protected:Nn \stex_invoke_symbol:n {
2725   \if_mode_math:
2726     \exp_after:wN \__stex_terms_invoke_math:n
2727   \else:
2728     \exp_after:wN \__stex_terms_invoke_text:n
2729   \fi: { #1 }
2730 }
```

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

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

```
2731 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2732   \peek_charcode_remove:NTF ! {
2733     \peek_charcode:NTF [ {
2734       \__stex_terms_invoke_op:nw { #1 }
2735     }{
2736       \peek_charcode_remove:NTF ! {
2737         \peek_charcode:NTF [ {
2738           \__stex_terms_invoke_op_custom:nw
2739         }{
2740           % TODO throw error
2741         }
2742       }{
2743         \__stex_terms_invoke_op:nw { #1 } []
2744       }
2745     }
2746   }{
2747     \peek_charcode_remove:NTF * {
2748       \__stex_terms_invoke_text:n { #1 }
2749     }{
2750       \peek_charcode:NTF [ {
2751         \__stex_terms_invoke_math:nw { #1 }
2752       }{
2753         \__stex_terms_invoke_math:nw { #1 } []
2754       }
2755     }
2756   }
2757 }
```

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

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

```
2758 \cs_new_protected:Npn \__stex_terms_invoke_op_custom:nw #1 [#2] {
2759   \stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
2760     \stex_highlight_term:nn{#1}{#2}
2761   }
2762 }
```

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

\\_stex\_terms\_invoke\_op:nw

```

2763 \cs_new_protected:Npn \_stex_terms_invoke_op:nw #1 [#2] {
2764   \_stex_terms_args:n { #2 }
2765   \cs_if_exist:cTF {
2766     stex_op_notation_ #1 \c_hash_str
2767     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2768   }{
2769     \csname stex_op_notation_ #1 \c_hash_str
2770       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2771     \endcsname
2772   }{
2773     \msg_error:nnxx{stex}{error/noop}{#1}{\l__stex_terms_variant_str \c_hash_str \l__stex_te
2774   }
2775 }

```

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

\\_stex\_terms\_invoke\_math:nw

```

2776 \cs_new_protected:Npn \_stex_terms_invoke_math:nw #1 [#2] {
2777   \_stex_terms_args:n { #2 }
2778   \seq_if_empty:cTF {
2779     l_stex_symdecl_ #1 _notations
2780   } {
2781     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
2782   } {
2783     \seq_if_in:cxTF {
2784       l_stex_symdecl_ #1 _notations
2785     }
2786     { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
2787       \str_set:Nn \l_stex_current_symbol_str { #1 }
2788       \stex_debug:nn{terms}{Using~
2789         #1\c_hash_str\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str \\
2790         \expandafter\meaning\csname stex_notation_ #1 \c_hash_str
2791         \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2792         _cs\endcsname
2793       }
2794       \use:c{
2795         stex_notation_ #1 \c_hash_str
2796         \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2797         _cs
2798       }
2799     }{
2800       \str_if_empty:NTF \l__stex_terms_variant_str {
2801         \str_if_empty:NTF \l__stex_terms_lang_str {
2802           \seq_get_left:cN {
2803             l_stex_symdecl_ #1 _notations
2804           } \l_tmpa_str
2805           \str_set:Nn \l_stex_current_symbol_str { #1 }
2806           \stex_debug:nn{terms}{Using~
2807             #1\c_hash_str\l_tmpa_str \\
2808             \expandafter\meaning\csname stex_notation_ #1 \c_hash_str
2809             \l_tmpa_str
2810             _cs\endcsname
2811           }

```

```

2812         \use:c{
2813             stex_notation_ #1 \c_hash_str \l_tmpa_str
2814             _cs
2815         }
2816     }{
2817         \msg_error:nnxx{stex}{error/nonotation}{#1}{
2818             ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2819         }
2820     }
2821 }{
2822     \msg_error:nnxx{stex}{error/nonotation}{#1}{
2823         ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2824     }
2825 }
2826 }
2827 }
2828 }

```

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

`\__stex_terms_invoke_text:n`

```

2829 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
2830     \peek_charcode_remove:NTF ! {
2831         \stex_term_custom:nn { #1 } { }
2832     }{
2833         \prop_set_eq:Nc \l_tmpa_prop {
2834             l_stex_symdecl_ #1 _prop
2835         }
2836         \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2837         \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2838     }
2839 }

```

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

## 31.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
2840 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2841 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2842 \int_new:N \l__stex_terms_downprec
2843 \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
2844 \tl_set:Nn \l__stex_terms_left_bracket_str (
2845 \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

```
2846 \cs_new_protected:Nn \_stex_terms_maybe_brackets:nn {
2847   \bool_if:NTF \l__stex_terms_brackets_done_bool {
2848     \bool_set_false:N \l__stex_terms_brackets_done_bool
2849     #2
2850   } {
2851     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
2852       \bool_if:NTF \l_stex_inarray_bool { #2 }{
2853         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
2854         \dobrackets { #2 }
2855       }
2856     }{ #2 }
2857   }
2858 }
```

(End definition for \\_stex\_terms\_maybe\_brackets:nn.)

\dobrackets

```
2859 \bool_new:N \l__stex_terms_brackets_done_bool
2860 %\RequirePackage{scalerel}
2861 \cs_new_protected:Npn \dobrackets #1 {
2862   %\ThisStyle{\if D@m@switch
2863   %   \exp_args:Nnx \use:nn
2864   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
2865   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
2866   %   \else
2867     \exp_args:Nnx \use:nn
2868     {
2869       \bool_set_true:N \l__stex_terms_brackets_done_bool
2870       \int_set:Nn \l__stex_terms_downprec \infprec
2871       \l__stex_terms_left_bracket_str
2872       #1
2873     }
2874     {
2875       \bool_set_false:N \l__stex_terms_brackets_done_bool
2876       \l__stex_terms_right_bracket_str
2877       \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2878     }
2879   %\fi}
2880 }
```

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

\withbrackets

```
2881 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2882   \exp_args:Nnx \use:nn
2883   {
2884     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
2885     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
2886     #3
2887   }
2888   {
2889     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
2890     {\l__stex_terms_left_bracket_str}
```

```

2891 \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
2892 {\l__stex_terms_right_bracket_str}
2893 }
2894 }

```

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

## `\STEXinvisible`

```

2895 \cs_new_protected:Npn \STEXinvisible #1 {
2896   \stex_annotate_invisible:n { #1 }
2897 }

```

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

OMDoc terms:

## `\_stex_term_math_oms:nnnn`

```

2898 \cs_new_protected:Nn \_stex_term_oms:nnn {
2899   \stex_annotate:nnn{ OMID }{ #2 }{
2900     \stex_highlight_term:nn { #1 } { #3 }
2901   }
2902 }
2903
2904 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2905   \__stex_terms_maybe_brackets:nn { #3 }{
2906     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2907   }
2908 }

```

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

## `\_stex_term_math_oma:nnnn`

```

2909 \cs_new_protected:Nn \_stex_term_oma:nnn {
2910   \stex_annotate:nnn{ OMA }{ #2 }{
2911     \stex_highlight_term:nn { #1 } { #3 }
2912   }
2913 }
2914
2915 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2916   \__stex_terms_maybe_brackets:nn { #3 }{
2917     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2918   }
2919 }

```

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

## `\_stex_term_math_omb:nnnn`

```

2920 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2921   \stex_annotate:nnn{ OMBIND }{ #2 }{
2922     \stex_highlight_term:nn { #1 } { #3 }
2923   }
2924 }
2925
2926 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2927   \__stex_terms_maybe_brackets:nn { #3 }{
2928     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }

```

```

2929 }
2930 }

```

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

`\_stex_term_math_arg:nnn`

```

2931 \cs_new_protected:Nn \_stex_term_arg:nn {
2932   \stex_unhighlight_term:n {
2933     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2934   }
2935 }
2936 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
2937   \exp_args:Nnx \use:nn
2938     { \int_set:Nn \l__stex_terms_downprec { #2 }
2939       \_stex_term_arg:nn { #1 }{ #3 }
2940     }
2941     { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2942     }

```

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

`\_stex_term_math_assoc_arg:nnnn`

```

2943 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
2944   % TODO sequences
2945   \clist_set:Nn \l_tmpa_clist{ #3 }
2946   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
2947     \tl_set:Nn \l_tmpa_tl { #3 }
2948   }{
2949     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
2950     \clist_reverse:N \l_tmpa_clist
2951     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
2952
2953     \clist_map_inline:Nn \l_tmpa_clist {
2954       \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
2955         \exp_args:Nno
2956           \l_tmpa_cs { ##1 } \l_tmpa_tl
2957       }
2958     }
2959   }
2960   \exp_args:Nnno
2961     \_stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2962 }

```

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

`\stex_term_custom:nn`

```

2963 \cs_new_protected:Nn \stex_term_custom:nn {
2964   \str_set:Nn \l__stex_terms_custom_uri { #1 }
2965   \str_set:Nn \l_tmpa_str { #2 }
2966   \tl_clear:N \l_tmpa_tl
2967   \int_zero:N \l_tmpa_int
2968   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2969   \__stex_terms_custom_loop:
2970 }

```

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

\\_\_stex\_terms\_custom\_loop:

```

2971 \cs_new_protected:Nn \__stex_terms_custom_loop: {
2972   \bool_set_false:N \l_tmpa_bool
2973   \bool_while_do:nn {
2974     \str_if_eq_p:ee X {
2975       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2976     }
2977   }{
2978     \int_incr:N \l_tmpa_int
2979   }
2980
2981   \peek_charcode:NTF [ {
2982     % notation/text component
2983     \__stex_terms_custom_component:w
2984   } {
2985     \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2986       % all arguments read => finish
2987       \__stex_terms_custom_final:
2988     } {
2989       % arguments missing
2990       \peek_charcode_remove:NTF * {
2991         % invisible, specific argument position or both
2992         \peek_charcode:NTF [ {
2993           % visible specific argument position
2994           \__stex_terms_custom_arg:wn
2995         } {
2996           % invisible
2997           \peek_charcode_remove:NTF * {
2998             % invisible specific argument position
2999             \__stex_terms_custom_arg_inv:wn
3000           } {
3001             % invisible next argument
3002             \__stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
3003           }
3004         } {
3005           % next normal argument
3006           \__stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
3007         }
3008       }
3009     }
3010   }
3011 }

```

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

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

```

3012 \cs_new_protected:Npn \__stex_terms_custom_arg_inv:wn [ #1 ] #2 {
3013   \bool_set_true:N \l_tmpa_bool
3014   \__stex_terms_custom_arg:wn [ #1 ] { #2 }
3015 }

```

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

\\_stex\_terms\_custom\_arg:wn

```

3016 \cs_new_protected:Npn \_stex_terms_custom_arg:wn [ #1 ] #2 {
3017   \str_set:Nx \l_tmpb_str {
3018     \str_item:Nn \l_tmpa_str { #1 }
3019   }
3020   \str_case:VnTF \l_tmpb_str {
3021     { X } {
3022       \msg_error:nnx{stex}{error/notationarg}{\_stex_terms_custom_uri}
3023     }
3024     { i } { \_stex_terms_custom_set_X:n { #1 } }
3025     { b } { \_stex_terms_custom_set_X:n { #1 } }
3026     { a } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
3027     { B } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
3028   }{}{
3029     \msg_error:nnx{stex}{error/notationarg}{\_stex_terms_custom_uri}
3030   }
3031
3032   \bool_if:nTF \l_tmpa_bool {
3033     \tl_put_right:Nx \l_tmpa_tl {
3034       \stex_annotate_invisible:n {
3035         \_stex_term_arg:nn { \int_eval:n { #1 } }
3036         \exp_not:n { { #2 } }
3037       }
3038     }
3039   } {
3040     \tl_put_right:Nx \l_tmpa_tl {
3041       \_stex_term_arg:nn { \int_eval:n { #1 } }
3042       \exp_not:n { { #2 } }
3043     }
3044   }
3045
3046   \_stex_terms_custom_loop:
3047 }

```

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

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

```

3048 \cs_new_protected:Nn \_stex_terms_custom_set_X:n {
3049   \str_set:Nx \l_tmpa_str {
3050     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
3051     X
3052     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
3053   }
3054 }

```

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

\\_stex\_terms\_custom\_component:

```

3055 \cs_new_protected:Npn \_stex_terms_custom_component:w [ #1 ] {
3056   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
3057   \_stex_terms_custom_loop:
3058 }

```

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

`\_stex_terms_custom_final:`

```

3059 \cs_new_protected:Nn \_stex_terms_custom_final: {
3060   \int_compare:nNnTF \l_tmpb_int = 0 {
3061     \exp_args:Nnno \_stex_term_oms:nnn
3062   }{
3063     \str_if_in:NnTF \l_tmpa_str {b} {
3064       \exp_args:Nnno \_stex_term_ombind:nnn
3065     } {
3066       \exp_args:Nnno \_stex_term_oma:nnn
3067     }
3068   }
3069   { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
3070 }

```

(End definition for `\_stex_terms_custom_final:.`)

`\symref`  
`\symname`

```

3071 \NewDocumentCommand \symref { m m }{
3072   \let\compemph_uri_prev:\compemph@uri
3073   \let\compemph@uri\symrefemph@uri
3074   \STEXsymbol{#1}![#2]
3075   \let\compemph@uri\compemph_uri_prev:
3076 }
3077
3078 \keys_define:nn { stex / symname } {
3079   post      .str_set_x:N      = \l_stex_symname_post_str
3080 }
3081
3082 \cs_new_protected:Nn \stex_symname_args:n {
3083   \str_clear:N \l_stex_symname_post_str
3084   \keys_set:nn { stex / symname } { #1 }
3085 }
3086
3087 \NewDocumentCommand \symname { 0{} m }{
3088   \stex_symname_args:n { #1 }
3089   \stex_get_symbol:n { #2 }
3090   \str_set:Nx \l_tmpa_str {
3091     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3092   }
3093   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3094
3095   \let\compemph_uri_prev:\compemph@uri
3096   \let\compemph@uri\symrefemph@uri
3097   \exp_args:NNx \use:nn
3098   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }![
3099     \l_tmpa_str \l_stex_symname_post_str
3100   ] }
3101   \let\compemph@uri\compemph_uri_prev:
3102 }

```

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

## 31.3 Notation Components

3103  $\langle @@=\text{stex\_notationcomps} \rangle$

$\text{\stex\_highlight\_term:nn}$

```

3104
3105 \str_new:N \l_stex_current_symbol_str
3106 \cs_new_protected:Nn \stex_highlight_term:nn {
3107   \exp_args:Nnx
3108   \use:nn {
3109     \str_set:Nx \l_stex_current_symbol_str { #1 }
3110     #2
3111   } {
3112     \str_set:Nx \exp_not:N \l_stex_current_symbol_str
3113     { \l_stex_current_symbol_str }
3114   }
3115 }
3116
3117 \cs_new_protected:Nn \stex_unhighlight_term:n {
3118   % \latexml_if:TF {
3119   %   #1
3120   % } {
3121   %   \rustex_if:TF {
3122   %     #1
3123   %   } {
3124     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
3125   %   }
3126   % }
3127 }
```

(End definition for  $\text{\stex\_highlight\_term:nn}$ . This function is documented on page 40.)

```

\comp
\compemph@uri 3128 \cs_new_protected:Npn \comp #1 {
\compemph 3129   \str_if_empty:NF \l_stex_current_symbol_str {
\defemph 3130     \rustex_if:TF {
\defemph@uri 3131       \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
\symrefemph 3132     }{
\symrefemph@uri 3133       \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3134     }
3135   }
3136 }
3137
3138 \cs_new_protected:Npn \compemph@uri #1 #2 {
3139   \compemph{ #1 }
3140 }
3141
3142
3143 \cs_new_protected:Npn \compemph #1 {
3144   #1
3145 }
3146
3147 \cs_new_protected:Npn \defemph@uri #1 #2 {
3148   \defemph{#1}
3149 }
```

```

3150
3151 \cs_new_protected:Npn \defemph #1 {
3152   \textbf{#1}
3153 }
3154
3155 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3156   \symrefemph{#1}
3157 }
3158
3159 \cs_new_protected:Npn \symrefemph #1 {
3160   \textbf{#1}
3161 }

```

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

### `\ellipses`

```

3162 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix
\parrayline
\parraylineh
\parraycell
3163 \bool_new:N \l_stex_inparray_bool
3164 \bool_set_false:N \l_stex_inparray_bool
3165 \NewDocumentCommand \parray { m m } {
3166   \begingroup
3167   \bool_set_true:N \l_stex_inparray_bool
3168   \begin{array}{#1}
3169     #2
3170   \end{array}
3171   \endgroup
3172 }
3173
3174 \NewDocumentCommand \prmatrix { m } {
3175   \begingroup
3176   \bool_set_true:N \l_stex_inparray_bool
3177   \begin{matrix}
3178     #1
3179   \end{matrix}
3180   \endgroup
3181 }
3182
3183 \def \maybepline {
3184   \bool_if:NT \l_stex_inparray_bool {\hline}
3185 }
3186
3187 \def \parrayline #1 #2 {
3188   #1 #2 \bool_if:NT \l_stex_inparray_bool {\}
3189 }
3190
3191 \def \pmrow #1 { \parrayline{#1} }
3192
3193 \def \parraylineh #1 #2 {
3194   #1 #2 \bool_if:NT \l_stex_inparray_bool {\hline}
3195 }
3196

```



```

3197 \def \parraycell #1 {
3198   #1 \bool_if:NT \l_stex_inarray_bool {&}
3199 }

```

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

```

3200 \endpackage

```

## Chapter 32

# STEX -Structural Features Implementation

```
3201 <*package>
3202
3203 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3204
3205 <@@=stex_features>
3206
3207     Warnings and error messages
3208 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3209     Symbol~#1~can~not~be~assigned~in~copymodule~#2
3210 }
3211 \msg_new:nnn{stex}{error/interpretmodule/noddefinien}{
3212     Symbol~#1~not~assigned~in~interpretmodule~#2
3213 }
```

### 32.1 Imports with modification

```
3213 \cs_new_protected:Nn \stex_get_symbol_in_copymodule:n {
3214     \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3215         \__stex_features_get_symbol_from_cs:n { #1 }
3216     }{
3217         % argument is a string
3218         % is it a command name?
3219         \cs_if_exist:cTF { #1 }{
3220             \cs_set_eq:Nc \l_tmpa_tl { #1 }
3221             \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3222             \str_if_empty:NNTF \l_tmpa_str {
3223                 \exp_args:Nx \cs_if_eq:NNTF {
3224                     \tl_head:N \l_tmpa_tl
3225                 } \stex_invoke_symbol:n {
3226                     \exp_args:No \__stex_features_get_symbol_from_cs:n { \use:c { #1 } }
3227                 }{
3228                     \__stex_features_get_symbol_from_string:n { #1 }
3229                 }
3230             }
3231         }
3232     }
```

```

3229     }
3230   } {
3231     \__stex_features_get_symbol_from_string:n { #1 }
3232   }
3233   }{
3234     % argument is not a command name
3235     \__stex_features_get_symbol_from_string:n { #1 }
3236     % \l_stex_all_symbols_seq
3237   }
3238 }
3239 }
3240
3241 \cs_new_protected:Nn \__stex_features_get_symbol_from_string:n {
3242   \str_set:Nn \l_tmpa_str { #1 }
3243   \bool_set_false:N \l_tmpa_bool
3244   \bool_if:NF \l_tmpa_bool {
3245     \tl_set:Nn \l_tmpa_tl {
3246       \msg_set:nnn{stex}{error/unknownsymbol}{
3247         No~symbol~#1~found!
3248       }
3249       \msg_error:nn{stex}{error/unknownsymbol}
3250     }
3251     \str_set:Nn \l_tmpa_str { #1 }
3252     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3253     \seq_map_inline:Nn \l__stex_features_copymodule_fields_seq {
3254       \str_set:Nn \l_tmpb_str { ##1 }
3255       \str_if_eq:eeT { \l_tmpa_str } {
3256         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3257       } {
3258         \seq_map_break:n {
3259           \tl_set:Nn \l_tmpa_tl {
3260             \str_set:Nn \l_stex_get_symbol_uri_str {
3261               ##1
3262             }
3263             \__stex_features_get_symbol_check:
3264           }
3265         }
3266       }
3267     }
3268     \l_tmpa_tl
3269   }
3270 }
3271
3272 \cs_new_protected:Nn \__stex_features_get_symbol_from_cs:n {
3273   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3274   { \tl_tail:N \l_tmpa_tl }
3275   \tl_if_single:NTF \l_tmpa_tl {
3276     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3277       \exp_after:wN \str_set:Nn \exp_after:wN
3278       \l_stex_get_symbol_uri_str \l_tmpa_tl
3279       \__stex_features_get_symbol_check:
3280     }{
3281       % TODO
3282       % tail is not a single group

```

```

3283     }
3284 }{
3285     % TODO
3286     % tail is not a single group
3287 }
3288 }
3289
3290 \cs_new_protected:Nn \__stex_features_get_symbol_check: {
3291     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq {?} \l_stex_get_symbol_uri_str
3292     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 3 {
3293         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
3294         \str_set:Nx \l_tmpa_str {\seq_use:Nn \l_tmpa_seq ?}
3295         \seq_if_in:NoF \l__stex_features_copymodule_modules_seq \l_tmpa_str {
3296             \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3297                 \l_stex_current_copymodule_name_str\Allowed:~\seq_use:Nn \l__stex_features_copymodule_modules_seq \l_tmpa_str
3298             }
3299         }
3300     }{
3301         \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3302             \l_stex_current_copymodule_name_str~(inexplicably)
3303         }
3304     }
3305 }
3306
3307 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
3308     \stex_import_module_uri:nn { #1 } { #2 }
3309     \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3310     \stex_import_require_module:nnnn
3311     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3312     { \l_stex_import_path_str } { \l_stex_import_name_str }
3313     \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3314     \seq_set_eq:NN \l__stex_features_copymodule_modules_seq \l_stex_collect_imports_seq
3315     \seq_clear:N \l__stex_features_copymodule_fields_seq
3316     \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3317         \seq_map_inline:cn {c_stex_module_###1_constants}{
3318             \exp_args:NNx \seq_put_right:Nn \l__stex_features_copymodule_fields_seq {
3319                 ###1 ? #####1
3320             }
3321         }
3322     }
3323     \seq_clear:N \l_tmpa_seq
3324     \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3325         name      = \l_stex_current_copymodule_name_str ,
3326         module    = \l_stex_current_module_str ,
3327         from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3328         includes  = \l_tmpa_seq ,
3329         fields    = \l_tmpa_seq
3330     }
3331     \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3332         as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3333     \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_features_copymodule_modules_seq
3334     \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_features_copymodule_fields_seq {,~}
3335     \stex_if_smsmode:F {
3336         \begin{stex_annotate_env} {#4} {

```

```

3337     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3338   }
3339   \stex_annotate_invisible:nnn{from}{\l_stex_import_ns_str ?\l_stex_import_name_str}{\}
3340 }
3341 \bool_set_eq:NN \l__stex_features_oldhtml_bool \l_stex_html_do_output_bool
3342 \bool_set_false:N \l_stex_html_do_output_bool
3343 }
3344 \cs_new_protected:Nn \stex_copymodule_end:n {
3345   \def \l_tmpa_cs ##1 ##2 {#1}
3346   \bool_set_eq:NN \l_stex_html_do_output_bool \l__stex_features_oldhtml_bool
3347   \tl_clear:N \l_tmpa_tl
3348   \tl_clear:N \l_tmpb_tl
3349   \prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3350   \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3351     \seq_map_inline:cn {c_stex_module_##1_constants}{
3352       \tl_clear:N \l_tmpc_tl
3353       \l_tmpa_cs{##1}{####1}
3354       \str_if_exist:cTF {l__stex_features_copymodule_##1?####1_name_str} {
3355         \tl_put_right:Nx \l_tmpa_tl {
3356           \prop_set_from_keyval:cn {
3357             l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_#
3358           }{
3359             \exp_after:wN \prop_to_keyval:N \csname
3360               l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_#
3361             \endcsname
3362           }
3363           \seq_clear:c {
3364             l_stex_symdecl_
3365             \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_name_s
3366             _notations
3367           }
3368         }
3369         \tl_put_right:Nx \l_tmpc_tl {
3370           \stex_copy_notations:nn {\l_stex_current_module_str ? \use:c{l__stex_features_copy
3371           \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_features_copymodule_##1?####1_n
3372         }
3373         \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{l__stex_features_
3374         \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3375           \tl_put_right:Nx \l_tmpc_tl {
3376             \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?#
3377           }
3378           \tl_put_right:Nx \l_tmpa_tl {
3379             \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3380             \stex_invoke_symbol:n {
3381               \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_na
3382             }
3383           }
3384         }
3385       }
3386     }{
3387       \tl_put_right:Nx \l_tmpc_tl {
3388         \stex_copy_notations:nn {\l_stex_current_module_str ? \l_stex_current_copymodule_n
3389       }
3390       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1_prop}

```

```

3391 \prop_put:Nnx \l_tmpa_prop { name }{ \l_stex_current_copymodule_name_str / ####1 }
3392 \prop_put:Nnx \l_tmpa_prop { module }{ \l_stex_current_module_str }
3393 \tl_put_right:Nx \l_tmpa_tl {
3394   \prop_set_from_keyval:cn {
3395     l_stex_symdecl\l_stex_current_module_str ? \l_stex_current_copymodule_name_str
3396   }{
3397     \prop_to_keyval:N \l_tmpa_prop
3398   }
3399   \seq_clear:c {
3400     l_stex_symdecl_
3401     \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3402     _notations
3403   }
3404 }
3405 \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str}
3406 \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3407   \tl_put_right:Nx \l_tmpc_tl {
3408     \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}
3409   }
3410   \tl_put_right:Nx \l_tmpa_tl {
3411     \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3412       \stex_invoke_symbol:n {
3413         \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3414       }
3415     }
3416   }
3417 }
3418 }
3419 \tl_if_exist:cT {l__stex_features_copymodule_##1?####1_def_tl}{
3420   \tl_put_right:Nx \l_tmpc_tl {
3421     \stex_annotate_invisible:nnn{definiens}{\use:c{l__stex_features_copymodule_##1?####1_def_tl}}
3422   }
3423 }
3424 \tl_put_right:Nx \l_tmpb_tl {
3425   \stex_annotate:nnn{assignment} {##1?####1} { \l_tmpc_tl }
3426 }
3427 }
3428 }
3429 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3430 \tl_put_left:Nx \l_tmpa_tl {
3431   \prop_set_from_keyval:cn {
3432     l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _prop
3433   }{
3434     \prop_to_keyval:N \l_stex_current_copymodule_prop
3435   }
3436 }
3437 \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl
3438 \stex_debug:nn{copymodule}{result:\meaning \l_tmpa_tl}
3439 \exp_args:Nx \stex_do_aftergroup:n {
3440   \exp_args:No \exp_not:n \l_tmpa_tl
3441 }
3442 \l_tmpb_tl
3443 \stex_if_smsmode:F {
3444   \end{stex_annotate_env}

```

```

3445 }
3446 }
3447
3448 \NewDocumentEnvironment {copymodule} { 0{} m m}{
3449   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ structure }
3450   \stex_deactivate_macro:Nn \symdecl {module~environments}
3451   \stex_deactivate_macro:Nn \symdef {module~environments}
3452   \stex_deactivate_macro:Nn \notation {module~environments}
3453   \stex_reactivate_macro:N \assign
3454   \stex_reactivate_macro:N \renamedec1
3455   \stex_reactivate_macro:N \donotcopy
3456   \stex_smsmode_do:
3457 }{
3458   \stex_copymodule_end:n {}
3459 }
3460
3461 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
3462   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ realization }
3463   \stex_deactivate_macro:Nn \symdecl {module~environments}
3464   \stex_deactivate_macro:Nn \symdef {module~environments}
3465   \stex_deactivate_macro:Nn \notation {module~environments}
3466   \stex_reactivate_macro:N \assign
3467   \stex_reactivate_macro:N \renamedec1
3468   \stex_reactivate_macro:N \donotcopy
3469   \stex_smsmode_do:
3470 }{
3471   \stex_copymodule_end:n {
3472     \tl_if_exist:cF {
3473       l__stex_features_copymodule_##1?##2_def_tl
3474     }{
3475       \msg_error:nxxx{stex}{error/interpretmodule/nodedefiniens}{
3476         ##1?##2
3477       }{\l_stex_current_copymodule_name_str}
3478     }
3479   }
3480 }
3481
3482 \NewDocumentCommand \donotcopy { 0{} m}{
3483   \stex_import_module_uri:nn { #1 } { #2 }
3484   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3485   \seq_map_inline:Nn \l_stex_collect_imports_seq {
3486     \seq_remove_all:Nn \l_stex_features_copymodule_modules_seq { ##1 }
3487     \seq_map_inline:cn {c_stex_module_##1_constants}{
3488       \seq_remove_all:Nn \l_stex_features_copymodule_fields_seq { ##1 ? #####1 }
3489       \bool_lazy_any_p:nT {
3490         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_name_str}}
3491         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_macroname_str}}
3492         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_def_tl}}
3493       }{
3494         % TODO throw error
3495       }
3496     }
3497   }
3498 }

```

```

3499 \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
3500 \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_ns_str ?\l_stex_import_name_str }
3501 \prop_put:Nnx \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
3502 }
3503
3504 \NewDocumentCommand \assign { m m }{
3505   \stex_get_symbol_in_copymodule:n {#1}
3506   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
3507   \tl_set:cn {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _def_tl}{#2}
3508 }
3509
3510 \keys_define:nn { stex / renamedec1 } {
3511   name .str_set_x:N = \l_stex_renamedec1_name_str
3512 }
3513 \cs_new_protected:Nn \__stex_features_renamedec1_args:n {
3514   \str_clear:N \l_stex_renamedec1_name_str
3515
3516   \keys_set:nn { stex / renamedec1 } { #1 }
3517 }
3518
3519 \NewDocumentCommand \renamedec1 { O{} m m }{
3520   \__stex_features_renamedec1_args:n { #1 }
3521   \stex_get_symbol_in_copymodule:n {#2}
3522   \stex_debug:nn{renamedec1}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
3523   \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
3524   \str_if_empty:NTF \l_stex_renamedec1_name_str {
3525     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3526       \l_stex_get_symbol_uri_str
3527     } }
3528   } {
3529     \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex_r
3530     \stex_debug:nn{renamedec1}{@~\l_stex_current_module_str ? \l_stex_renamedec1_name_str}
3531     \prop_set_eq:cc {l_stex_symdecl_
3532       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3533       _prop
3534     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
3535     \seq_set_eq:cc {l_stex_symdecl_
3536       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3537       _notations
3538     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
3539     \prop_put:cnx {l_stex_symdecl_
3540       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3541       _prop
3542     }{ name }{ \l_stex_renamedec1_name_str }
3543     \prop_put:cnx {l_stex_symdecl_
3544       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3545       _prop
3546     }{ module }{ \l_stex_current_module_str }
3547     \exp_args:NNx \seq_put_left:Nn \l__stex_features_copymodule_fields_seq {
3548       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3549     }
3550     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3551       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3552     } }

```



```

3553 }
3554 }
3555 %\NewDocumentCommand \notation_in_copymodules: { 0{} m } {
3556 % \stex_notation_args:n { #1 }
3557 % \tl_clear:N \l_stex_symdecl_definiens_tl
3558 % \stex_get_symbol_in_copymodule:n { #2 }
3559 % \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
3560 % % todo
3561 %}
3562 \stex_deactivate_macro:Nn \assign {copymodules}
3563 \stex_deactivate_macro:Nn \renamedekl {copymodules}
3564 \stex_deactivate_macro:Nn \donotcopy {copymodules}
3565
3566
3567 \seq_new:N \l_stex_implicit_morphisms_seq
3568 \NewDocumentCommand \implicitmorphism { 0{} m m }{
3569 \stex_import_module_uri:nn { #1 } { #2 }
3570 \stex_debug:nn{implicits}{
3571 Implicit~morphism:~
3572 \l_stex_module_ns_str ? \l__stex_features_name_str
3573 }
3574 \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
3575 \l_stex_module_ns_str ? \l__stex_features_name_str
3576 }{
3577 \msg_error:nnn{stex}{error/conflictingmodules}{
3578 \l_stex_module_ns_str ? \l__stex_features_name_str
3579 }
3580 }
3581
3582 % TODO
3583
3584
3585
3586 \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
3587 \l_stex_module_ns_str ? \l__stex_features_name_str
3588 }
3589 }
3590

```

## 32.2 The feature environment

structural@feature

```

3591
3592 \NewDocumentEnvironment{structural@feature}{ m m m }{
3593 \stex_if_in_module:F {
3594 \msg_set:nnn{stex}{error/nomodule}{
3595 Structural~Feature~has~to~occur~in~a~module:\\
3596 Feature~#2~of~type~#1\\
3597 In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
3598 }
3599 \msg_error:nn{stex}{error/nomodule}
3600 }
3601

```

```

3602 \str_set:Nx \l_stex_module_name_str {
3603   \prop_item:Nn \l_stex_current_module_prop
3604     { name } / #2 - feature
3605 }
3606
3607 \str_set:Nx \l_stex_module_ns_str {
3608   \prop_item:Nn \l_stex_current_module_prop
3609     { ns }
3610 }
3611
3612
3613 \str_clear:N \l_tmpa_str
3614 \seq_clear:N \l_tmpa_seq
3615 \tl_clear:N \l_tmpa_tl
3616 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
3617   origname = #2,
3618   name      = \l_stex_module_name_str ,
3619   ns        = \l_stex_module_ns_str ,
3620   imports   = \exp_not:o { \l_tmpa_seq } ,
3621   constants = \exp_not:o { \l_tmpa_seq } ,
3622   content   = \exp_not:o { \l_tmpa_tl } ,
3623   file      = \exp_not:o { \g_stex_currentfile_seq } ,
3624   lang      = \l_stex_module_lang_str ,
3625   sig       = \l_tmpa_str ,
3626   meta      = \l_tmpa_str ,
3627   feature   = #1 ,
3628 }
3629
3630 \stex_if_smsmode:F {
3631   \begin{stex_annotate_env}{ feature:#1 }{}
3632   \stex_annotate_invisible:nnn{header}{}{ #3 }
3633 }
3634 }{
3635   \str_set:Nx \l_tmpa_str {
3636     c_stex_feature_
3637     \prop_item:Nn \l_stex_current_module_prop { ns } ?
3638     \prop_item:Nn \l_stex_current_module_prop { name }
3639     _prop
3640   }
3641   \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
3642   \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
3643   \stex_if_smsmode:TF {
3644     \exp_args:Nx \stex_add_to_sms:n {
3645       \prop_gset_from_keyval:cn {
3646         c_stex_feature_
3647         \prop_item:Nn \l_stex_current_module_prop { ns } ?
3648         \prop_item:Nn \l_stex_current_module_prop { name }
3649         _prop
3650       } {
3651         origname = #2,
3652         name      = \prop_item:cn { \l_tmpa_str } { name } ,
3653         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
3654         imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
3655         constants = \prop_item:cn { \l_tmpa_str } { constants } ,

```

```

3656         content = \prop_item:cn { \l_tmpa_str } { content } ,
3657         file     = \prop_item:cn { \l_tmpa_str } { file } ,
3658         lang     = \prop_item:cn { \l_tmpa_str } { lang } ,
3659         sig      = \prop_item:cn { \l_tmpa_str } { sig } ,
3660         meta     = \prop_item:cn { \l_tmpa_str } { meta } ,
3661         feature  = \prop_item:cn { \l_tmpa_str } { feature }
3662     }
3663 }
3664 } {
3665     \end{stex_annotate_env}
3666 }
3667 }
3668

```

## 32.3 Features

structure

```

3669
3670 \prop_new:N \l_stex_all_structures_prop
3671
3672 \keys_define:nn { stex / features / structure } {
3673     name .str_set_x:N = \l__stex_features_structure_name_str ,
3674 }
3675
3676 \cs_new_protected:Nn \__stex_features_structure_args:n {
3677     \str_clear:N \l__stex_features_structure_name_str
3678     \keys_set:nn { stex / features / structure } { #1 }
3679 }
3680
3681 %\stex_new_feature:nnnn { structure } { 0{ } m } {
3682 % \__stex_features_structure_args:n { ##1 }
3683 % \str_if_empty:NT \l__stex_features_structure_name_str {
3684 %     \str_set:Nx \l__stex_features_structure_name_str { ##2 }
3685 % }
3686 %} {
3687 %
3688 %}
3689
3690 \NewDocumentEnvironment{mathstructure}{0{ } m }{
3691     \__stex_features_structure_args:n { #1 }
3692     \str_if_empty:NT \l__stex_features_structure_name_str {
3693         \str_set:Nx \l__stex_features_structure_name_str { #2 }
3694     }
3695     \exp_args:Nnnx
3696     \begin{structural@feature}{ structure }
3697         { \l__stex_features_structure_name_str }{}
3698         \seq_clear:N \l_tmpa_seq
3699         \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
3700     \stex_smsmode_do:
3701 }{
3702     \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
3703     \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
3704     \str_set:Nx \l_tmpa_str {

```

```

3705     \prop_item:Nn \l_stex_current_module_prop { ns } ?
3706     \prop_item:Nn \l_stex_current_module_prop { name }
3707   }
3708   \seq_map_inline:Nn \l_tmpa_seq {
3709     \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
3710   }
3711   \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
3712   \exp_args:Nnx
3713   \AddToHookNext { env / mathstructure / after }{
3714     \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}}{
3715       \_stex_term_math_oms:nnnn { \l_tmpa_str }{}{0}{}
3716     }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
3717     \STEXexport {
3718       \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3719       {\prop_item:Nn \l_stex_current_module_prop { origname }}
3720       {\l_tmpa_str}
3721       \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3722       {#2}{\l_tmpa_str}
3723     %   \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3724     %     \prop_item:Nn \l_stex_current_module_prop { origname },
3725     %     \l_tmpa_str
3726     %   }
3727     %   \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3728     %     #2,\l_tmpa_str
3729     %   }
3730     %   \tl_set:cx { #2 } {
3731     %     \stex_invoke_structure:n { \l_tmpa_str }
3732   }
3733 }
3734
3735 \end{structural@feature}
3736 % \g_stex_last_feature_prop
3737 }

```

\instantiate

```

3738 \seq_new:N \l__stex_features_structure_field_seq
3739 \str_new:N \l__stex_features_structure_field_str
3740 \str_new:N \l__stex_features_structure_def_tl
3741 \prop_new:N \l__stex_features_structure_prop
3742 \NewDocumentCommand \instantiate { m O{} m }{
3743   \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
3744   \prop_set_eq:Nc \l__stex_features_structure_prop {
3745     c_stex_feature_\l_tmpa_str _prop
3746   }
3747   \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
3748   \seq_map_inline:Nn \l__stex_features_structure_field_seq {
3749     \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
3750     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3751       \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
3752       \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
3753       {!} \l_tmpa_tl
3754       \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3755         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
3756         \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl

```

```

3757     \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3758   }{
3759     \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
3760     \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3761     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
3762       \l_tmpa_tl
3763     \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3764       \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
3765       \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3766     }{
3767       \tl_clear:N \l_tmpb_tl
3768     }
3769   }
3770 }{
3771   \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
3772   \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3773     \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
3774     \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
3775     \tl_clear:N \l_tmpa_tl
3776   }{
3777     % TODO throw error
3778   }
3779 }
3780 % \l_tmpa_str: name
3781 % \l_tmpa_tl: definiens
3782 % \l_tmpb_tl: notation
3783 \tl_if_empty:NT \l__stex_features_structure_field_str {
3784   % TODO throw error
3785 }
3786 \str_clear:N \l_tmpb_str
3787
3788 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3789 \seq_map_inline:Nn \l_tmpa_seq {
3790   \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
3791   \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
3792   \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
3793     \seq_map_break:n {
3794       \str_set:Nn \l_tmpb_str { ####1 }
3795     }
3796   }
3797 }
3798 \prop_get:cnN { l_stex_symdecl_ \l_tmpb_str _prop } {args}
3799   \l_tmpb_str
3800
3801 \tl_if_empty:NNTF \l_tmpb_tl {
3802   \tl_if_empty:NF \l_tmpa_tl {
3803     \exp_args:Nx \use:n {
3804       \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fe
3805     }
3806   }
3807 }{
3808   \tl_if_empty:NNTF \l_tmpa_tl {
3809     \exp_args:Nx \use:n {
3810       \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\

```

```

3811     }
3812
3813   }{
3814     \exp_args:Nx \use:n {
3815       \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
3816       \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
3817     }
3818   }
3819 }
3820 % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
3821 % \prop_item:Nn \l_stex_current_module_prop {name} ?
3822 % #3/\l__stex_features_structure_field_str
3823 % \par
3824 % \expandafter\present\csname
3825 %   \l_stex_symdecl_
3826 %   \prop_item:Nn \l_stex_current_module_prop {ns} ?
3827 %   \prop_item:Nn \l_stex_current_module_prop {name} ?
3828 %   #3/\l__stex_features_structure_field_str
3829 %   _prop
3830 % \endcsname
3831 }
3832
3833 \tl_clear:N \l__stex_features_structure_def_tl
3834
3835 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3836 \seq_map_inline:Nn \l_tmpa_seq {
3837   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3838   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3839   \exp_args:Nx \use:n {
3840     \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
3841
3842     }
3843   }
3844
3845   \prop_if_exist:cF {
3846     \l_stex_symdecl_
3847     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3848     \prop_item:Nn \l_stex_current_module_prop {name} ?
3849     #3/\l_tmpa_str
3850     _prop
3851   }{
3852     \prop_get:cnN { \l_stex_symdecl_ ##1 _prop } {args}
3853     \l_tmpb_str
3854     \exp_args:Nx \use:n {
3855       \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
3856     }
3857   }
3858 }
3859
3860 \symdecl*[type={\STEXsymbol{module-type}}{
3861   \_stex_term_math_oms:nnnn {
3862     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3863     \prop_item:Nn \l__stex_features_structure_prop {name}
3864   }-{}{0}-{}

```

```

3865   }}]{#3}
3866
3867   % TODO: -> sms file
3868
3869   \tl_set:cx{ #3 }{
3870     \stex_invoke_structure:nnn {
3871       \prop_item:Nn \l_stex_current_module_prop {ns} ?
3872       \prop_item:Nn \l_stex_current_module_prop {name} ? #3
3873     } {
3874       \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3875       \prop_item:Nn \l__stex_features_structure_prop {name}
3876     }
3877   }
3878   \stex_smsmode_do:
3879 }

```

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

\stex\_invoke\_structure:nnn

```

3880 % #1: URI of the instance
3881 % #2: URI of the instantiated module
3882 \cs_new_protected:Nn \stex_invoke_structure:nnn {
3883   \tl_if_empty:nTF{ #3 }{
3884     \prop_set_eq:Nc \l__stex_features_structure_prop {
3885       c_stex_feature_ #2 _prop
3886     }
3887     \tl_clear:N \l_tmpa_tl
3888     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3889     \seq_map_inline:Nn \l_tmpa_seq {
3890       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3891       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3892       \cs_if_exist:cT {
3893         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
3894       }{
3895         \tl_if_empty:NF \l_tmpa_tl {
3896           \tl_put_right:Nn \l_tmpa_tl {,}
3897         }
3898         \tl_put_right:Nx \l_tmpa_tl {
3899           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
3900         }
3901       }
3902     }
3903     \exp_args:No \mathstrut \l_tmpa_tl
3904   }{
3905     \stex_invoke_symbol:n{#1/#3}
3906   }
3907 }

```

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

3908 </package>

## Chapter 33

# STEX -Statements Implementation

```
3909 <*package>
3910
3911 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3912
3913 \protected\def\ignorespacesandpars{
3914   \begingroup\catcode13=10\relax
3915   \@ifnextchar\par{
3916     \endgroup\expandafter\ignorespacesandpars\@gobble
3917   }{
3918     \endgroup
3919   }
3920 }
3921
3922 <@@=stex_statements>
3923
3924   Warnings and error messages
```

\titleemph

```
3924 \def\titleemph#1{\textbf{#1}}
```

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

### 33.1 Definitions

definiendum

```
3925 \keys_define:nn {stex / definiendum }{
3926   post      .tl_set:N      = \l__stex_statements_definiendum_post_tl,
3927   root      .str_set_x:N   = \l__stex_statements_definiendum_root_str,
3928   gfa       .str_set_x:N   = \l__stex_statements_definiendum_gfa_str
3929 }
3930 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
3931   \str_clear:N \l__stex_statements_definiendum_root_str
3932   \tl_clear:N \l__stex_statements_definiendum_post_tl
3933   \str_clear:N \l__stex_statements_definiendum_gfa_str
```



```

3934 \keys_set:nn { stex / definiendum }{ #1 }
3935 }
3936 \NewDocumentCommand \definiendum { 0{} m m } {
3937   \__stex_statements_definiendum_args:n { #1 }
3938   \stex_get_symbol:n { #2 }
3939   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3940   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
3941     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
3942       \tl_set:Nn \l_tmpa_tl { #3 }
3943     } {
3944       \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
3945       \tl_set:Nn \l_tmpa_tl {
3946         \l__stex_statements_definiendum_root_str\l__stex_statements_definiendum_post_tl
3947       }
3948     }
3949   } {
3950     \tl_set:Nn \l_tmpa_tl { #3 }
3951   }
3952
3953   % TODO root
3954   \rustex_if:TF {
3955     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
3956   } {
3957     \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
3958   }
3959 }
3960 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

## definame

```

3961
3962 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3963
3964 \NewDocumentCommand \definame { 0{} m } {
3965   \__stex_statements_definiendum_args:n { #1 }
3966   % TODO: root
3967   \stex_get_symbol:n { #2 }
3968   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3969   \str_set:Nx \l_tmpa_str {
3970     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3971   }
3972   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3973   \rustex_if:TF {
3974     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3975       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3976     }
3977   } {
3978     \defemph@uri {
3979       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3980     } { \l_stex_get_symbol_uri_str }
3981   }
3982 }
3983 \stex_deactivate_macro:Nn \definame {definition~environments}

```

```

3984
3985 \NewDocumentCommand \Definame { 0{} m } {
3986   \__stex_statements_definiendum_args:n { #1 }
3987   \stex_get_symbol:n { #2 }
3988   \str_set:Nx \l_tmpa_str {
3989     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3990   }
3991   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3992   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3993   \rustex_if:TF {
3994     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3995       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3996     }
3997   } {
3998     \defemph@uri {
3999       \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
4000     } { \l_stex_get_symbol_uri_str }
4001   }
4002 }
4003 \stex_deactivate_macro:Nn \Definame {definition-environments}
4004
4005 \NewDocumentCommand \Symname { 0{} m }{
4006   \stex_symname_args:n { #1 }
4007   \stex_get_symbol:n { #2 }
4008   \str_set:Nx \l_tmpa_str {
4009     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4010   }
4011   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4012   \let\compemph_uri_prev:\compemph@uri
4013   \let\compemph@uri\symrefemph@uri
4014   \exp_args:NNx \use:nn
4015   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }! [
4016     \exp_after:wN \stex_capitalize:n \l_tmpa_str
4017     \l_stex_symname_post_str
4018   ] }
4019   \let\compemph@uri\compemph_uri_prev:
4020 }

```

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

#### sdefinition

```

4021
4022 \keys_define:nn {stex / sdefinition }{
4023   type      .str_set_x:N = \sdefinitiontype,
4024   id        .str_set_x:N = \sdefinitionid,
4025   name      .str_set_x:N = \sdefinitionname,
4026   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4027   title     .tl_set:N     = \sdefinitiontitle
4028 }
4029 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
4030   \str_clear:N \sdefinitiontype
4031   \str_clear:N \sdefinitionid
4032   \str_clear:N \sdefinitionname
4033   \clist_clear:N \l__stex_statements_sdefinition_for_clist

```

```

4034 \tl_clear:N \sdefinitiontitle
4035 \keys_set:nn { stex / sdefinition }{ #1 }
4036 }
4037
4038 \NewDocumentEnvironment{sdefinition}{O{}}{
4039   \__stex_statements_sdefinition_args:n{ #1 }
4040   \stex_reactivate_macro:N \definiendum
4041   \stex_reactivate_macro:N \definame
4042   \stex_reactivate_macro:N \Definame
4043   \stex_if_smsmode:F{
4044     \seq_clear:N \l_tmpa_seq
4045     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4046       \str_if_eq:nnF{ ##1 }{ }{
4047         \stex_get_symbol:n { ##1 }
4048         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4049           \l_stex_get_symbol_uri_str
4050         }
4051       }
4052     }
4053     \exp_args:Nnnx
4054     \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {,}}
4055     \str_if_empty:NF \sdefinitiontype {
4056       \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{ }
4057     }
4058     \clist_set:Nn \l_tmpa_clist \sdefinitiontype
4059     \tl_clear:N \l_tmpa_tl
4060     \clist_map_inline:Nn \l_tmpa_clist {
4061       \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
4062         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
4063       }
4064     }
4065     \tl_if_empty:NTF \l_tmpa_tl {
4066       \__stex_statements_sdefinition_start:
4067     }{
4068       \l_tmpa_tl
4069     }
4070   }
4071   \stex_ref_new_doc_target:n \sdefinitionid
4072   \stex_smsmode_do:
4073 }{
4074   \str_if_empty:NF \sdefinitionname { \symdecl*{\sdefinitionname} }
4075   \stex_if_smsmode:F {
4076     \clist_set:Nn \l_tmpa_clist \sdefinitiontype
4077     \tl_clear:N \l_tmpa_tl
4078     \clist_map_inline:Nn \l_tmpa_clist {
4079       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
4080         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
4081       }
4082     }
4083     \tl_if_empty:NTF \l_tmpa_tl {
4084       \__stex_statements_sdefinition_end:
4085     }{
4086       \l_tmpa_tl
4087     }

```

```

4088     \end{stex_annotate_env}
4089   }
4090 }

```

\stexpatchdefinition

```

4091 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
4092   \par\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
4093     ~(\sdefinitiontitle)
4094   }~}
4095 }
4096 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\par\medskip}
4097
4098 \newcommand\stexpatchdefinition[3] [] {
4099   \str_set:Nx \l_tmpa_str{ #1 }
4100   \str_if_empty:NTF \l_tmpa_str {
4101     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
4102     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
4103   }{
4104     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
4105     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
4106   }
4107 }

```

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

\inlinedef inline:

```

4108 \keys_define:nn {stex / inlinedef }{
4109   type      .str_set_x:N = \sdefinitiontype,
4110   id        .str_set_x:N = \sdefinitionid,
4111   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4112   name      .str_set_x:N = \sdefinitionname
4113 }
4114 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
4115   \str_clear:N \sdefinitiontype
4116   \str_clear:N \sdefinitionid
4117   \str_clear:N \sdefinitionname
4118   \clist_clear:N \l__stex_statements_sdefinition_for_clist
4119   \keys_set:nn { stex / inlinedef }{ #1 }
4120 }
4121 \NewDocumentCommand \inlinedef { 0{} m } {
4122   \begingroup
4123   \__stex_statements_inlinedef_args:n{ #1 }
4124   \stex_ref_new_doc_target:n \sdefinitionid
4125   \stex_reactivate_macro:N \definiendum
4126   \stex_reactivate_macro:N \definame
4127   \stex_reactivate_macro:N \Definame
4128   \stex_if_smsmode:TF{
4129     \str_if_empty:NF \sdefinitionname { \symdecl*{\sdefinitionname} }
4130   }{
4131     \seq_clear:N \l_tmpa_seq
4132     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4133       \str_if_eq:nnF{ ##1 }{}{
4134         \stex_get_symbol:n { ##1 }
4135         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {

```

```

4136         \l_stex_get_symbol_uri_str
4137     }
4138 }
4139 }
4140 \exp_args:Nnx
4141 \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
4142     \str_if_empty:NF \sdefinitiontype {
4143         \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{ }
4144     }
4145     #2
4146     \str_if_empty:NF \sdefinitionname { \symdecl*{\sdefinitionname} }
4147 }
4148 }
4149 \endgroup
4150 \stex_smsmode_do:
4151 }

```

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

## 33.2 Assertions

**sassertion**

```

4152
4153 \keys_define:nn {stex / sassertion }{
4154     type      .str_set_x:N = \sassertiontype,
4155     id        .str_set_x:N = \sassertionid,
4156     title     .tl_set:N    = \sassertiontitle ,
4157     for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4158     name      .str_set_x:N = \sassertionname
4159 }
4160 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
4161     \str_clear:N \sassertiontype
4162     \str_clear:N \sassertionid
4163     \str_clear:N \sassertionname
4164     \clist_clear:N \l__stex_statements_sassertion_for_clist
4165     \tl_clear:N \sassertiontitle
4166     \keys_set:nn { stex / sassertion }{ #1 }
4167 }
4168
4169 %\tl_new:N \g__stex_statements_aftergroup_tl
4170
4171 \NewDocumentEnvironment{sassertion}{0{}}{
4172     \__stex_statements_sassertion_args:n{ #1 }
4173     \stex_if_smsmode:F {
4174         \seq_clear:N \l_tmpa_seq
4175         \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4176             \str_if_eq:nnF{ ##1 }{ }{
4177                 \stex_get_symbol:n { ##1 }
4178                 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4179                     \l_stex_get_symbol_uri_str
4180                 }
4181             }
4182         }

```

```

4183 \exp_args:Nnnx
4184 \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}
4185 \str_if_empty:NF \sassertiontype {
4186   \stex_annotate_invisible:nnn{type}{\sassertiontype}{\sassertiontype}{}
4187 }
4188 \clist_set:No \l_tmpa_clist \sassertiontype
4189 \tl_clear:N \l_tmpa_tl
4190 \clist_map_inline:Nn \l_tmpa_clist {
4191   \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
4192     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
4193   }
4194 }
4195 \tl_if_empty:NTF \l_tmpa_tl {
4196   \__stex_statements_sassertion_start:
4197 }{
4198   \l_tmpa_tl
4199 }
4200 }
4201 \stex_ref_new_doc_target:n \sassertionid
4202 \stex_smsmode_do:
4203 ){
4204   \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
4205   \stex_if_smsmode:F {
4206     \clist_set:No \l_tmpa_clist \sassertiontype
4207     \tl_clear:N \l_tmpa_tl
4208     \clist_map_inline:Nn \l_tmpa_clist {
4209       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
4210         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
4211       }
4212     }
4213     \tl_if_empty:NTF \l_tmpa_tl {
4214       \__stex_statements_sassertion_end:
4215     }{
4216       \l_tmpa_tl
4217     }
4218   }
4219 }
4220 }

```

\stexpatchassertion

```

4221
4222 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
4223   \par\noindent\titllemph{Assertion~\tl_if_empty:NF \sassertiontitle {
4224     (\sassertiontitle)
4225   }~}
4226 }
4227 \cs_new_protected:Nn \__stex_statements_sassertion_end: { \par\medskip}
4228
4229 \newcommand\stexpatchassertion[3] [] {
4230   \str_set:Nx \l_tmpa_str{ #1 }
4231   \str_if_empty:NTF \l_tmpa_str {
4232     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
4233     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
4234   }{

```

```

4235     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
4236     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
4237   }
4238 }

```

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

`\inlineass` inline:

```

4239 \keys_define:nn {stex / inlineass }{
4240   type      .str_set_x:N = \sassertiontype,
4241   id        .str_set_x:N = \sassertionid,
4242   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4243   name      .str_set_x:N = \sassertionname
4244 }
4245 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
4246   \str_clear:N \sassertiontype
4247   \str_clear:N \sassertionid
4248   \str_clear:N \sassertionname
4249   \clist_clear:N \l__stex_statements_sassertion_for_clist
4250   \keys_set:nn { stex / inlineass }{ #1 }
4251 }
4252 \NewDocumentCommand \inlineass { 0{} m } {
4253   \begingroup
4254   \__stex_statements_inlineass_args:n{ #1 }
4255   \stex_ref_new_doc_target:n \sassertionid
4256   \stex_if_smsmode:TF{
4257     \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
4258   }{
4259     \seq_clear:N \l_tmpa_seq
4260     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4261       \str_if_eq:nnF{ ##1 }{ }{
4262         \stex_get_symbol:n { ##1 }
4263         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4264           \l_stex_get_symbol_uri_str
4265         }
4266       }
4267     }
4268     \exp_args:Nnx
4269     \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{
4270       \str_if_empty:NF \sassertiontype {
4271         \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
4272       }
4273       #2
4274       \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
4275     }
4276   }
4277   \endgroup
4278   \stex_smsmode_do:
4279 }

```

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

## 33.3 Examples

sexample

```

4280
4281 \keys_define:nn {stex / sexample }{
4282   type      .str_set_x:N = \exampletype,
4283   id        .str_set_x:N = \sexampleid,
4284   title     .tl_set:N     = \sexampletitle,
4285   for       .clist_set:N  = \l__stex_statements_sexample_for_clist,
4286 }
4287 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
4288   \str_clear:N \sexampletype
4289   \str_clear:N \sexampleid
4290   \tl_clear:N \sexampletitle
4291   \clist_clear:N \l__stex_statements_sexample_for_clist
4292   \keys_set:nn { stex / sexample }{ #1 }
4293 }
4294
4295 \NewDocumentEnvironment{sexample}{0{}}{
4296   \__stex_statements_sexample_args:n{ #1 }
4297   \stex_if_smsmode:F {
4298     \seq_clear:N \l_tmpa_seq
4299     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4300       \str_if_eq:nnF{ ##1 }{}{
4301         \stex_get_symbol:n { ##1 }
4302         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4303           \l_stex_get_symbol_uri_str
4304         }
4305       }
4306     }
4307     \exp_args:Nnnx
4308     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
4309     \str_if_empty:NF \sexampletype {
4310       \stex_annotate_invisible:nnn{type}{\sexampletype}{}
4311     }
4312     \clist_set:Nn \l_tmpa_clist \sexampletype
4313     \tl_clear:N \l_tmpa_tl
4314     \clist_map_inline:Nn \l_tmpa_clist {
4315       \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
4316         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
4317       }
4318     }
4319     \tl_if_empty:NTF \l_tmpa_tl {
4320       \__stex_statements_sexample_start:
4321     }{
4322       \l_tmpa_tl
4323     }
4324   }
4325   \stex_ref_new_doc_target:n \sexampleid
4326   \stex_smsmode_do:
4327 }{
4328   \str_if_empty:NF \sexamplename { \symdecl*{\sexamplename} }
4329   \stex_if_smsmode:F {
4330     \clist_set:Nn \l_tmpa_clist \sexampletype

```



```

4331 \tl_clear:N \l_tmpa_tl
4332 \clist_map_inline:Nn \l_tmpa_clist {
4333   \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
4334     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
4335   }
4336 }
4337 \tl_if_empty:NTF \l_tmpa_tl {
4338   \__stex_statements_sexample_end:
4339 }{
4340   \l_tmpa_tl
4341 }
4342 \end{stex_annotate_env}
4343 }
4344 }

```

\stexpatchexample

```

4345
4346 \cs_new_protected:Nn \__stex_statements_sexample_start: {
4347   \par\noindent\titleemph{Example~\tl_if_empty:NF \sexampltitle {
4348     (\sexampltitle)
4349   }~}
4350 }
4351 \cs_new_protected:Nn \__stex_statements_sexample_end: {\par\medskip}
4352
4353 \newcommand\stexpatchexample[3] [] {
4354   \str_set:Nx \l_tmpa_str{ #1 }
4355   \str_if_empty:NTF \l_tmpa_str {
4356     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
4357     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
4358   }{
4359     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
4360     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
4361   }
4362 }

```

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

\inlineex inline:

```

4363 \keys_define:nn {stex / inlineex }{
4364   type      .str_set_x:N = \sexampltype,
4365   id        .str_set_x:N = \sexampleid,
4366   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
4367   name      .str_set_x:N = \sexamplname
4368 }
4369 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
4370   \str_clear:N \sexampltype
4371   \str_clear:N \sexampleid
4372   \str_clear:N \sexamplname
4373   \clist_clear:N \l__stex_statements_sexample_for_clist
4374   \keys_set:nn { stex / inlineex }{ #1 }
4375 }
4376 \NewDocumentCommand \inlineex { 0{} m } {
4377   \begingroup
4378   \__stex_statements_inlineex_args:n{ #1 }

```

```

4379 \stex_ref_new_doc_target:n \sexampleid
4380 \stex_if_smsmode:TF{
4381   \str_if_empty:NF \sexamplename { \symdecl*{\examplename} }
4382 }{
4383   \seq_clear:N \l_tmpa_seq
4384   \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4385     \str_if_eq:nnF{ ##1 }{ }{
4386       \stex_get_symbol:n { ##1 }
4387       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4388         \l_stex_get_symbol_uri_str
4389       }
4390     }
4391   }
4392   \exp_args:Nnx
4393   \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpa_seq {},}{ }{
4394     \str_if_empty:NF \sexamplename {
4395       \stex_annotate_invisible:nnn{type}{\sexamplename}{ }
4396     }
4397     #2
4398     \str_if_empty:NF \sexamplename { \symdecl*{\sexamplename} }
4399   }
4400 }
4401 \endgroup
4402 \stex_smsmode_do:
4403 }

```

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

## 33.4 Logical Paragraphs

sparagraph

```

4404 \keys_define:nn { stex / sparagraph } {
4405   id      .str_set:N = \sparagraphid ,
4406   title   .tl_set:N  = \l_stex_sparagraph_title_tl ,
4407   type    .str_set:N  = \sparagraphtype ,
4408   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
4409   from    .tl_set:N   = \sparagraphfrom ,
4410   to      .tl_set:N   = \sparagraphto ,
4411   start   .tl_set:N   = \l_stex_sparagraph_start_tl ,
4412   name    .str_set:N   = \sparagraphname
4413 }
4414
4415 \cs_new_protected:Nn \stex_sparagraph_args:n {
4416   \tl_clear:N \l_stex_sparagraph_title_tl
4417   \tl_clear:N \sparagraphfrom
4418   \tl_clear:N \sparagraphto
4419   \tl_clear:N \l_stex_sparagraph_start_tl
4420   \str_clear:N \sparagraphid
4421   \str_clear:N \sparagraphtype
4422   \clist_clear:N \l__stex_statements_sparagraph_for_clist
4423   \str_clear:N \sparagraphname
4424   \keys_set:nn { stex / sparagraph }{ #1 }
4425 }

```

```

4426 \newif\if@in@omtext\@in@omtextfalse
4427
4428 \NewDocumentEnvironment {sparagraph} { 0{} } {
4429   \stex_sparagraph_args:n { #1 }
4430   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4431     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
4432   }{
4433     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
4434   }
4435   \@in@omtexttrue
4436   \stex_if_smsmode:F {
4437     \seq_clear:N \l_tmpa_seq
4438     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
4439       \str_if_eq:nnF{ ##1 }{}{
4440         \stex_get_symbol:n { ##1 }
4441         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4442           \l_stex_get_symbol_uri_str
4443         }
4444       }
4445     }
4446     \exp_args:Nnnx
4447     \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
4448     \str_if_empty:NF \sparagraphtype {
4449       \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}
4450     }
4451     \str_if_empty:NF \sparagraphfrom {
4452       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
4453     }
4454     \str_if_empty:NF \sparagraphto {
4455       \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
4456     }
4457     \clist_set:Nn \l_tmpa_clist \sparagraphtype
4458     \tl_clear:N \l_tmpa_tl
4459     \clist_map_inline:Nn \sparagraphtype {
4460       \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
4461         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
4462       }
4463     }
4464     \tl_if_empty:NTF \l_tmpa_tl {
4465       \__stex_statements_sparagraph_start:
4466     }{
4467       \l_tmpa_tl
4468     }
4469   }
4470   \stex_ref_new_doc_target:n \sparagraphid
4471   \stex_smsmode_do:
4472   \ignorespacesandpars
4473 }{
4474   \stex_if_smsmode:F {
4475     \clist_set:Nn \l_tmpa_clist \sparagraphtype
4476     \tl_clear:N \l_tmpa_tl
4477     \clist_map_inline:Nn \l_tmpa_clist {
4478       \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
4479         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}

```

```

4480     }
4481   }
4482   \str_if_empty:NF \sparagraphname { \symdecl*{\sparagraphname} }
4483   \tl_if_empty:NTF \l_tmpa_tl {
4484     \__stex_statements_sparagraph_end:
4485   }{
4486     \l_tmpa_tl
4487   }
4488   \end{stex_annotate_env}
4489 }
4490 }

```

# \stexpatchparagraph

```

4491
4492 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
4493   \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4494     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
4495       \titleemph{\l_stex_sparagraph_title_tl}:~
4496     }
4497   }{
4498     \titleemph{\l_stex_sparagraph_start_tl}~
4499   }
4500 }
4501 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
4502
4503 \newcommand\stexpatchparagraph[3] [] {
4504   \str_set:Nx \l_tmpa_str{ #1 }
4505   \str_if_empty:NTF \l_tmpa_str {
4506     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
4507     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
4508   }{
4509     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2 }
4510     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
4511   }
4512 }
4513
4514 \keys_define:nn { stex / inlinepara } {
4515   id      .str_set_x:N = \sparagraphid ,
4516   type    .str_set_x:N = \sparagraphtype ,
4517   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
4518   from    .tl_set:N    = \sparagraphfrom ,
4519   to      .tl_set:N    = \sparagraphto ,
4520   name    .str_set:N   = \sparagraphname
4521 }
4522 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
4523   \tl_clear:N \sparagraphfrom
4524   \tl_clear:N \sparagraphto
4525   \str_clear:N \sparagraphid
4526   \str_clear:N \sparagraphtype
4527   \clist_clear:N \l__stex_statements_sparagraph_for_clist
4528   \str_clear:N \sparagraphname
4529   \keys_set:nn { stex / inlinepara }{ #1 }
4530 }
4531 \NewDocumentCommand \inlinepara { 0{} m } {

```

```

4532 \begingroup
4533 \__stex_statements_inlinepara_args:n{ #1 }
4534 \stex_ref_new_doc_target:n \sparagraphid
4535 \stex_if_smsmode:TF{
4536   \str_if_empty:NF \sparagraphname { \symdecl*\sparagraphname } }
4537 }{
4538   \seq_clear:N \l_tmpa_seq
4539   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
4540     \str_if_eq:nnF{ ##1 }{}{
4541       \stex_get_symbol:n { ##1 }
4542       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4543         \l_stex_get_symbol_uri_str
4544       }
4545     }
4546   }
4547   \exp_args:Nnx
4548   \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
4549     \str_if_empty:NF \sparagraphtype {
4550       \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}}
4551   }
4552   \str_if_empty:NF \sparagraphfrom {
4553     \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}}
4554   }
4555   \str_if_empty:NF \sparagraphto {
4556     \stex_annotate_invisible:nnn{to}{\sparagraphto}{}}
4557   }
4558   #2
4559   \str_if_empty:NF \sparagraphname { \symdecl*\sparagraphname } }
4560 }
4561 }
4562 \endgroup
4563 \stex_smsmode_do:
4564 }
4565

```

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

symboldoc

```

4566 \NewDocumentEnvironment{symboldoc}{ m }{
4567   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
4568   \seq_clear:N \l_tmpb_seq
4569   \seq_map_inline:Nn \l_tmpa_seq {
4570     \str_if_eq:nnF{ ##1 }{}{
4571       \stex_get_symbol:n { ##1 }
4572       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
4573         \l_stex_get_symbol_uri_str
4574       }
4575     }
4576   }
4577   \par
4578   \exp_args:Nnnx
4579   \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
4580 }{
4581   \end{stex_annotate_env}
4582 }

```

4583 `</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>

```
4584 <*package>
4585 <@@=stex_sproof>
4586
4587 %%%%%%%%%%%%%% sproof.dtx %%%%%%%%%%%%%%
4588
```

### 34.2 Proofs

We first define some keys for the proof environment.

```
4589 \keys_define:nn { stex / spf } {
4590   id          .str_set:N = \l__stex_sproof_spf_id_str,
4591   display     .tl_set:N  = \l__stex_sproof_spf_display_tl,
4592   for         .tl_set:N  = \l__stex_sproof_spf_for_tl ,
4593   from        .tl_set:N  = \l__stex_sproof_spf_from_tl ,
4594   proofend    .tl_set:N  = \l__stex_sproof_spf_proofend_tl,
4595   type        .tl_set:N  = \l__stex_sproof_spf_type_tl,
4596   title       .tl_set:N  = \l__stex_sproof_spf_title_tl,
4597   continues   .tl_set:N  = \l__stex_sproof_spf_continues_tl,
4598   functions   .tl_set:N  = \l__stex_sproof_spf_functions_tl,
4599   method      .tl_set:N  = \l__stex_sproof_spf_method_tl
4600 }
4601 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
4602   \str_clear:N \l__stex_sproof_spf_id_str
4603   \tl_clear:N \l__stex_sproof_spf_display_tl
4604   \tl_clear:N \l__stex_sproof_spf_for_tl
4605   \tl_clear:N \l__stex_sproof_spf_from_tl
4606   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
4607   \tl_clear:N \l__stex_sproof_spf_type_tl
4608   \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

```

4609 \tl_clear:N \l__stex_sproof_spf_continues_tl
4610 \tl_clear:N \l__stex_sproof_spf_functions_tl
4611 \tl_clear:N \l__stex_sproof_spf_method_tl
4612 \keys_set:nn { stex / spf }{ #1 }
4613 }

```

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

```

4614 \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.

```

4615 \newcount\count_ten
4616 \newenvironment{pst@with@label}[1]{
4617   \edef\pst@label{#1}
4618   \advance\count_ten by 1\relax
4619   \count_ten=1
4620 }{
4621   \advance\count_ten by -1\relax
4622 }

```

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

```

4623 \def\the@pst@label{
4624   \pst@make@label\pst@label{\number\count_ten}\l__stex_sproof_pstlabel_postfix_tl
4625 }

```

*(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.

```

4626 \keys_define:nn { stex / pstlabel }{
4627   prefix      .tl_set:N   = \l__stex_sproof_pstlabel_prefix_tl,
4628   delimiter   .tl_set:N   = \l__stex_sproof_pstlabel_delimiter_tl,
4629   postfix     .tl_set:N   = \l__stex_sproof_pstlabel_postfix_tl
4630 }
4631 \cs_new_protected:Nn \__stex_sproof_pstlabel_args:n {

```

---

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



```

4632 \tl_set:Nn \l__stex_sproof_pstlabel_prefix_tl {P}
4633 \tl_set:Nn \l__stex_sproof_pstlabel_delimiter_tl {.}
4634 \tl_clear:N \l__stex_sproof_pstlabel_postfix_tl
4635 }
4636 \__stex_sproof_pstlabel_args:n {}
4637 \newcommand\setpstlabelstyle[1]{
4638   \__stex_sproof_pstlabel_args:n {#1}
4639 }
4640 \newcommand\setpstlabelstyledefault{%
4641   \__stex_sproof_pstlabel_args:n{prefix=P,delimiter=.,postfix={}}
4642 }

```

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

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

```

4643 \ExplSyntaxOff
4644 \def\pst@make@label@long#1#2{\@for\@I:=#1\do{\expandafter\expandafter\expandafter\@I\csname
4645 \def\pst@make@label@angles#1#2{\ensuremath{\@for\@I:=#1\do{\rangle}}#2}
4646 \def\pst@make@label@short#1#2{#2}
4647 \def\pst@make@label@empty#1#2{}
4648 \ExplSyntaxOn
4649 \def\pstlabelstyle#1{%
4650   \def\pst@make@label{\use:c{pst@make@label@#1}}%
4651 }%
4652 \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.

```

4653 \def\next@pst@label{%
4654   \global\advance\count\count10 by 1%
4655 }%

```

(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

```

4656 \def\sproof@box{
4657   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
4658 }
4659 \def\spf@proofend{\sproof@box}
4660 \def\sproofend{
4661   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
4662     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
4663   }
4664 }
4665 \def\sProofEndSymbol#1{\def\sproof@box{#1}}

```

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

**spf@\*@kw**

```

4666 \def\spf@proofsketch@kw{Proof Sketch}
4667 \def\spf@proof@kw{Proof}
4668 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

4669 \AddToHook{begindocument}{
4670   \ltx@ifpackageloaded{babel}{
4671     \makeatletter
4672     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4673     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4674       \input{sproof-ngerman.ldf}
4675     }
4676     \clist_if_in:NnT \l_tmpa_clist {finnish}{
4677       \input{sproof-finnish.ldf}
4678     }
4679     \clist_if_in:NnT \l_tmpa_clist {french}{
4680       \input{sproof-french.ldf}
4681     }
4682     \clist_if_in:NnT \l_tmpa_clist {russian}{
4683       \input{sproof-russian.ldf}
4684     }
4685     \makeatother
4686   }{}
4687 }

```

`spfsketch`

```

4688 \newcommand\spfsketch[2][]{
4689   \__stex_sproof_spf_args:n{#1}
4690   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4691     \titleemph{
4692       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
4693         \spf@proofsketch@kw
4694       }{
4695         \l__stex_sproof_spf_type_tl
4696       }
4697     }:
4698   }
4699   {~#2}
4700   %\sref@label@id{this \ifx\spf@type\@empty\spf@proofsketch@kw\else\spf@type\fi}
4701   \sproofend
4702 }

```

(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>

```

4703 \newenvironment{spfeq}[2][]{
4704   \__stex_sproof_spf_args:n{#1}
4705   %\sref@target
4706   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4707     \titleemph{
4708       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
4709         \spf@proof@kw
4710       }{

```

<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

```

4711     \l__stex_sproof_spf_type_tl
4712   }
4713   }:
4714 }
4715 {-#2}
4716 \begin{displaymath}\begin{array}{rcll}
4717 }{
4718 \end{array}\end{displaymath}
4719 }

```

(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.

```

4720 \newenvironment{spf@proof}[2] []{
4721   \l__stex_sproof_spf_args:n{#1}
4722   %\sref@target
4723   \count_ten=10
4724   \par\noindent
4725   \tl_if_eq:NNTF \l__stex_sproof_spf_display_tl\spf@flow{
4726     \titleemph{
4727       \tl_if_empty:NNTF \l__stex_sproof_spf_type_tl {
4728         \spf@proof@kw
4729       }{
4730         \l__stex_sproof_spf_type_tl
4731       }
4732     }:
4733   }
4734   {-#2}
4735   %\sref@label@id{this \ifx\spf@type\empty\spf@proof@kw\else\spf@type\fi}
4736   \def\pst@label{}
4737   \newcount\pst@count% initialize the labeling mechanism
4738   \begin{description}\begin{pst@with@label}{\l__stex_sproof_pstlabel_prefix_tl}
4739   }{
4740     \end{pst@with@label}\end{description}
4741   }
4742   \newenvironment{sproof}[2] []{\begin{spf@proof}[#1]{#2}}{\sproofend\end{spf@proof}}
4743   \newenvironment{sProof}[2] []{\begin{spf@proof}[#1]{#2}}{\end{spf@proof}}

```

**\spfidea**

```

4744 \newcommand\spfidea[2] []{
4745   \l__stex_sproof_spf_args:n{#1}
4746   \titleemph{
4747     \tl_if_empty:NNTF \l__stex_sproof_spf_type_tl {Proof~Idea}{
4748       \l__stex_sproof_spf_type_tl
4749     }:
4750   }-#2
4751   \sproofend
4752 }

```

(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
4753 \newenvironment{spfstep}[1][]{
4754   \_stex_sproof_spf_args:n{#1}
4755   \@in@omtexttrue
4756   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4757     \item[\the@pst@label]
4758   }
4759   \tl_if_empty:NF \l__stex_sproof_spf_title_tl {
4760     {(\titleemph{\l__stex_sproof_spf_title_tl})\enspace}
4761   }
4762   %\sref@label@id{\pst@label}
4763   \ignorespacesandpars
4764 }{
4765   \next@pst@label\ignorespacesandpars
4766 }

```

**sproofcomment**

```

4767 \newenvironment{sproofcomment}[1][]{
4768   \_stex_sproof_spf_args:n{#1}
4769   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4770     \item[\the@pst@label]
4771   }
4772 }{
4773   \next@pst@label
4774 }

```

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.

```

4775 \newenvironment{subproof}[2][]{
4776   \_stex_sproof_spf_args:n{#1}
4777   \def\@test{#2}
4778   \ifx\@test\empty\else
4779     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4780       \item[\the@pst@label]
4781     }{#2}
4782   \fi
4783   \begin{pst@with@label}{\pst@label,\number\count_ten}
4784 }{
4785   \end{pst@with@label}\next@pst@label
4786 }

```

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

```

4787 \newenvironment{spfcases}[2][]{
4788   \def\@test{#1}
4789   \ifx\@test\empty
4790     \begin{subproof}[method=by-cases]{#2}

```

---

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

```

4791 \else
4792   \begin{subproof}[#1,method=by-cases]{#2}
4793 \fi
4794 }{
4795   \end{subproof}
4796 }

```

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

```

4797 \newenvironment{spfcase}[2] [] {
4798   \__stex_sproof_spf_args:n{#1}
4799   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4800     \item[\the@pst@label]
4801   }
4802   \def\@test{#2}
4803   \ifx\@test\@empty
4804   \else
4805     {\titleemph{#2}:~}
4806   \fi
4807   \begin{pst@with@label}{\pst@label,\number\count_ten}
4808 }{
4809   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4810     \sproofend
4811   }
4812   \end{pst@with@label}
4813   \next@pst@label
4814 }

```

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

```

4815 \newcommand\spfcasesketch[3] [] {
4816   \__stex_sproof_spf_args:n{#1}
4817   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4818     \item[\the@pst@label]
4819   }
4820   \def\@test{#2}
4821   \ifx\@test\@empty
4822   \else
4823     {\titleemph{#2}:~}
4824   \fi#3
4825   \next@pst@label
4826 }%

```

### 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.

```

4827 \keys_define:nn { stex / just }{
4828   id          .str_set:x:N = \l__stex_sproof_just_id_str,
4829   method      .tl_set:N    = \l__stex_sproof_just_method_tl,
4830   premises    .tl_set:N    = \l__stex_sproof_just_premises_tl,
4831   args        .tl_set:N    = \l__stex_sproof_just_args_tl
4832 }

```

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`

4833 `\newenvironment{justification}[1] [] {}{}`

`\premise`

4834 `\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.

4835 `\newcommand\justarg[2] [] {#2}`

4836 `\</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

```
4837 <*package>
4838
4839 %%%%%%%%%% others.dtx %%%%%%%%%%
4840
4841 <@@=stex_others>
    Warnings and error messages
4842 % None

\MSC Math subject classifier

4843 \NewDocumentCommand \MSC {m} {
4844 % TODO
4845 }

(End definition for \MSC. This function is documented on page 21.)
    Patching tikzinput, if loaded
4846 \@ifpackageloaded{tikzinput}{
4847 \RequirePackage{stex-tikzinput}
4848 }{}
4849 </package>
```

## Chapter 36

# STEX -Metatheory Implementation

```
4850 \*package>
4851 \@@=stex_modules>
4852
4853 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
4854
4855 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
4856 \begingroup
4857 \stex_module_setup:nn{
4858   ns=\c_stex_metatheory_ns_str,
4859   meta=NONE
4860 }{Metatheory}
4861 \stex_reactivate_macro:N \symdecl
4862 \stex_reactivate_macro:N \notation
4863 \stex_reactivate_macro:N \symdef
4864 \ExplSyntaxOff
4865 \csname stex_suppress_html:n\endcsname{
4866   % is-a (a:A, a \in A, a is an A, etc.)
4867   \symdecl[args=ai]{isa}
4868   \notation[typed]{isa}{#1 \comp{:} #2}{##1 \comp, ##2}
4869   \notation[in]{isa}{#1 \comp\in #2}{##1 \comp, ##2}
4870   \notation[pred]{isa}{#2\comp(#1 \comp)}{##1 \comp, ##2}
4871
4872   % bind (\forall, \Pi, \lambda etc.)
4873   \symdecl[args=Bi]{bind}
4874   \notation[forall]{bind}{\comp\forall #1.\;#2}{##1 \comp, ##2}
4875   \notation[\Pi]{bind}{\comp\prod_{#1}#2}{##1 \comp, ##2}
4876   \notation[deffun]{bind}{\comp( #1 \comp{ }\;\to\; ) #2}{##1 \comp, ##2}
4877
4878   % dummy variable
4879   \symdecl{dummyvar}
4880   \notation[underscore]{dummyvar}{\comp\_}
4881   \notation[dot]{dummyvar}{\comp\cdot}
4882   \notation[dash]{dummyvar}{\comp{\rm --}}
4883
4884   %fromto (function space, Hom-set, implication etc.)
```



```

4885 \symdecl[args=ai]{fromto}
4886 \notation[xarrow]{fromto}{#1 \comp\to #2}{##1 \comp\times ##2}
4887 \notation[arrow]{fromto}{#1 \comp\to #2}{##1 \comp\to ##2}
4888
4889 % mapto (lambda etc.)
4890 \%symdecl[args=Bi]{mapto}
4891 \%notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
4892 \%notation[lambda]{mapto}{\comp\lambda #1 \comp.; #2}{#1 \comp, #2}
4893 \%notation[lambdau]{mapto}{\comp\lambda_{#1} \comp.; #2}{#1 \comp, #2}
4894
4895 % function/operator application
4896 \symdecl[args=ia]{apply}
4897 \notation[prec=0;0x\infpres,parens]{apply}{#1 \comp( #2 \comp)}{##1 \comp, ##2}
4898 \notation[prec=0;0x\infpres,lambda]{apply}{#1 \; #2 }{##1 \; ; ##2}
4899
4900 % ‘type’ of all collections (sets, classes, types, kinds)
4901 \symdecl{collection}
4902 \notation[U]{collection}{\comp{\mathcal{U}}}
4903 \notation[set]{collection}{\comp{\textsf{Set}}}
4904
4905 % sequences
4906 \symdecl[args=1]{seqtype}
4907 \notation[kleene]{seqtype}{#1^{\comp\ast}}
4908
4909 \symdef[args=2,li,prec=nobrackets]{sequence-index}{#1}_{#2}
4910 \notation[ui,prec=nobrackets]{sequence-index}{#1}^{#2}
4911
4912 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{,\ellipses}}{##1\comp,##2}
4913 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{,\ellipses},#2}{##1\comp,##2}
4914 \symdef[args=aai,prec=nobrackets]{aseqfromtovia}{#1\comp{,\ellipses},#2\comp{,\ellipses},#3}
4915
4916 % letin (‘let’, local definitions, variable substitution)
4917 \symdecl[args=bii]{letin}
4918 \notation[let]{letin}{\comp{\rm let}}{#1\comp{=}#2;\comp{\rm in}}{#3}
4919 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
4920 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
4921
4922 % structures
4923 \symdecl*[args=1]{module-type}
4924 \notation{module-type}{\mathtt{MOD} #1}
4925 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
4926 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{##1 \comp, ##2}
4927
4928 }
4929 \ExplSyntaxOn
4930 \stex_add_to_current_module:n{
4931   \let\nappa\apply
4932   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
4933   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
4934   \def\livar{\csname sequence-index\endcsname[li]}
4935   \def\uivar{\csname sequence-index\endcsname[ui]}
4936   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
4937   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
4938   \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}

```

```
4939 }  
4940 \_stex_modules_end_module:  
4941 \endgroup  
4942 </package>
```

## Chapter 37

# Tikzinput Implementation

```
4943 <*package>
4944
4945 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
4946
4947 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
4948 \RequirePackage{l3keys2e}
4949
4950 \keys_define:nn { tikzinput } {
4951   image .bool_set:N = \c_tikzinput_image_bool,
4952   image .default:n = false ,
4953   unknown .code:n = {}
4954 }
4955
4956 \ProcessKeysOptions { tikzinput }
4957
4958 \bool_if:NTF \c_tikzinput_image_bool {
4959   \RequirePackage{graphicx}
4960
4961   \providecommand\usetikzlibrary[]{}
4962   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
4963 }{
4964   \RequirePackage{tikz}
4965   \RequirePackage{standalone}
4966
4967   \newcommand \tikzinput [2] [] {
4968     \setkeys{Gin}{#1}
4969     \ifx \Gin@ewidth \Gin@exclamation
4970       \ifx \Gin@eheight \Gin@exclamation
4971         \input { #2 }
4972       \else
4973         \resizebox{!}{ \Gin@eheight }{
4974           \input { #2 }
4975         }
4976       \fi
4977     \else
4978       \ifx \Gin@eheight \Gin@exclamation
4979         \resizebox{ \Gin@ewidth }{!}{
4980           \input { #2 }
```

```

4981     }
4982     \else
4983     \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
4984     \input { #2 }
4985     }
4986     \fi
4987 \fi
4988 }
4989 }
4990
4991 \newcommand \ctikzinput [2] [] {
4992   \begin{center}
4993     \tikzinput [1] {#2}
4994   \end{center}
4995 }
4996
4997 \@ifpackageloaded{stex}{
4998   \RequirePackage{stex-tikzinput}
4999 }{}
5000
5001 </package>
5002 <*stex>
5003 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
5004 \RequirePackage{stex}
5005 \RequirePackage{tikzinput}
5006
5007 \newcommand\mhtikzinput [2] [] {%
5008   \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
5009   \stex_in_repository:nn\Gin@mhrepos{
5010     \tikzinput [1]{\mhpath{##1}{#2}}
5011   }
5012 }
5013 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [1] {#2}\end{center}}
5014 </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.

```
5015 \*cls)
5016 \@@=document_structure)
5017 \ProvidesExplClass{document-structure}{2022/02/10}{3.0}{Modular Document Structure Class}
5018 \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`

```
5019 \keys_define:nn{ document-structure / pkg }{
5020   class      .str_set_x:N = \c_document_structure_class_str,
5021   minimal    .bool_set:N = \c_document_structure_minimal_bool,
5022   report     .code:n      = {
5023     \ClassWarning{document-structure}{the option 'report' is deprecated, use 'class=report',
5024     \str_set:Nn \c_document_structure_class_str {report}
5025   },
5026   book       .code:n      = {
5027     \ClassWarning{document-structure}{the option 'book' is deprecated, use 'class=book', ins
5028     \str_set:Nn \c_document_structure_class_str {book}
5029   },
5030   bookpart   .code:n      = {
5031     \ClassWarning{document-structure}{the option 'bookpart' is deprecated, use 'class=book,t
5032     \str_set:Nn \c_document_structure_class_str {book}
5033     \str_set:Nn \c_document_structure_topsect_str {chapter}
5034   },
```

```

5035 docopt      .str_set_x:N = \c_document_structure_docopt_str,
5036 unknown    .code:n      = {
5037   \PassOptionsToPackage{ \CurrentOption }{ document-structure }
5038 }
5039 }
5040 \ProcessKeysOptions{ document-structure / pkg }
5041 \str_if_empty:NT \c_document_structure_class_str {
5042   \str_set:Nn \c_document_structure_class_str {article}
5043 }
5044 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
5045   {\c_document_structure_class_str}
5046

```

### 38.3 Beefing up the document environment

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

```

5047 \RequirePackage{document-structure}
5048 \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>

```

5049 \keys_define:nn { document-structure / document }{
5050   id .str_set_x:N = \c_document_structure_document_id_str
5051 }
5052 \let\__document_structure_orig_document=\document
5053 \renewcommand{\document}[1][]{
5054   \keys_set:nn{ document-structure / document }{ #1 }
5055   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
5056   \__document_structure_orig_document
5057 }

```

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

```

5058 }
5059 \</cls>

```

### 38.4 Implementation: document-structure Package

```

5060 \<*package>
5061 \ProvidesExplPackage{document-structure}{2022/02/10}{3.0}{Modular Document Structure}
5062 \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

```

5063
5064 \keys_define:nn{ document-structure / pkg }{
5065   class      .str_set_x:N = \c_document_structure_class_str,
5066   topsect    .str_set_x:N = \c_document_structure_topsect_str,
5067   % showignores .bool_set:N = \c_document_structure_showignores_bool,
5068 }
5069 \ProcessKeysOptions{ document-structure / pkg }
5070 \str_if_empty:NT \c_document_structure_class_str {
5071   \str_set:Nn \c_document_structure_class_str {article}
5072 }
5073 \str_if_empty:NT \c_document_structure_topsect_str {
5074   \str_set:Nn \c_document_structure_topsect_str {section}
5075 }

```

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

```

5076 \RequirePackage{xspace}
5077 \RequirePackage{comment}
5078 \AddToHook{begindocument}{
5079   \ltx@ifpackageloaded{babel}{
5080     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5081     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5082       \makeatletter\input{omdoc-ngerman.ldf}\makeatother
5083     }
5084   }{}
5085 }

```

`\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.

```

5086 \int_new:N \l_document_structure_section_level_int
5087 \str_case:VnF \c_document_structure_topsect_str {
5088   {part}{
5089     \int_set:Nn \l_document_structure_section_level_int {0}
5090   }
5091   {chapter}{
5092     \int_set:Nn \l_document_structure_section_level_int {1}
5093   }
5094 }{
5095   \str_case:VnF \c_document_structure_class_str {
5096     {book}{
5097       \int_set:Nn \l_document_structure_section_level_int {0}
5098     }
5099     {report}{
5100       \int_set:Nn \l_document_structure_section_level_int {0}
5101     }
5102   }{
5103     \int_set:Nn \l_document_structure_section_level_int {2}
5104   }
5105 }

```

## 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

```
5106 \def\current@section@level{document}%
5107 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
5108 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
5109 \cs_new_protected:Npn \skipomgroup {
5110   \ifcase\l_document_structure_section_level_int
5111   \or\stepcounter{part}
5112   \or\stepcounter{chapter}
5113   \or\stepcounter{section}
5114   \or\stepcounter{subsection}
5115   \or\stepcounter{subsubsection}
5116   \or\stepcounter{paragraph}
5117   \or\stepcounter{subparagraph}
5118   \fi
5119 }
```

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

`blindomgroup`

```
5120 \newcommand\at@begin@blindomgroup[1]{%
5121 \newenvironment{blindomgroup}
5122 {
5123   \int_incr:N\l_document_structure_section_level_int
5124   \at@begin@blindomgroup\l_document_structure_section_level_int
5125 }{}}
```

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

```
5126 \newcommand\omgroup@nonum[2]{
5127   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
5128   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
5129 }
```

*(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.

```
5130 \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.



```

5131 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
5132   \@nameuse{#1}{#2}
5133 }{
5134   \cs_if_exist:NTF\rdfmata@sectioning{
5135     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
5136   }{
5137     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
5138   }
5139 }
5140 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
5141 }

```

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

omgroup

```

5142 \keys_define:nn { document-structure / omgroup }{
5143   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
5144   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
5145   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
5146   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
5147   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
5148   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
5149   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
5150   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
5151   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
5152   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
5153 }
5154 \cs_new_protected:Nn \__document_structure_omgroup_args:n {
5155   \str_clear:N \l__document_structure_omgroup_id_str
5156   \str_clear:N \l__document_structure_omgroup_date_str
5157   \clist_clear:N \l__document_structure_omgroup_creators_clist
5158   \clist_clear:N \l__document_structure_omgroup_contributors_clist
5159   \tl_clear:N \l__document_structure_omgroup_srccite_tl
5160   \tl_clear:N \l__document_structure_omgroup_type_tl
5161   \tl_clear:N \l__document_structure_omgroup_short_tl
5162   \tl_clear:N \l__document_structure_omgroup_display_tl
5163   \tl_clear:N \l__document_structure_omgroup_intro_tl
5164   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
5165   \keys_set:nn { document-structure / omgroup } { #1 }
5166 }

```

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.

```

5167 \newif\if@mainmatter\@mainmattertrue
5168 \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.

```

5169 \keys_define:nn { document-structure / sectioning }{
5170   name .str_set_x:N = \l__document_structure_sect_name_str ,
5171   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
5172   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
5173   num .bool_set:N = \l__document_structure_sect_num_bool ,
5174 }

```

```

5175 \cs_new_protected:Nn \__document_structure_sect_args:n {
5176   \str_clear:N \l__document_structure_sect_name_str
5177   \str_clear:N \l__document_structure_sect_ref_str
5178   \bool_set_false:N \l__document_structure_sect_clear_bool
5179   \bool_set_false:N \l__document_structure_sect_num_bool
5180   \keys_set:nn { document-structure / sectioning } { #1 }
5181 }
5182 \newcommand\omdoc@sectioning[3][]{
5183   \__document_structure_sect_args:n {#1}
5184   \let\omdoc@sect@name\l__document_structure_sect_name_str
5185   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
5186   \if@mainmatter% numbering not overridden by frontmatter, etc.
5187     \bool_if:NTF \l__document_structure_sect_num_bool {
5188       \omgroup@num{#2}{#3}
5189     }{
5190       \omgroup@nonum{#2}{#3}
5191     }
5192     \def\current@section@level{\omdoc@sect@name}
5193   \else
5194     \omgroup@nonum{#2}{#3}
5195   \fi
5196 }% 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.

```

5197 \newcommand\omgroup@redefine@addtocontents[1]{%
5198   %\edef\__document_structureimport{#1}%
5199   %\@for\@I:=\__document_structureimport\do{%
5200     %\edef\@path{\csname module@\@I @path\endcsname}%
5201     %\@ifundefined{tf@toc}\relax%
5202     %    {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
5203   %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
5204   %\def\addcontentsline##1##2##3{%
5205     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
5206   %\else% hyperref.sty not loaded
5207   %\def\addcontentsline##1##2##3{%
5208     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
5209   %\fi
5210 }% 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.

```

5211 \int_new:N \l_document_structure_omgroup_level_int
5212 \newenvironment{omgroup}[2][]{% keys, title
5213 {
5214   \__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.

```

5215 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
5216   \omgroup@redefine@addtocontents{
5217     %\@ifundefined{module@id}\used@modules%
5218     %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}

```

```

5219     }
5220 }

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

5221 \int_incr:N \l_document_structure_omgroup_level_int
5222 \int_incr:N \l_document_structure_section_level_int
5223 \ifcase\l_document_structure_section_level_int
5224   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
5225   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
5226   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
5227   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
5228   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
5229   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
5230   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
5231 \fi
5232 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
5233 \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
5234 }% for customization
5235 {}

```

and finally, we localize the sections

```

5236 \newcommand\omdoc@part@kw{Part}
5237 \newcommand\omdoc@chapter@kw{Chapter}
5238 \newcommand\omdoc@section@kw{Section}
5239 \newcommand\omdoc@subsection@kw{Subsection}
5240 \newcommand\omdoc@subsubsection@kw{Subsubsection}
5241 \newcommand\omdoc@paragraph@kw{paragraph}
5242 \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`

```

5243 \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).

```

5244 \cs_if_exist:NTF\frontmatter{
5245   \let\__document_structure_orig_frontmatter\frontmatter
5246   \let\frontmatter\relax
5247 }{
5248   \tl_set:Nn\__document_structure_orig_frontmatter{
5249     \clearpage
5250     \@mainmatterfalse
5251     \pagenumbering{roman}
5252   }
5253 }

```

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

```

5264 \newenvironment{frontmatter}{
5265   \__document_structure_orig_frontmatter
5266 }{
5267   \cs_if_exist:NTF\mainmatter{
5268     \mainmatter
5269   }{
5270     \clearpage
5271     \@mainmattertrue
5272     \pagenumbering{arabic}
5273   }
5274 }

```

```

5275 \newenvironment{backmatter}{
5276   \__document_structure_orig_backmatter
5277 }{
5278   \cs_if_exist:NTF\mainmatter{
5279     \mainmatter
5280   }{
5281     \clearpage
5282     \@mainmattertrue
5283     \pagenumbering{arabic}
5284   }
5285 }

```

5286 \@mainmattertrue\pagenumbering{arabic}

```

5287 \def \c__document_structure_document_str{document}
5288 \newcommand\afterprematurestop{}
5289 \def\prematurestop@endomgroup{
5290   \unless\ifx\@currentvir\c__document_structure_document_str
5291     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
5292       \expandafter\prematurestop@endomgroup
5293     \fi
5294   }

```

```

5295 \providecommand\prematurestop{
5296   \message{Stopping~sTeX~processing~prematurely}
5297   \prematurestop@endomgroup
5298   \afterprematurestop
5299   \end{document}
5300 }

```

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

## 38.8 Global Variables

**\setSGvar** set a global variable

```

5301 \RequirePackage{etoolbox}
5302 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

**\useSGvar** use a global variable

```

5303 \newrobustcmd\useSGvar[1]{%
5304   \@ifundefined{sTeX@Gvar@#1}
5305   {\PackageError{document-structure}
5306     {The sTeX Global variable #1 is undefined}
5307     {set it with \protect\setSGvar}}
5308   \@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.

```

5309 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
5310   \@ifundefined{sTeX@Gvar@#1}
5311   {\PackageError{document-structure}
5312     {The sTeX Global variable #1 is undefined}
5313     {set it with \protect\setSGvar}}
5314   {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}

```

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

## Chapter 39

# NotesSlides – Implementation

### 39.1 Class and Package Options

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

```
5315 \*cls)
5316 \@@=notesslides)
5317 \ProvidesExplClass{notesslides}{2022/02/10}{3.0}{notesslides Class}
5318 \RequirePackage{l3keys2e,expl-keystr-compat}
5319
5320 \keys_define:nn{notesslides / cls}{
5321   class .code:n = {
5322     \PassOptionsToClass{\CurrentOption}{omdoc}
5323     \str_if_eq:nnT{#1}{book}{
5324       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5325     }
5326     \str_if_eq:nnT{#1}{report}{
5327       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5328     }
5329   },
5330   notes .bool_set:N = \c__notesslides_notes_bool ,
5331   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
5332   unknown .code:n = {
5333     \PassOptionsToClass{\CurrentOption}{omdoc}
5334     \PassOptionsToClass{\CurrentOption}{beamer}
5335     \PassOptionsToPackage{\CurrentOption}{notesslides}
5336   }
5337 }
5338 \ProcessKeysOptions{ notesslides / cls }
5339 \bool_if:NTF \c__notesslides_notes_bool {
5340   \PassOptionsToPackage{notes=true}{notesslides}
5341 }{
5342   \PassOptionsToPackage{notes=false}{notesslides}
5343 }
5344 \</cls)
```

now we do the same for the notesslides package.

```

5345 <*package>
5346 \ProvidesExplPackage{notesslides}{2022/02/10}{3.0}{notesslides Package}
5347 \RequirePackage{13keys2e,expl-keystr-compat}
5348
5349 \keys_define:nn{notesslides / pkg}{
5350   topsect      .str_set_x:N = \c__notesslides_topsect_str,
5351   defaulttopsect .str_set_x:N = \c__notesslides_defaulttopsec_str,
5352   notes        .bool_set:N = \c__notesslides_notes_bool ,
5353   slides       .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
5354   sectocframes .bool_set:N = \c__notesslides_sectocframes_bool ,
5355   frameimages  .bool_set:N = \c__notesslides_frameimages_bool ,
5356   fiboxed      .bool_set:N = \c__notesslides_fiboxed_bool ,
5357   nopproblems  .bool_set:N = \c__notesslides_nopproblems_bool,
5358   unknown      .code:n      = {
5359     \PassOptionsToClass{\CurrentOption}{stex}
5360     \PassOptionsToClass{\CurrentOption}{tikzinput}
5361   }
5362 }
5363 \ProcessKeysOptions{ notesslides / pkg }
5364 \newif\ifnotes
5365 \bool_if:NTF \c__notesslides_notes_bool {
5366   \notesttrue
5367 }{
5368   \notesfalse
5369 }
5370

```

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

```

5371 \str_if_empty:NTF \c__notesslides_topsect_str {
5372   \str_set_eq:NN \__notesslides_topsect \c__notesslides_defaulttopsec_str
5373 }{
5374   \str_set_eq:NN \__notesslides_topsect \c__notesslides_topsect_str
5375 }
5376 </package>

```

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

```

5377 <*cls>
5378 \bool_if:NTF \c__notesslides_notes_bool {
5379   \LoadClass{document-structure}
5380 }{
5381   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
5382   \newcounter{Item}
5383   \newcounter{paragraph}
5384   \newcounter{subparagraph}
5385   \newcounter{Hfootnote}
5386   \RequirePackage{document-structure}
5387 }

```

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

```

5388 \RequirePackage{notesslides}
5389 </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.

```

5390 \*package>
5391 \bool_if:NT \c__notesslides_notes_bool {
5392   \RequirePackage{a4wide}
5393   \RequirePackage{marginnote}
5394   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
5395   \RequirePackage{mdframed}
5396   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
5397   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
5398 }
5399 \RequirePackage{stex-tikzinput}
5400 \RequirePackage{etoolbox}
5401 \RequirePackage{amssymb}
5402 \RequirePackage{amsmath}
5403 \RequirePackage{comment}
5404 \RequirePackage{textcomp}
5405 \RequirePackage{url}
5406 \RequirePackage{graphicx}
5407 \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>

```

5408 \bool_if:NT \c__notesslides_notes_bool {
5409   \renewcommand\usetheme[2][\]{\usepackage[#1]{beamernotestheme#2}}
5410 }

```

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

```

5411 \newcounter{slide}
5412 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
5413 \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.

```

5414 \bool_if:NTF \c__notesslides_notes_bool {
5415   \renewenvironment{note}{\ignorespaces}{\ignorespaces}{}
5416 }{
5417   \excludcomment{note}
5418 }

```

---

<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.

```
5419 \bool_if:NT \c__notesslides_notes_bool {
5420   \newlength{\slideframewidth}
5421   \setlength{\slideframewidth}{1.5pt}
```

**frame** We first define the keys.

```
5422 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
5423   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
5424     \bool_set_true:N #1
5425   }{
5426     \bool_set_false:N #1
5427   }
5428 }
5429 \keys_define:nn{notesslides / frame}{
5430   label .str_set_x:N = \l__notesslides_frame_label_str,
5431   allowframebreaks .code:n = {
5432     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
5433   },
5434   allowdisplaybreaks .code:n = {
5435     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
5436   },
5437   fragile .code:n = {
5438     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
5439   },
5440   shrink .code:n = {
5441     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
5442   },
5443   squeeze .code:n = {
5444     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
5445   },
5446   t .code:n = {
5447     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
5448   },
5449 }
5450 \cs_new_protected:Nn \__notesslides_frame_args:n {
5451   \str_clear:N \l__notesslides_frame_label_str
5452   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
5453   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
5454   \bool_set_true:N \l__notesslides_frame_fragile_bool
5455   \bool_set_true:N \l__notesslides_frame_shrink_bool
5456   \bool_set_true:N \l__notesslides_frame_squeeze_bool
5457   \bool_set_true:N \l__notesslides_frame_t_bool
5458   \keys_set:nn { notesslides / frame }{ #1 }
5459 }
```

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

```
5460 \renewenvironment{frame}[1][]{
5461   \__notesslides_frame_args:n{#1}
5462   \sffamily
5463   \stepcounter{slide}
5464   \def\@currentlabel{\theslide}
5465   \str_if_empty:NF \l__notesslides_frame_label_str {
5466     \label{\l__notesslides_frame_label_str}
```

5467 }

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

```

5468 \def\itemize@level{outer}
5469 \def\itemize@outer{outer}
5470 \def\itemize@inner{inner}
5471 \renewcommand\newpage{\addtocounter{framenum}{1}}
5472 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
5473 \renewenvironment{itemize}{
5474   \ifx\itemize@level\itemize@outer
5475     \def\itemize@label{\$ \rhd$}
5476   \fi
5477   \ifx\itemize@level\itemize@inner
5478     \def\itemize@label{\$ \scriptstyle \rhd$}
5479   \fi
5480   \begin{list}
5481     {\itemize@label}
5482     {\setlength{\labelsep}{.3em}
5483      \setlength{\labelwidth}{.5em}
5484      \setlength{\leftmargin}{1.5em}
5485     }
5486   \edef\itemize@level{\itemize@inner}
5487 }{
5488   \end{list}
5489 }

```

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

```

5490 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth]
5491 }{
5492   \medskip\miko@slidelabel\end{mdframed}
5493 }

```

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

`\frametitle`

```

5494 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
5495 }

```

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

EdN:21

`\pause` 21

```

5496 \bool_if:NT \c__notesslides_notes_bool {
5497   \newcommand\pause{}
5498 }

```

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

`nparagraph`

```

5499 \bool_if:NTF \c__notesslides_notes_bool {
5500   \newenvironment{nparagraph}[1] [] {\begin{sparagraph}[#1]}{\end{sparagraph}}
5501 }{
5502   \excludecomment{nparagraph}
5503 }

```

---

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

nomgroup

```
5504 \bool_if:NTF \c__notesslides_notes_bool {
5505   \newenvironment{nomgroup}[2] [] {\begin{omgroup}[#1]{#2}}{\end{omgroup}}
5506 }{
5507   \excludecomment{nomgroup}
5508 }
```

ndefinition

```
5509 \bool_if:NTF \c__notesslides_notes_bool {
5510   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}{\end{sdefinition}}
5511 }{
5512   \excludecomment{ndefinition}
5513 }
```

nassertion

```
5514 \bool_if:NTF \c__notesslides_notes_bool {
5515   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}{\end{sassertion}}
5516 }{
5517   \excludecomment{nassertion}
5518 }
```

nsproof

```
5519 \bool_if:NTF \c__notesslides_notes_bool {
5520   \newenvironment{nproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
5521 }{
5522   \excludecomment{nproof}
5523 }
```

nexample

```
5524 \bool_if:NTF \c__notesslides_notes_bool {
5525   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}{\end{sexample}}
5526 }{
5527   \excludecomment{nexample}
5528 }
```

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

```
5529 \def\inputref@preskip{\smallskip}
5530 \def\inputref@postskip{\medskip}
```

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

\inputref\*

```
5531 \let\orig@inputref\inputref
5532 \def\inputref{@ifstar\ninputref\orig@inputref}
5533 \newcommand\ninputref[2] [] {
5534   \bool_if:NT \c__notesslides_notes_bool {
5535     \orig@inputref[#1]{#2}
5536   }
5537 }
```

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

## 39.3 Header and Footer Lines

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

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

```

5538 \newlength{\slidelogoheight}
5539
5540 \bool_if:NTF \c__notesslides_notes_bool {
5541   \setlength{\slidelogoheight}{.4cm}
5542 }{
5543   \setlength{\slidelogoheight}{1cm}
5544 }
5545 \newsavebox{\slidelogo}
5546 \sbox{\slidelogo}{\TeX}
5547 \newrobustcmd{\setslidelogo}[1]{
5548   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
5549 }
```

(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.

```

5550 \def\source{Michael Kohlhase}% customize locally
5551 \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.

```

5552 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
5553 \newsavebox{\cclogo}
5554 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
5555 \newif\ifcchref\cchreffalse
5556 \AtBeginDocument{
5557   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
5558 }
5559 \def\licensing{
5560   \ifcchref
5561     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
5562   \else
5563     {\usebox{\cclogo}}
5564   \fi
5565 }
5566 \newrobustcmd{\setlicensing}[2][]{
5567   \def\@url{#1}
5568   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
5569   \ifx\@url\@empty
5570     \def\licensing{{\usebox{\cclogo}}}
5571   \else
5572     \def\licensing{
```

```

5573     \ifcchref
5574     \href{#1}{\usebox{\cclogo}}
5575     \else
5576     {\usebox{\cclogo}}
5577     \fi
5578   }
5579   \fi
5580 }

```

(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>

```

5581 \newrobustcmd\miko@slidelabel{
5582   \vbox to \slidelogoheight{
5583     \vss\hbox to \slidewidth
5584     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
5585   }
5586 }

```

(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.

```

5587 \def\Gin@mhrepos{}
5588 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
5589 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
5590 \newrobustcmd\frameimage[2][{}{
5591   \stepcounter{slide}
5592   \bool_if:NT \c__notesslides_frameimages_bool {
5593     \def\Gin@ewidth{}\setkeys{Gin}{#1}
5594     \bool_if:NF \c__notesslides_notes_bool { \vfill }
5595     \begin{center}
5596       \bool_if:NTF \c__notesslides_fiboxed_bool {
5597         \fbox{
5598           \ifx\Gin@ewidth\@empty
5599             \ifx\Gin@mhrepos\@empty
5600               \mhgraphics[width=\slidewidth,#1]{#2}
5601             \else
5602               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5603             \fi
5604           \else% Gin@ewidth empty
5605             \ifx\Gin@mhrepos\@empty
5606               \mhgraphics[#1]{#2}
5607             \else
5608               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5609             \fi
5610           \fi% Gin@ewidth empty
5611         }
5612       }{
5613         \ifx\Gin@ewidth\@empty

```

---

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

```

5614         \ifx\Gin@mhrepos\@empty
5615             \mhgraphics[width=\slidewidth,#1]{#2}
5616         \else
5617             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5618         \fi
5619         \ifx\Gin@mhrepos\@empty
5620             \mhgraphics[#1]{#2}
5621         \else
5622             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5623         \fi
5624     \fi% Gin@ewidth empty
5625 }
5626 \end{center}
5627 \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
5628 \bool_if:NF \c__notesslides_notes_bool { \vfill }
5629 }
5630 } % 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.

```

5631 \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

```

5632 \AddToHook{begindocument}{
5633     \definecolor{green}{rgb}{0,.5,0}
5634     \definecolor{purple}{cmyk}{.3,1,0,.17}
5635 }

```

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`.

```

5636 % \def\STpresent#1{\textcolor{blue}{#1}}
5637 \def\defemph#1{\textcolor{magenta}{#1}}
5638 \def\symrefemph#1{\textcolor{cyan}{#1}}
5639 \def\compemph#1{\textcolor{blue}{#1}}
5640 \def\titleemph#1{\textcolor{blue}{#1}}
5641 \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.

```

5642 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
5643 \def\smalltextwarning{
5644     \pgfuseimage{miko@small@dbend}
5645     \xspace
5646 }
5647 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}

```

```

5648 \newrobustcmd\textwarning{
5649   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
5650   \xspace
5651 }
5652 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}
5653 \newrobustcmd\bigtextwarning{
5654   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
5655   \xspace
5656 }

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

5657 \newrobustcmd\putgraphicsat[3]{
5658   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
5659 }
5660 \newrobustcmd\putat[2]{
5661   \begin{picture}(0,0)\put(#1){#2}\end{picture}
5662 }

```

## 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`.

```

5663 \bool_if:NT \c__notesslides_sectocframes_bool {
5664   \str_if_eq:VnTF \__notesslidesstopsect{part}{
5665     \newcounter{chapter}\counterwithin*{section}{chapter}
5666   }{
5667     \str_if_eq:VnT\__notesslidesstopsect{chapter}{
5668       \newcounter{chapter}\counterwithin*{section}{chapter}
5669     }
5670   }
5671 }

```

`\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

5672 \def\part@prefix{}
5673 \@ifpackageloaded{document-structure}{}{
5674   \str_case:VnF \__notesslidesstopsect {
5675     {part}{
5676       \int_set:Nn \l_document_structure_section_level_int {0}
5677       \def\thesection{\arabic{chapter}.\arabic{section}}
5678       \def\part@prefix{\arabic{chapter}.}
5679     }
5680     {chapter}{
5681       \int_set:Nn \l_document_structure_section_level_int {1}
5682       \def\thesection{\arabic{chapter}.\arabic{section}}
5683       \def\part@prefix{\arabic{chapter}.}
5684     }
5685   }{
5686     \int_set:Nn \l_document_structure_section_level_int {2}
5687     \def\part@prefix{}

```

```

5688 }
5689 }
5690
5691 \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

```

5692 \renewenvironment{omgroup}[2][]{
5693   \__document_structure_omgroup_args:n { #1 }
5694   \int_incr:N \l_document_structure_omgroup_level_int
5695   \int_incr:N \l_document_structure_section_level_int
5696   \bool_if:NT \c__notesslides_sectocframes_bool {
5697     \stepcounter{slide}
5698     \begin{frame}[noframenumbering]
5699     \vfill\Large\centering
5700     \red{
5701       \ifcase\l_document_structure_section_level_int\or
5702         \stepcounter{part}
5703         \def\__notesslideslabel{\omdoc@part@kw~\Roman{part}}
5704         \def\currentsectionlevel{\omdoc@part@kw}
5705       \or
5706         \stepcounter{chapter}
5707         \def\__notesslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
5708         \def\currentsectionlevel{\omdoc@chapter@kw}
5709       \or
5710         \stepcounter{section}
5711         \def\__notesslideslabel{\part@prefix\arabic{section}}
5712         \def\currentsectionlevel{\omdoc@section@kw}
5713       \or
5714         \stepcounter{subsection}
5715         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
5716         \def\currentsectionlevel{\omdoc@subsection@kw}
5717       \or
5718         \stepcounter{subsubsection}
5719         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
5720         \def\currentsectionlevel{\omdoc@subsubsection@kw}
5721       \or
5722         \stepcounter{paragraph}
5723         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{paragraph}}
5724         \def\currentsectionlevel{\omdoc@paragraph@kw}
5725       \else
5726         \def\__notesslideslabel{}
5727         \def\currentsectionlevel{\omdoc@paragraph@kw}
5728       \fi% end ifcase
5729       \__notesslideslabel%\sref@label@id\__notesslideslabel
5730       \quad #2%
5731     }%
5732     \vfill%
5733     \end{frame}%
5734   }
5735   \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str%

```



```

5736 }{}
5737 }

```

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

```

5738 \def\inserttheorembodyfont{\normalfont}
5739 %\bool_if:NF \c__notesslides_notes_bool {
5740 % \defbeamertemplate{theorem begin}{miko}
5741 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
5742 % \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
5743 % \inserttheorempunctuation\inserttheorembodyfont\hspace}
5744 % \defbeamertemplate{theorem end}{miko}{}}

```

and we set it as the default one.

```

5745 % \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.

```

5746 % \expandafter\def\csname Parent2\endcsname{}
5747 %}
5748
5749 \AddToHook{begindocument}{% this does not work for some reasons
5750 \setbeamertemplate{theorems}[ams style]
5751 }
5752 \bool_if:NT \c__notesslides_notes_bool {
5753 \renewenvironment{columns}[1][{}]{%
5754 \par\noindent%
5755 \begin{minipage}%
5756 \slidewidth\centering\leavevmode%
5757 }{}%
5758 \end{minipage}\par\noindent%
5759 }%
5760 \newsavebox\columnbox%
5761 \renewenvironment<>{column}[2][{}]{%
5762 \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
5763 }{}%
5764 \end{minipage}\end{lrbox}\usebox\columnbox%
5765 }%
5766 }
5767 \bool_if:NTF \c__notesslides_noproblems_bool {
5768 \newenvironment{problems}{}{}
5769 }{
5770 \excludecomment{problems}
5771 }

```

## 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.

```

5772 \gdef\printexcursions{}
5773 \newcommand\excursionref[2]{% label, text
5774 \bool_if:NT \c__notesslides_notes_bool {

```

```

5775 \begin{sparagraph}[title=Excursion]
5776 #2 \sref[fallback=the appendix]{#1}.
5777 \end{sparagraph}
5778 }
5779 }
5780 \newcommand\activate@excursion[2][]{
5781 \gappto\printexcursions{\inputref{#1}{#2}}
5782 }
5783 \newcommand\excursion[4][]{% repos, label, path, text
5784 \bool_if:NT \c__notesslides_notes_bool {
5785 \activate@excursion{#1}{#3}\excursionref{#2}{#4}
5786 }
5787 }

```

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

\excursiongroup

```

5788 \keys_define:nn{notesslides / excursiongroup }{
5789 id .str_set_x:N = \l__notesslides_excursion_id_str,
5790 intro .tl_set:N = \l__notesslides_excursion_intro_tl,
5791 mhrepos .str_set_x:N = \l__notesslides_excursion_mhrepos_str
5792 }
5793 \cs_new_protected:Nn \__notesslides_excursion_args:n {
5794 \tl_clear:N \l__notesslides_excursion_intro_tl
5795 \str_clear:N \l__notesslides_excursion_id_str
5796 \str_clear:N \l__notesslides_excursion_mhrepos_str
5797 \keys_set:nn {notesslides / excursiongroup }{ #1 }
5798 }
5799 \newcommand\excursiongroup[1][]{
5800 \__notesslides_excursion_args:n{ #1 }
5801 \ifdefempty\printexcursions{}% only if there are excursions
5802 {\begin{note}
5803 \begin{omgroup}[#1]{Excursions}%
5804 \ifdefempty\l__notesslides_excursion_intro_tl}{
5805 \inputref[\l__notesslides_excursion_mhrepos_str]{
5806 \l__notesslides_excursion_intro_tl
5807 }
5808 }
5809 \printexcursions%
5810 \end{omgroup}
5811 \end{note}}
5812 }
5813 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
5814 \end{package}

```

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

## Chapter 40

# The Implementation

### 40.1 Package Options

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

```
5815 <*package>
5816 <@@=problems>
5817 \ProvidesExplPackage{problem}{2019/03/20}{1.3}{Semantic Markup for Problems}
5818 \RequirePackage{l3keys2e,expl-keystr-compatible}
5819
5820 \keys_define:nn { problem / pkg }{
5821   notes      .default:n    = { true },
5822   notes      .bool_set:N   = \c__problems_notes_bool,
5823   gnotes     .default:n    = { true },
5824   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
5825   hints      .default:n    = { true },
5826   hints      .bool_set:N   = \c__problems_hints_bool,
5827   solutions  .default:n    = { true },
5828   solutions  .bool_set:N   = \c__problems_solutions_bool,
5829   pts        .default:n    = { true },
5830   pts        .bool_set:N   = \c__problems_pts_bool,
5831   min        .default:n    = { true },
5832   min        .bool_set:N   = \c__problems_min_bool,
5833   boxed      .default:n    = { true },
5834   boxed      .bool_set:N   = \c__problems_boxed_bool,
5835   unknown    .code:n       = {}
5836 }
5837 \newif\ifsolutions
5838
5839 \ProcessKeysOptions{ problem / pkg }
5840 \bool_if:NTF \c__problems_solutions_bool {
5841   \solutionstrue
5842 }{
5843   \solutionsfalse
5844 }
```

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

```
5845 \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.

```
5846 \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.

```
5847 \def\prob@problem@kw{Problem}
5848 \def\prob@solution@kw{Solution}
5849 \def\prob@hint@kw{Hint}
5850 \def\prob@note@kw{Note}
5851 \def\prob@gnote@kw{Grading}
5852 \def\prob@pt@kw{pt}
5853 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
5854 \AddToHook{begindocument}{
5855   \ltx@ifpackageloaded{babel}{
5856     \makeatletter
5857     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5858     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5859       \input{problem-ngerman.ldf}
5860     }
5861     \clist_if_in:NnT \l_tmpa_clist {finnish}{
5862       \input{problem-finnish.ldf}
5863     }
5864     \clist_if_in:NnT \l_tmpa_clist {french}{
5865       \input{problem-french.ldf}
5866     }
5867     \clist_if_in:NnT \l_tmpa_clist {russian}{
5868       \input{problem-russian.ldf}
5869     }
5870     \makeatother
5871   }{ }
5872 }
```

## 40.2 Problems and Solutions

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

```
5873 \keys_define:nn{ problem / problem }{
5874   id      .str_set_x:N = \l__problems_prob_id_str,
5875   pts     .tl_set:N    = \l__problems_prob_pts_tl,
5876   min     .tl_set:N    = \l__problems_prob_min_tl,
5877   title   .tl_set:N    = \l__problems_prob_title_tl,
5878   type    .tl_set:N    = \l__problems_prob_type_tl,
5879   refnum  .int_set:N    = \l__problems_prob_refnum_int
5880 }
5881 \cs_new_protected:Nn \__problems_prob_args:n {
```

```

5882 \str_clear:N \l__problems_prob_id_str
5883 \tl_clear:N \l__problems_prob_pts_tl
5884 \tl_clear:N \l__problems_prob_min_tl
5885 \tl_clear:N \l__problems_prob_title_tl
5886 \tl_clear:N \l__problems_prob_type_tl
5887 \int_zero_new:N \l__problems_prob_refnum_int
5888 \keys_set:nn { problem / problem }{ #1 }
5889 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
5890   \let\l__problems_prob_refnum_int\undefined
5891 }
5892 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

5893 \newcounter{problem}
5894 \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.

```

5895 \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

```

5896 \newcommand\prob@number{
5897   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
5898     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
5899   }{
5900     \int_if_exist:NTF \l__problems_prob_refnum_int {
5901       \prob@label{\int_use:N \l__problems_prob_refnum_int }
5902     }{
5903       \prob@label\theproblem
5904     }
5905   }
5906 }

```

(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.

```

5907 \newcommand\prob@title[3]{%
5908   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
5909     #2 \l__problems_inclprob_title_tl #3
5910   }{
5911     \tl_if_exist:NTF \l__problems_prob_title_tl {
5912       #2 \l__problems_prob_title_tl #3
5913     }{
5914       #1
5915     }
5916   }
5917 }

```

(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.

```

5918 \def\prob@heading{
5919   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
5920   %\sref@label{id}\prob@problem@kw~\prob@number}{~}
5921 }

```

(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`

```

5922 \newenvironment{sproblem}[1][{}]{
5923   \__problems_prob_args:n{#1}%\sref@target%
5924   \@in@omtexttrue% we are in a statement (for inline definitions)
5925   \stepcounter{problem}\record@problem
5926   \def\current@section@level{\prob@problem@kw}
5927   \tl_if_exist:NTF \l__problems_inclprob_type_tl {
5928     \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
5929   }{
5930     \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
5931   }
5932   \str_if_exist:NTF \l__problems_inclprob_id_str {
5933     \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
5934   }{
5935     \str_set_eq:NN \sproblemid \l__problems_prob_id_str
5936   }
5937
5938
5939   \clist_set:No \l_tmpa_clist \sproblemtype
5940   \tl_clear:N \l_tmpa_tl
5941   \clist_map_inline:Nn \l_tmpa_clist {
5942     \tl_if_exist:cT {\__problems_sproblem_##1_start:}{
5943       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_start:}}
5944     }
5945   }
5946   \tl_if_empty:NTF \l_tmpa_tl {
5947     \__problems_sproblem_start:
5948   }{
5949     \l_tmpa_tl
5950   }
5951   \stex_ref_new_doc_target:n \sproblemid
5952 }{
5953   \clist_set:No \l_tmpa_clist \sproblemtype
5954   \tl_clear:N \l_tmpa_tl
5955   \clist_map_inline:Nn \l_tmpa_clist {
5956     \tl_if_exist:cT {\__problems_sproblem_##1_end:}{
5957       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_end:}}
5958     }

```

```

5959 }
5960 \tl_if_empty:NTF \l_tmpa_tl {
5961   \__problems_sproblem_end:
5962 }{
5963   \l_tmpa_tl
5964 }
5965
5966
5967 \smallskip
5968 }
5969
5970
5971 \cs_new_protected:Nn \__problems_sproblem_start: {
5972   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
5973 }
5974 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
5975
5976 \newcommand\stexpatchproblem[3][] {
5977   \str_set:Nx \l_tmpa_str{ #1 }
5978   \str_if_empty:NTF \l_tmpa_str {
5979     \tl_set:Nn \__problems_sproblem_start: { #2 }
5980     \tl_set:Nn \__problems_sproblem_end: { #3 }
5981   }{
5982     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
5983     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
5984   }
5985 }
5986
5987
5988 \bool_if:NT \c__problems_boxed_bool {
5989   \surroundwithmdframed{problem}
5990 }

```

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

```

5991 \def\record@problem{
5992   \protected@write\@auxout{}
5993   {
5994     \string\@problem{\prob@number}
5995     {
5996       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
5997         \l__problems_inclprob_pts_tl
5998       }{
5999         \l__problems_prob_pts_tl
6000       }
6001     }%
6002     {
6003       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
6004         \l__problems_inclprob_min_tl
6005       }{
6006         \l__problems_prob_min_tl
6007       }
6008     }
6009   }
6010 }

```

(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).

```
6011 \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.

```
6012 \keys_define:nn { problem / solution }{
6013   id                .str_set_x:N = \l__problems_solution_id_str ,
6014   for               .tl_set:N    = \l__problems_solution_for_tl ,
6015   height            .dim_set:N   = \l__problems_solution_height_dim ,
6016   creators          .clist_set:N = \l__problems_solution_creators_clist ,
6017   contributors      .clist_set:N = \l__problems_solution_contributors_clist ,
6018   srccite           .tl_set:N    = \l__problems_solution_srccite_tl
6019 }
6020 \cs_new_protected:Nn \__problems_solution_args:n {
6021   \str_clear:N \l__problems_solution_id_str
6022   \tl_clear:N \l__problems_solution_for_tl
6023   \tl_clear:N \l__problems_solution_srccite_tl
6024   \clist_clear:N \l__problems_solution_creators_clist
6025   \clist_clear:N \l__problems_solution_contributors_clist
6026   \dim_zero:N \l__problems_solution_height_dim
6027   \keys_set:nn { problem / solution }{ #1 }
6028 }
```

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

```
6029 \newcommand\@startsolution[1][{}]{
6030   \__problems_solution_args:n { #1 }
6031   \@in@omtexttrue% we are in a statement.
6032   \bool_if:NF \c__problems_boxed_bool { \hrule }
6033   \smallskip\noindent
6034   {\textbf\prob@solution@kw : \enspace}
6035   \begin{small}
6036   \def\current@section@level{\prob@solution@kw}
6037   \ignorespacesandpars
6038 }
```

**\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.

```
6039 \newcommand\startsolutions{
6040   \specialcomment{solution}{\@startsolution}{
6041     \bool_if:NF \c__problems_boxed_bool {
6042       \hrule\medskip
6043     }
6044     \end{small}%
6045   }
6046   \bool_if:NT \c__problems_boxed_bool {
6047     \surroundwithmdframed{solution}
6048   }
6049 }
```



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

\stopsolutions

```
6050 \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.

```
6051 \ifsolutions
6052   \startsolutions
6053 \else
6054   \stopsolutions
6055 \fi
```

exnote

```
6056 \bool_if:NTF \c__problems_notes_bool {
6057   \newenvironment{exnote}[1][]{
6058     \par\smallskip\hrule\smallskip
6059     \noindent\textbf{\prob@note@kw : }\small
6060   }{
6061     \smallskip\hrule
6062   }
6063 }{
6064   \excludecomment{exnote}
6065 }
```

hint

```
6066 \bool_if:NTF \c__problems_notes_bool {
6067   \newenvironment{hint}[1][]{
6068     \par\smallskip\hrule\smallskip
6069     \noindent\textbf{\prob@hint@kw :~ }\small
6070   }{
6071     \smallskip\hrule
6072   }
6073 \newenvironment{exhint}[1][]{
6074   \par\smallskip\hrule\smallskip
6075   \noindent\textbf{\prob@hint@kw :~ }\small
6076 }{
6077   \smallskip\hrule
6078 }
6079 }{
6080   \excludecomment{hint}
6081   \excludecomment{exhint}
6082 }
```

gnote

```
6083 \bool_if:NTF \c__problems_notes_bool {
6084   \newenvironment{gnote}[1][]{
6085     \par\smallskip\hrule\smallskip
6086     \noindent\textbf{\prob@gnote@kw : }\small
6087   }{
6088     \smallskip\hrule
6089   }
6090 }{
6091   \excludecomment{gnote}
6092 }
```

## 40.3 Multiple Choice Blocks

EdN:23

mcb 23

```
6093 \newenvironment{mcb}{
6094   \begin{enumerate}
6095 }{
6096   \end{enumerate}
6097 }
```

we define the keys for the mcc macro

```
6098 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
6099   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
6100     \bool_set_true:N #1
6101   }{
6102     \bool_set_false:N #1
6103   }
6104 }
6105 \keys_define:nn { problem / mcc }{
6106   id          .str_set_x:N = \l__problems_mcc_id_str ,
6107   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
6108   T           .default:n   = { true } ,
6109   T           .bool_set:N  = \l__problems_mcc_t_bool ,
6110   F           .default:n   = { true } ,
6111   F           .bool_set:N  = \l__problems_mcc_f_bool ,
6112   Ttext       .code:n      = {
6113     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
6114   } ,
6115   Ftext       .code:n      = {
6116     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
6117   }
6118 }
6119 \cs_new_protected:Nn \l__problems_mcc_args:n {
6120   \str_clear:N \l__problems_mcc_id_str
6121   \tl_clear:N \l__problems_mcc_feedback_tl
6122   \bool_set_true:N \l__problems_mcc_t_bool
6123   \bool_set_true:N \l__problems_mcc_f_bool
6124   \bool_set_true:N \l__problems_mcc_Ttext_bool
6125   \bool_set_false:N \l__problems_mcc_Ftext_bool
6126   \keys_set:nn { problem / mcc }{ #1 }
6127 }
```

\mcc

```
6128 \newcommand\mcc[2][]{
6129   \l__problems_mcc_args:n{ #1 }
6130   \item #2
6131   \ifsolutions
6132     \\\
6133     \bool_if:NT \l__problems_mcc_t_bool {
6134       % TODO!
6135       % \ifcsstring{mcc@T}{T}{\mcc@Ttext}%
6136     }
6137     \bool_if:NT \l__problems_mcc_f_bool {
```

---

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

```

6138         % TODO!
6139         % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
6140     }
6141     \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
6142         !
6143     }{
6144         \l__problems_mcc_feedback_tl
6145     }
6146     \fi
6147 } %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.

```

6148
6149 \keys_define:nn{ problem / inclproblem }{
6150     id      .str_set:N = \l__problems_inclprob_id_str,
6151     pts     .tl_set:N  = \l__problems_inclprob_pts_tl,
6152     min     .tl_set:N  = \l__problems_inclprob_min_tl,
6153     title   .tl_set:N  = \l__problems_inclprob_title_tl,
6154     refnum  .int_set:N  = \l__problems_inclprob_refnum_int,
6155     type    .tl_set:N  = \l__problems_inclprob_type_tl,
6156     mhrepos .str_set:N  = \l__problems_inclprob_mhrepos_str
6157 }
6158 \cs_new_protected:Nn \l__problems_inclprob_args:n {
6159     \str_clear:N \l__problems_prob_id_str
6160     \tl_clear:N \l__problems_inclprob_pts_tl
6161     \tl_clear:N \l__problems_inclprob_min_tl
6162     \tl_clear:N \l__problems_inclprob_title_tl
6163     \tl_clear:N \l__problems_inclprob_type_tl
6164     \int_zero_new:N \l__problems_inclprob_refnum_int
6165     \str_clear:N \l__problems_inclprob_mhrepos_str
6166     \keys_set:nn { problem / inclproblem }{ #1 }
6167     \tl_if_empty:NT \l__problems_inclprob_pts_tl {
6168         \let\l__problems_inclprob_pts_tl\undefined
6169     }
6170     \tl_if_empty:NT \l__problems_inclprob_min_tl {
6171         \let\l__problems_inclprob_min_tl\undefined
6172     }
6173     \tl_if_empty:NT \l__problems_inclprob_title_tl {
6174         \let\l__problems_inclprob_title_tl\undefined
6175     }
6176     \tl_if_empty:NT \l__problems_inclprob_type_tl {
6177         \let\l__problems_inclprob_type_tl\undefined
6178     }
6179     \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
6180         \let\l__problems_inclprob_refnum_int\undefined
6181     }
6182 }

```

```

6183
6184 \cs_new_protected:Nn \__problems_inclprob_clear: {
6185   \let\l__problems_inclprob_id_str\undefined
6186   \let\l__problems_inclprob_pts_tl\undefined
6187   \let\l__problems_inclprob_min_tl\undefined
6188   \let\l__problems_inclprob_title_tl\undefined
6189   \let\l__problems_inclprob_type_tl\undefined
6190   \let\l__problems_inclprob_refnum_int\undefined
6191   \let\l__problems_inclprob_mhrepos_str\undefined
6192 }
6193 \__problems_inclprob_clear:
6194
6195 \newcommand\includeproblem[2][ ]{
6196   \__problems_inclprob_args:n{ #1 }
6197   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
6198     \input{#2}
6199   }{
6200     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
6201       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
6202     }
6203   }
6204   \__problems_inclprob_clear:
6205 }

```

(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.

```

6206 \AddToHook{enddocument}{
6207   \bool_if:NT \c__problems_pts_bool {
6208     \message{Total:~\arabic{pts}~points}
6209   }
6210   \bool_if:NT \c__problems_min_bool {
6211     \message{Total:~\arabic{min}~minutes}
6212   }
6213 }

```

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

```

6214 \def\pts#1{
6215   \bool_if:NT \c__problems_pts_bool {
6216     \marginpar{#1~\prob@pt@kw}
6217   }
6218 }
6219 \def\min#1{
6220   \bool_if:NT \c__problems_min_bool {
6221     \marginpar{#1~\prob@min@kw}
6222   }
6223 }

```

`\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.

```

6224 \newcounter{pts}
6225 \def\show@pts{
6226   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
6227     \bool_if:NT \c__problems_pts_bool {
6228       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
6229       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
6230     }
6231   }{
6232     \tl_if_exist:NT \l__problems_prob_pts_tl {
6233       \bool_if:NT \c__problems_pts_bool {
6234         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
6235         \addtocounter{pts}{\l__problems_prob_pts_tl}
6236       }
6237     }
6238   }
6239 }

```

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

`\show@min`

```

6240 \newcounter{min}
6241 \def\show@min{
6242   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
6243     \bool_if:NT \c__problems_min_bool {
6244       \marginpar{\l__problems_inclprob_min_tl\ min}
6245       \addtocounter{min}{\l__problems_inclprob_min_tl}
6246     }
6247   }{
6248     \tl_if_exist:NT \l__problems_prob_min_tl {
6249       \bool_if:NT \c__problems_min_bool {
6250         \marginpar{\l__problems_prob_min_tl\ min}
6251         \addtocounter{min}{\l__problems_prob_min_tl}
6252       }
6253     }
6254   }
6255 }
6256 \</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.

```
6257 <@@=hwexam>
6258 <*cls>
6259 \ProvidesExplClass{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
6260 \RequirePackage{l3keys2e,expl-keystr-compatible}
6261 \DeclareOption*{
6262   \PassOptionsToClass{\CurrentOption}{document-structure}
6263   \PassOptionsToPackage{\CurrentOption}{stex}
6264   \PassOptionsToPackage{\CurrentOption}{hwexam}
6265   \PassOptionsToPackage{\CurrentOption}{tikzinput}
6266 }
6267 \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.

```
6268 \LoadClass{document-structure}
6269 \RequirePackage{stex}
6270 \RequirePackage{hwexam}
6271 \RequirePackage{tikzinput}
6272 \RequirePackage{graphicx}
6273 \RequirePackage{a4wide}
6274 \RequirePackage{amssymb}
6275 \RequirePackage{amstext}
6276 \RequirePackage{amsmath}
```

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

```

6277 \newcommand\assig@default@type{\hwexam@assignment@kw}
6278 \def\document@hwexamtype{\assig@default@type}
6279 <@@=document_structure>
6280 \keys_define:nn { document-structure / document }{
6281 id .str_set_x:N = \c_document_structure_document_id_str,
6282 hwexamtype .tl_set:N = \document@hwexamtype
6283 }
6284 <@@=hwexam>
6285 </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.

```
6286 \*package>
6287 \ProvidesExplPackage{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
6288 \RequirePackage{l3keys2e,expl-keystr-compat}
6289
6290 \newif\iftest\testfalse
6291 \DeclareOption{test}{\testtrue}
6292 \newif\ifmultiple\multiplefalse
6293 \DeclareOption{multiple}{\multipletrue}
6294 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
6295 \ProcessOptions
```

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

```
6296 \RequirePackage{keyval}[1997/11/10]
6297 \RequirePackage{problem}
```

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

```
6298 \newcommand\hwexam@assignment@kw{Assignment}
6299 \newcommand\hwexam@given@kw{Given}
6300 \newcommand\hwexam@due@kw{Due}
6301 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
6302 blank~for~extra~space}
6303 \def\hwexam@minutes@kw{minutes}
6304 \newcommand\correction@probs@kw{prob.}
6305 \newcommand\correction@pts@kw{total}
6306 \newcommand\correction@reached@kw{reached}
6307 \newcommand\correction@sum@kw{Sum}
6308 \newcommand\correction@grade@kw{grade}
6309 \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

```

6310 \AddToHook{begindocument}{
6311 \ltx@ifpackageloaded{babel}{
6312 \makeatletter
6313 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6314 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6315 \input{hwexam-ngerman.ldf}
6316 }
6317 \clist_if_in:NnT \l_tmpa_clist {finnish}{
6318 \input{hwexam-finnish.ldf}
6319 }
6320 \clist_if_in:NnT \l_tmpa_clist {french}{
6321 \input{hwexam-french.ldf}
6322 }
6323 \clist_if_in:NnT \l_tmpa_clist {russian}{
6324 \input{hwexam-russian.ldf}
6325 }
6326 \makeatother
6327 }{}
6328 }
6329

```

## 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.

```

6330 \newcounter{assignment}
6331 \numberproblemsin{assignment}
6332 \renewcommand\prob@label[1]{\assignment@number.#1}

```

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

```

6333 \keys_define:nn { hwexam / assignment } {
6334 id .str_set:N = \l__hwexam_assign_id_str,
6335 number .int_set:N = \l__hwexam_assign_number_int,
6336 title .tl_set:N = \l__hwexam_assign_title_tl,
6337 type .tl_set:N = \l__hwexam_assign_type_tl,
6338 given .tl_set:N = \l__hwexam_assign_given_tl,
6339 due .tl_set:N = \l__hwexam_assign_due_tl,
6340 loadmodules .code:n = {
6341 \bool_set_true:N \l__hwexam_assign_loadmodules_bool
6342 }
6343 }
6344 \cs_new_protected:Nn \__hwexam_assignment_args:n {
6345 \str_clear:N \l__hwexam_assign_id_str
6346 \int_set:Nn \l__hwexam_assign_number_int {-1}
6347 \tl_clear:N \l__hwexam_assign_title_tl
6348 \tl_clear:N \l__hwexam_assign_type_tl
6349 \tl_clear:N \l__hwexam_assign_given_tl
6350 \tl_clear:N \l__hwexam_assign_due_tl
6351 \bool_set_false:N \l__hwexam_assign_loadmodules_bool

```

```

6352 \keys_set:nn { hwexam / assignment }{ #1 }
6353 }

```

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.

```

6354 \newcommand\given@due[2]{
6355 \bool_lazy_all:nF {
6356 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
6357 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
6358 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
6359 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
6360 }{ #1 }
6361
6362 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
6363 \tl_if_empty:NF \l__hwexam_assign_given_tl {
6364 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
6365 }
6366 }{
6367 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
6368 }
6369
6370 \bool_lazy_or:nnF {
6371 \bool_lazy_and_p:nn {
6372 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
6373 }{
6374 \tl_if_empty_p:V \l__hwexam_assign_due_tl
6375 }
6376 }{
6377 \bool_lazy_and_p:nn {
6378 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
6379 }{
6380 \tl_if_empty_p:V \l__hwexam_assign_due_tl
6381 }
6382 }{ ,~ }
6383
6384 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
6385 \tl_if_empty:NF \l__hwexam_assign_due_tl {
6386 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
6387 }
6388 }{
6389 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
6390 }
6391
6392 \bool_lazy_all:nF {
6393 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
6394 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
6395 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
6396 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
6397 }{ #2 }
6398 }

```

`\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.

```

6399 \newcommand\assignment@title[3]{
6400 \tl_if_empty:NTF \l__hwexam_inclasssign_title_tl {
6401 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
6402 #1
6403 }{
6404 #2\l__hwexam_assign_title_tl#3
6405 }
6406 }{
6407 #2\l__hwexam_inclasssign_title_tl#3
6408 }
6409 }

```

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

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

```

6410 \newcommand\assignment@number{
6411 \int_compare:nNnTF \l__hwexam_inclasssign_number_int = {-1} {
6412 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
6413 \arabic{assignment}
6414 } {
6415 \int_use:N \l__hwexam_assign_number_int
6416 }
6417 }{
6418 \int_use:N \l__hwexam_inclasssign_number_int
6419 }
6420 }

```

(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.

```

6421 \newenvironment{assignment}[1][ ]{
6422 \__hwexam_assignment_args:n { #1 }
6423 %\sref@target
6424 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
6425 \global\stepcounter{assignment}
6426 }{
6427 \global\setcounter{assignment}{\int_use:N\l__hwexam_assign_number_int}
6428 }
6429 \setcounter{problem}{0}
6430 \def\current@section@level{\document@hwexamtype}
6431 %\sref@label@id{\document@hwexamtype \thesection}
6432 \begin{@assignment}
6433 }{
6434 \end{@assignment}
6435 }

```

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

```

6436 \def\ass@title{
6437 \protect\document@hwexamtype~\arabic{assignment}
6438 \assignment@title{}\{;\}\{;\} -- \given@due{}\{;\}
6439 }
6440 \ifmultiple
6441 \newenvironment{@assignment}{
6442 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
6443 \begin{omgroup}[loadmodules]{\ass@title}
6444 }{
6445 \begin{omgroup}{\ass@title}
6446 }
6447 }{
6448 \end{omgroup}
6449 }

```

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

```

6450 \else
6451 \newenvironment{@assignment}{
6452 \begin{center}\bf
6453 \Large@title\strut\
6454 \document@hwexamtype~\arabic{assignment}\assignment@title{}\{;\}\{;\}\{;\}
6455 \large\given@due{--;\}\{;\}--}
6456 \end{center}
6457 }{}
6458 \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.

```

6459 \keys_define:nn { hwexam / inclassignment } {
6460 %id .str_set_x:N = \l__hwexam_assign_id_str,
6461 number .int_set:N = \l__hwexam_inclassign_number_int,
6462 title .tl_set:N = \l__hwexam_inclassign_title_tl,
6463 type .tl_set:N = \l__hwexam_inclassign_type_tl,
6464 given .tl_set:N = \l__hwexam_inclassign_given_tl,
6465 due .tl_set:N = \l__hwexam_inclassign_due_tl,
6466 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
6467 }
6468 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
6469 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
6470 \tl_clear:N \l__hwexam_inclassign_title_tl
6471 \tl_clear:N \l__hwexam_inclassign_type_tl
6472 \tl_clear:N \l__hwexam_inclassign_given_tl
6473 \tl_clear:N \l__hwexam_inclassign_due_tl
6474 \str_clear:N \l__hwexam_inclassign_mhrepos_str
6475 \keys_set:nn { hwexam / inclassignment }{ #1 }
6476 }
6477 \__hwexam_inclassignment_args:n {}
6478
6479 \newcommand\inputassignment[2][ ]{

```

```

6480 \_hwexam_inclassnment_args:n { #1 }
6481 \str_if_empty:NTF \l__hwexam_inclassn_mhrepos_str {
6482 \input{#2}
6483 }{
6484 \stex_in_repository:nn{\l__hwexam_inclassn_mhrepos_str}{
6485 \input{\mhp{path}\l__hwexam_inclassn_mhrepos_str}{#2}}
6486 }
6487 }
6488 \_hwexam_inclassnment_args:n {}
6489 }
6490 \newcommand\includeassignment[2][ ]{
6491 \newpage
6492 \inputassignment[#1]{#2}
6493 }

```

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

## 42.4 Typesetting Exams

\quizheading

```

6494 \ExplSyntaxOff
6495 \newcommand\quizheading[1]{%
6496 \def\@tas{#1}%
6497 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
6498 \ifx\@tas\@empty\else%
6499 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
6500 \fi%
6501 }
6502 \ExplSyntaxOn

```

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

\testheading

```

6503
6504 \def\hwexamheader{\input{hwexam-default.header}}
6505
6506 \def\hwexamminutes{
6507 \tl_if_empty:NTF \testheading@duration {
6508 {\testheading@min}~\hwexam@minutes@kw
6509 }{
6510 \testheading@duration
6511 }
6512 }
6513
6514 \keys_define:nn { hwexam / testheading } {
6515 min .tl_set:N = \testheading@min,
6516 duration .tl_set:N = \testheading@duration,
6517 reqpts .tl_set:N = \testheading@reqpts,
6518 tools .tl_set:N = \testheading@tools
6519 }
6520 \cs_new_protected:Nn \_hwexam_testheading_args:n {
6521 \tl_clear:N \testheading@min
6522 \tl_clear:N \testheading@duration

```

```

6523 \tl_clear:N \testheading@reqpts
6524 \tl_clear:N \testheading@tools
6525 \keys_set:nn { hwexam / testheading }{ #1 }
6526 }
6527 \newenvironment{testheading}[1][]{
6528   \_hwexam_testheading_args:n{ #1 }
6529   \newcount\check@time\check@time=\testheading@min
6530   \advance\check@time by -\theassignment@totalmin
6531   \newif\if@bonuspoints
6532   \tl_if_empty:NTF \testheading@reqpts {
6533     \@bonuspointsfalse
6534   }{
6535     \newcount\bonus@pts
6536     \bonus@pts=\theassignment@totalpts
6537     \advance\bonus@pts by -\testheading@reqpts
6538     \edef\bonus@pts{\the\bonus@pts}
6539     \@bonuspointstrue
6540   }
6541   \edef\check@time{\the\check@time}
6542
6543   \makeatletter\hwexamheader\makeatother
6544 }{
6545   \newpage
6546 }

```

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

\testspace

```

6547 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

6548 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

6549 \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.

```

6550 <@=problems>
6551 \renewcommand\@problem[3]{
6552   \stepcounter{assignment@probs}
6553   \def\__problemspts{#2}
6554   \ifx\__problemspts\@empty\else
6555     \addtocounter{assignment@totalpts}{#2}
6556   \fi
6557   \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
6558   \xdef\correction@probs{\correction@probs & #1}%
6559   \xdef\correction@pts{\correction@pts & #2}
6560   \xdef\correction@reached{\correction@reached &}

```

```

6561 }
6562 <@@=hwexam>

```

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

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

```

6563 \newcounter{assignment@probs}
6564 \newcounter{assignment@totalpts}
6565 \newcounter{assignment@totalmin}
6566 \def\correction@probs{\correction@probs@kw}
6567 \def\correction@pts{\correction@pts@kw}
6568 \def\correction@reached{\correction@reached@kw}
6569 \stepcounter{assignment@probs}
6570 \newcommand\correction@table{
6571 \resizebox{\textwidth}{!}{%
6572 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
6573 &\multicolumn{\theassignment@probs}{c|}||%|
6574 {\footnotesize\correction@forgrading@kw} &\\ \hline
6575 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
6576 \correction@pts & \theassignment@totalpts & \\ \hline
6577 \correction@reached & & \[.7cm]\hline
6578 \end{tabular}}
6579 </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}

```