

# The sTeX3 Package \*

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

2022-02-20

## 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 2022-02-20)

# 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 sTeX</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
9.1.1	HTML Annotations . . . . .	20
9.1.2	Babel Languages . . . . .	21
9.1.3	Auxiliary Methods . . . . .	21

<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
10.1.3	Using Content in Archives . . . . .	24
<b>11</b>	<b>sTeX-References</b>	<b>25</b>
11.1	Macros and Environments . . . . .	25
11.1.1	Setting Reference Targets . . . . .	25
11.1.2	Using References . . . . .	26
<b>12</b>	<b>sTeX-Modules</b>	<b>27</b>
12.1	Macros and Environments . . . . .	27
12.1.1	The <code>smodule</code> environment . . . . .	29
<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>34</b>
14.1	Macros and Environments . . . . .	34
<b>15</b>	<b>sTeX-Terms</b>	<b>36</b>
15.1	Macros and Environments . . . . .	36
<b>16</b>	<b>sTeX-Structural Features</b>	<b>38</b>
16.1	Macros and Environments . . . . .	38
16.1.1	Structures . . . . .	38
<b>17</b>	<b>sTeX-Statements</b>	<b>39</b>
17.1	Macros and Environments . . . . .	39
<b>18</b>	<b>sTeX-Proofs: Structural Markup for Proofs</b>	<b>40</b>
18.1	Introduction . . . . .	42
18.2	The User Interface . . . . .	43
18.2.1	Package Options . . . . .	43
18.2.2	Proofs and Proof steps . . . . .	43
18.2.3	Justifications . . . . .	43
18.2.4	Proof Structure . . . . .	44
18.2.5	Proof End Markers . . . . .	45
18.2.6	Configuration of the Presentation . . . . .	45
18.3	Limitations . . . . .	45
<b>19</b>	<b>sTeX-Metatheory</b>	<b>47</b>
19.1	Symbols . . . . .	47
<b>III</b>	<b>Extensions</b>	<b>48</b>

<b>20</b>	<b>Tikzinput</b>	<b>49</b>
20.1	Macros and Environments . . . . .	49
<b>21</b>	<b>document-structure: Semantic Markup for Open Mathematical Documents in L<sup>A</sup>T<sub>E</sub>X</b>	<b>50</b>
21.1	Introduction . . . . .	50
21.2	The User Interface . . . . .	51
21.2.1	Package and Class Options . . . . .	51
21.2.2	Document Structure . . . . .	51
21.2.3	Ignoring Inputs . . . . .	53
21.2.4	Structure Sharing . . . . .	53
21.2.5	Global Variables . . . . .	53
21.2.6	Colors . . . . .	54
21.3	Limitations . . . . .	54
<b>22</b>	<b>NotesSlides – Slides and Course Notes</b>	<b>55</b>
22.1	Introduction . . . . .	55
22.2	The User Interface . . . . .	55
22.2.1	Package Options . . . . .	55
22.2.2	Notes and Slides . . . . .	56
22.2.3	Header and Footer Lines of the Slides . . . . .	57
22.2.4	Frame Images . . . . .	57
22.2.5	Colors and Highlighting . . . . .	58
22.2.6	Front Matter, Titles, etc. . . . .	58
22.2.7	Excursions . . . . .	58
22.2.8	Miscellaneous . . . . .	59
22.3	Limitations . . . . .	59
<b>23</b>	<b>problem.sty: An Infrastructure for formatting Problems</b>	<b>60</b>
23.1	Introduction . . . . .	60
23.2	The User Interface . . . . .	60
23.2.1	Package Options . . . . .	60
23.2.2	Problems and Solutions . . . . .	61
23.2.3	Multiple Choice Blocks . . . . .	62
23.2.4	Including Problems . . . . .	62
23.2.5	Reporting Metadata . . . . .	62
23.3	Limitations . . . . .	62
<b>24</b>	<b>hwexam.sty/cls: An Infrastructure for formatting Assignments and Exams</b>	<b>64</b>
24.1	Introduction . . . . .	65
24.2	The User Interface . . . . .	65
24.2.1	Package and Class Options . . . . .	65
24.2.2	Assignments . . . . .	65
24.2.3	Typesetting Exams . . . . .	65
24.2.4	Including Assignments . . . . .	66
24.3	Limitations . . . . .	66
<b>IV</b>	<b>Implementation</b>	<b>68</b>

<b>25</b>	<b>STeX-Basics Implementation</b>	<b>69</b>
25.1	The STeXDocument Class . . . . .	69
25.2	Preliminaries . . . . .	69
25.3	Messages and logging . . . . .	70
25.4	HTML Annotations . . . . .	71
25.5	Babel Languages . . . . .	74
25.6	Auxiliary Methods . . . . .	75
<b>26</b>	<b>STeX-MathHub Implementation</b>	<b>76</b>
26.1	Generic Path Handling . . . . .	76
26.2	PWD and kpsewhich . . . . .	78
26.3	File Hooks and Tracking . . . . .	79
26.4	MathHub Repositories . . . . .	80
26.5	Using Content in Archives . . . . .	84
<b>27</b>	<b>STeX-References Implementation</b>	<b>89</b>
27.1	Document URIs and URLs . . . . .	89
27.2	Setting Reference Targets . . . . .	91
27.3	Using References . . . . .	93
<b>28</b>	<b>STeX-Modules Implementation</b>	<b>96</b>
28.1	The smodule environment . . . . .	100
28.2	Invoking modules . . . . .	105
<b>29</b>	<b>STeX-Module Inheritance Implementation</b>	<b>107</b>
29.1	SMS Mode . . . . .	107
29.2	Inheritance . . . . .	110
<b>30</b>	<b>STeX-Symbols Implementation</b>	<b>115</b>
30.1	Symbol Declarations . . . . .	115
30.2	Notations . . . . .	122
30.3	Variables . . . . .	131
<b>31</b>	<b>STeX-Terms Implementation</b>	<b>136</b>
31.1	Symbol Invocations . . . . .	136
31.2	Terms . . . . .	141
31.3	Notation Components . . . . .	147
31.4	Variables . . . . .	149
<b>32</b>	<b>STeX-Structural Features Implementation</b>	<b>151</b>
32.1	Imports with modification . . . . .	151
32.2	The feature environment . . . . .	158
32.3	Features . . . . .	159
<b>33</b>	<b>STeX-Statements Implementation</b>	<b>165</b>
33.1	Definitions . . . . .	165
33.2	Assertions . . . . .	170
33.3	Examples . . . . .	173
33.4	Logical Paragraphs . . . . .	175

<b>34 The Implementation</b>	<b>181</b>
34.1 Package Options . . . . .	181
34.2 Proofs . . . . .	181
34.3 Justifications . . . . .	187
<b>35 <math>\text{\LaTeX}</math>-Others Implementation</b>	<b>189</b>
<b>36 <math>\text{\LaTeX}</math>-Metatheory Implementation</b>	<b>190</b>
<b>37 Tikzinput Implementation</b>	<b>193</b>
<b>38 document-structure.sty Implementation</b>	<b>195</b>
38.1 The document-structure Class . . . . .	195
38.2 Class Options . . . . .	195
38.3 Beefing up the <code>document</code> environment . . . . .	196
38.4 Implementation: document-structure Package . . . . .	196
38.5 Package Options . . . . .	196
38.6 Document Structure . . . . .	198
38.7 Front and Backmatter . . . . .	201
38.8 Global Variables . . . . .	203
<b>39 NotesSlides – Implementation</b>	<b>204</b>
39.1 Class and Package Options . . . . .	204
39.2 Notes and Slides . . . . .	206
39.3 Header and Footer Lines . . . . .	210
39.4 Frame Images . . . . .	211
39.5 Colors and Highlighting . . . . .	212
39.6 Sectioning . . . . .	213
39.7 Excursions . . . . .	215
<b>40 The Implementation</b>	<b>217</b>
40.1 Package Options . . . . .	217
40.2 Problems and Solutions . . . . .	218
40.3 Multiple Choice Blocks . . . . .	224
40.4 Including Problems . . . . .	225
40.5 Reporting Metadata . . . . .	226
<b>41 Implementation: The hwexam Class</b>	<b>228</b>
41.1 Class Options . . . . .	228
<b>42 Implementation: The hwexam Package</b>	<b>230</b>
42.1 Package Options . . . . .	230
42.2 Assignments . . . . .	231
42.3 Including Assignments . . . . .	234
42.4 Typesetting Exams . . . . .	235
42.5 Leftovers . . . . .	237

**Part I**  
**Manual**

# Chapter 1

## What is sTeX?

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

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

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

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

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

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



# Chapter 2

## Quickstart

### 2.1 Setup

#### 2.1.1 The sTeX IDE

TODO: VSCode Plugin

#### 2.1.2 Manual Setup

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

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

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

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

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

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

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

---

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

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

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

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

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

The document we will consider is the following:

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

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

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

Compiling this document with `pdflatex` should yield the output

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

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

### \usemodule

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

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

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

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

---

`\symref`  
`\symname`

---

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

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

---

`\importmodule`

---

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

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

## Chapter 3

# Using $\text{\LaTeX}$

Both the `stex` package and document class offer the following options:

**lang** ( $\langle\textit{language}\rangle*$ ) Languages to load with the `babel` package.

**mathhub** ( $\langle\textit{directory}\rangle$ ) MathHub folder to search for repositories.

**sms** ( $\langle\textit{boolean}\rangle$ ) use *persisted* mode (not yet implemented).

**image** ( $\langle\textit{boolean}\rangle$ ) passed on to `tikzinput`.

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

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

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

## Chapter 5

# Creating New Modules and Symbols

TODO

### Example 1

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

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

## 5.1 Advanced Structuring Mechanisms

Given modules:

### Example 2

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

Module 2:  
Module 3:  
Module 4:

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

### Example 3

```
\begin{smodule}{ring}
\begin{copymodule}{group}{addition}
\renamedcl[name=universe]{universe}{runiverse}
\renamedcl[name=plus]{operation}{rplus}
\renamedcl[name=zero]{unit}{rzero}
\renamedcl[name=uminus]{inverse}{rminus}
\end{copymodule}
\notation*[plus,op=+,prec=60]{rplus}{#1 \comp+ #2}
\notation*[zero]{rzero}{\comp0}
\notation*[uminus,op=-]{rminus}{\comp- #1}
\begin{copymodule}{monoid}{multiplication}
\assign{universe}{\runiverse}
\renamedcl[name=times]{operation}{rtimes}
\renamedcl[name=one]{unit}{rone}
\end{copymodule}
\notation*[cdot,op=\cdot,prec=50]{rtimes}{#1 \comp\cdot #2}
\notation*[one]{rone}{\comp1}
Test: $\rtimes a{\rplus c}{\rtimes de}$
\end{smodule}
```

Module 5: Test:  $a \cdot (c + d \cdot e)$

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 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

```
 $\mult*\{\arg{a}\comp{\ast}\arg{b}\}$  is the
\mult{\comp{product of} \arg{a} \comp{and} \arg{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} \arg*\mathmult{a}{b}} again by \arg{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{\arg[2]{The proposition  $P$ } \comp{holds for every} \arg[1]{ $x \in A$ }}
```

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

<sup>4</sup>EdNOTE: TODO

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

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

### Example 10

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

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

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

### Example 11

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

$\textcolor{teal}{Multiplication}$  (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  $A$  operator precedence should be smaller than  $B$  argument precedences.

For example:

**Module 9:**

### Example 14

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

$a+b\cdot c$  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

This sub package provides general set up code, auxiliary methods and abstractions for xhtml annotations.

### 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 {&lt;log-prefix&gt;} {&lt;message&gt;}</code>
-----------------------------	--

---

Logs *<message>*, if the package option `debug` contains *<log-prefix>*.

#### 9.1.1 HTML Annotations

---

<code>\if@latexml</code>	L <sup>A</sup> T <sub>E</sub> X2e conditional for L <sup>A</sup> T <sub>E</sub> XML
--------------------------	---

---

---

<code>\latexml_if_p: *</code>	L <sup>A</sup> T <sub>E</sub> X3 conditionals for L <sup>A</sup> T <sub>E</sub> XML.
<code>\latexml_if:TF *</code>	

---

---

<code>\stex_if_do_html_p: *</code>	Whether to currently produce any HTML annotations (can be false in some advanced structuring environments, for example)
<code>\stex_if_do_html:TF *</code>	

---

---

<code>\stex_suppress_html:n</code>	Temporarily disables HTML annotations in its argument 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> .
--------------------------------	--

### 9.1.2 Babel Languages

---

<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`.

### 9.1.3 Auxiliary Methods

---

<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>\ignorespacesandpars</code>	ignores white space characters and <code>\par</code> control sequences. Expands tokens in the process.
-----------------------------------	--

---

# Chapter 10

## STEX-MathHub

This sub package provides code for handling ST<sub>E</sub>X archives, files, file paths and related 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> $\langle path-variable \rangle$ $\{ \langle string \rangle \}$ turns the $\langle string \rangle$ into a path by splitting it at <code>/</code> -characters and stores the result in $\langle path-variable \rangle$ . Also applies <code>\stex_path_canonicalize:N</code> .
--	--

---

---

<code>\stex_path_to_string:NN</code> <code>\stex_path_to_string:N</code>	The inverse; turns a path into a string and stores it in the second argument variable, or 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:N<math>\underline{T}</math></code> $\star$	Checks whether the path provided is <i>absolute</i> , i.e. starts with an empty segment
--	---

---

---

<code>\c_stex_pwd_seq</code> <code>\c_stex_pwd_str</code> <code>\c_stex_mainfile_seq</code> <code>\c_stex_mainfile_str</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>\g_stex_currentfile_seq</code>	The file being currently processed (respecting <code>\input</code> etc.)
--------------------------------------	--

---



---

<code>\stex_filestack_push:n</code>	Push and pop (repectively) a file path to the file stack, to keep track of the current file.
<code>\stex_filestack_pop:</code>	Are called in hooks <code>file/before</code> and <code>file/after</code> , respectively.

---

### 10.1.2 MathHub Archives

---

<code>\mathhub</code>	We determine the path to the local MathHub folder via one of three means, in order of precedence:
<code>\c_stex_mathhub_seq</code>	
<code>\c_stex_mathhub_str</code>	

---

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.

---

<code>\l_stex_current_repository_prop</code>
--

---

Always points to the *current* MathHub repository (if we currently are in one). Has the following fields corresponding to the entries in the `MANIFEST.MF`-file:

- `id`: The name of the archive, including its group (e.g. `smglom/calculus`),
- `ns`: The content namespace (for modules and symbols),
- `narr`: the narration namespace (for document references),
- `docurl`: The URL that is used as a basis for *external references*,
- `deps`: All archives that this archive depends on (currently not in use).

---

<code>\stex_set_current_repository:n</code>
---

---

Sets the current repository to the one with the provided ID. calls `\__stex_mathhub_do_manifest:n`, so works whether this repository's `MANIFEST.MF`-file has already been read or not.

---

<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.
---	--

---



---

<code>\stex_in_repository:nn</code>	<code>\stex_in_repository:nn{&lt;repository-name&gt;}{&lt;code&gt;}</code>
-------------------------------------	--

---

Change the current repository to `{<repository-name>}` (or not, if `{<repository-name>}` is empty), and passes its ID on to `{<code>}` as `#1`. Switches back to the previous repository after executing `{<code>}`.

### 10.1.3 Using Content in Archives

<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> <code>\mhinput</code>	<code>\inputref[&lt;archive-ID&gt;]{&lt;filename&gt;}</code>  Both <code>\input</code> the file <code>&lt;filename&gt;</code> in archive <code>&lt;archive-ID&gt;</code> (relative to the <code>source-</code> subdirectory). <code>\mhinput</code> does so directly. <code>\inputref</code> does so within an <code>\begingroup... \endgroup-</code> block, and skips it in <code>html-mode</code> , inserting a <i>reference</i> to the file instead. Both also set <code>\ifinputref</code> to true.
<hr/> <hr/> <code>\addmhbibresource</code>	<code>\inputref[&lt;archive-ID&gt;]{&lt;filename&gt;}</code>  Adds a <code>.bib</code> -file <code>&lt;filename&gt;</code> in archive <code>&lt;archive-ID&gt;</code> (relative to the top-directory of the archive!).
<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(s) (if existent) in descending order. Throws an error if no file by that name exists in any of the relevant <code>lib</code> -folders.
<hr/> <hr/> <code>\libusepackage</code>	<code>\libusepackage[&lt;args&gt;]{&lt;filename&gt;}</code>  Like <code>\libinput</code> , but looks for <code>.sty</code> -files and calls <code>\usepackage[&lt;meta{args}&gt;]{&lt;Arg{filename}&gt;}</code> instead of <code>\input</code> . Throws an error, if none or more than one suitable package file is found.
<hr/> <hr/> <code>\mhgraphics</code> <code>\cmhgraphics</code>	<i>If</i> the <code>graphicx</code> package is loaded, these macros are defined at <code>\begin{document}</code> . <code>\mhgraphics</code> takes the same arguments as <code>\includegraphics</code> , with the additional optional key <code>mhrefpos</code> . It then resolves the file path in <code>\mhgraphics[mhrefpos=Foo/Bar]{foo/bar.png}</code> relative to the <code>source-</code> folder of the <code>Foo/Bar</code> -archive. <code>\cmhgraphics</code> additionally wraps the image in a <code>center</code> -environment.
<hr/> <hr/> <code>\lstinputmhlisting</code> <code>\clstinputmhlisting</code>	Like <code>\mhgraphics</code> , but only defined if the <code>listings</code> -package is loaded, and with <code>\lstinputlisting</code> instead of <code>\includegraphics</code> .

# Chapter 11

## STEX-References

This sub package contains code related to links and cross-references

### 11.1 Macros and Environments

---

---

**\STEXreftitle****\STEXreftitle{<some title>}**

Sets the title of the current document to *<some title>*. A reference to the current document from *some other* document will then be displayed accordingly. e.g. if **\STEXreftitle{foo book}** is called, then referencing Definition 3.5 in this document in another document will display **Definition 3.5 in foo book**.

---

---

**\stex\_get\_document\_uri:**

Computes the current document uri from the current archive's **narr**-field and its location relative to the archive's **source**-directory. Reference targets are computed from this URI and the reference-id.

---

---

**\l\_stex\_current\_docns\_str**

Stores its result in **\l\_stex\_current\_docns\_str**

---

---

**\stex\_get\_document\_url:**

Computes the current URL from the current archive's **docurl**-field and its location relative to the archive's **source**-directory. Reference targets are computed from this URL and the reference-id, if this document is only included in SMS mode.

---

---

**\l\_stex\_current\_docurl\_str**

Stores its result in **\l\_stex\_current\_docurl\_str**

#### 11.1.1 Setting Reference Targets

---

---

**\stex\_ref\_new\_doc\_target:n****\stex\_ref\_new\_doc\_target:n{<id>}**

Sets a new reference target with id *<id>*.

---

---

**\stex\_ref\_new\_sym\_target:n****\stex\_ref\_new\_sym\_target:n{<uri>}**

Sets a new reference target for the symbol *<uri>*.

### 11.1.2 Using References

---

<u><code>\sref</code></u>	<code>\sref[<i>&lt;opt-args&gt;</i>]{<i>&lt;id&gt;</i>}</code>
---------------------------	--

References the label with if *<id>*. Optional arguments: TODO

---

<u><code>\srefsym</code></u>	<code>\srefsym[<i>&lt;opt-args&gt;</i>]{<i>&lt;symbol&gt;</i>}</code>
------------------------------	---

Like `\sref`, but references the *canonical label* for the provided symbol. The canonical target is the last of the following occurring in the document:

- A `\definiendum` or `\definame` for *<symbol>*,
- The `sassertion`, `sexample` or `sparagraph` with `for=<symbol>` that generated *<symbol>* in the first place, or
- A `\sparagraph` with `type=symdoc` and `for=<symbol>`.

---

<u><code>\srefsymuri</code></u>	<code>\srefsymuri{<i>&lt;URI&gt;</i>}{<i>&lt;text&gt;</i>}</code>
---------------------------------	---

A convenient short-hand for `\srefsym[linktext={<text>}]<URI>`, but requires the first argument to be a full URI already. Intended to be used in e.g. `\compemph@uri`, `\defemph@uri`, etc.



# Chapter 12

## STEX-Modules

This sub package contains code related to Modules

### 12.1 Macros and Environments

The content of a module with uri  $\langle <URI> \rangle$  is stored in four macros. All modifications of these macros are global:

---

---

`\c_stex_module_<URI>_prop`

A property list with the following fields:

**name** The *name* of the module,

**ns** the *namespace* in field **ns**,

**file** the *file* containing the module, as a sequence of path fragments

**lang** the module's *language*,

**sig** the language of the signature module, if the current file is a translation from some other language,

**deprecate** if this module is deprecated, the module that replaces it,

**meta** the metatheory of the module.

---

---

`\c_stex_module_<URI>_code`

The code to execute when this module is activated (i.e. imported), e.g. to set all the semantic macros, notations, etc.

---

---

`\c_stex_module_<URI>_constants`

The names of all constants declared in the module

---

---

`\c_stex_module_<URI>_constants`

The full URIs of all modules imported in this module

<hr/> <hr/> <code>\l_stex_current_module_str</code>	<code>\l_stex_current_module_str</code> always contains the URI of the current module (if existent).
<hr/> <hr/> <code>\l_stex_all_modules_seq</code>	Stores full URIs for all modules currently in scope.
<hr/> <hr/> <code>\stex_if_in_module_p: *</code> <code>\stex_if_in_module:TF *</code>	Conditional for whether we are currently in a module
<hr/> <hr/> <code>\stex_if_module_exists_p:n *</code> <code>\stex_if_module_exists:nTF *</code>	Conditional for whether a module with the provided URI is already known.
<hr/> <hr/> <code>\stex_add_to_current_module:n</code> <code>\STEXexport</code>	Adds the provided tokens to the <code>_code</code> control sequence of the current module. <code>\stex_add_to_current_module:n</code> is used internally, <code>\STEXexport</code> is intended for users and additionally executes the provided code immediately.
<hr/> <hr/> <code>\stex_add_constant_to_current_module:n</code>	Adds the declaration with the provided name to the <code>_constants</code> control sequence of the current module.
<hr/> <hr/> <code>\stex_add_import_to_current_module:n</code>	Adds the module with the provided full URI to the <code>_imports</code> control sequence of the current module.
<hr/> <hr/> <code>\stex_collect_imports:n</code>	Iterates over all imports of the provided (full URI of a) module and stores them as a topologically sorted list – including the provided module as the last element – in <code>\l_stex_collect_imports_seq</code>
<hr/> <hr/> <code>\stex_do_up_to_module:n</code>	Code that is <i>exported</i> from module (such as symbol declarations) should be local <i>to the current module</i> . For that reason, ideally all symbol declarations and similar commands should be called directly in the module environment, however, that is not always feasible, e.g. in structural features or <code>sparapraphs</code> . <code>\stex_do_up_to_module</code> therefore executes the provided code repeatedly in an <code>\aftergroup</code> up until the group level is equal to that of the innermost smodule environment.

---

---

`\stex_modules_current_namespace:`

Computes the current namespace as follows:

If the current file is `.../source/sub/file.tex` in some archive with namespace `http://some.namespace/foo`, then the namespace of is `http://some.namespace/foo/sub/file`. Otherwise, the namespace is the absolute file path of the current file (i.e. starting with `file:///`).

The result is stored in `\l_stex_modules_ns_str`. Additionally, the sub path relative to the current repository is stored in `\l_stex_modules_subpath_str`.

### 12.1.1 The `smodule` environment

`module`            `\begin{module}[\langle options \rangle]{\langle name \rangle}`

Opens a new module with name `\langle name \rangle`. Options are:

`title` (`\langle token list \rangle`) to display in customizations.

`type` (`\langle string \rangle*`) for use in customizations.

`deprecate` (`\langle module \rangle`) if set, will throw a warning when loaded, urging to use `\langle module \rangle` instead.

`id` (`\langle string \rangle`) for cross-referencing.

`ns` (`\langle URI \rangle`) the namespace to use. *Should not be used, unless you know precisely what you're doing.* If not explicitly set, is computed using `\stex_modules_current_namespace:`.

`lang` (`\langle language \rangle`) if not set, computed from the current file name (e.g. `foo.en.tex`).

`sig` (`\langle language \rangle`) if the current file is a translation of a file with the same base name but a different language suffix, setting `sig=<lang>` will preload the module from that language file. This helps ensuring that the (formal) content of both modules is (almost) identical across languages and avoids duplication.

`creators` (`\langle string \rangle*`) names of the creators.

`contributors` (`\langle string \rangle*`) names of contributors.

`srccite` (`\langle string \rangle`) a source citation for the content of this module.

---

---

`\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.

---

---

`\stexpatchmodule`    `\stexpatchmodule [\langle type \rangle] {\langle begincode \rangle} {\langle endcode \rangle}`

Customizes the presentation for those `smodule`-environments with `type=<type>`, or all others if no `\langle type \rangle` is given.

---

---

`\STEXModule`    `\STEXModule {\langle fragment \rangle}`

Attempts to find a module whose URI ends with `\langle fragment \rangle` 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.

---

---

`\stex_activate_module:n`

Activate the module with the provided URI; i.e. executes all macro code of the module's `_code`-macro (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 – all of which either declare module content or are needed in order to declare module content:

---

`\g_stex_smsmode_allowedmacros_tl`

---

Macros that are executed as is; i.e. sms mode continues immediately after. These macros may not take any arguments or otherwise gobble tokens.

Initially: `\makeatletter`, `\makeatother`, `\ExplSyntaxOn`, `\ExplSyntaxOff`.

---

`\g_stex_smsmode_allowedmacros_escape_tl`

---

Macros that are executed and potentially gobble up further tokens. These macros need to make sure, that the very last token they ultimately expand to is `\stex_smsmode_do:`.

Initially: `\symdecl`, `\notation`, `\symdef`, `\importmodule`, `\STEXexport`, `\inlineass`, `\inlinedef`, `\inlineex`, `\endinput`, `\setnotation`, `\copynotation`.

---

`\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_do:` needs to be the last token in the `\begin`-code. Since `\end`-statements take no arguments anyway, those are called directly and sms mode continues afterwards.

Initially: `smodule`, `copymodule`, `interpretmodule`, `sdefinition`, `sexample`, `sassertion`, `sparagraph`.

---

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

---

Tests whether SMS mode is currently active.

---

<code>\stex_file_in_smsmode:nn</code>	<code>\stex_in_smsmode:nn {&lt;filename&gt;} {&lt;code&gt;}</code>
---------------------------------------	--

---

Executes `<code>` in SMS mode, followed by the content of `<filename>`. `<code>` can be used e.g. to set the current repository, and is executed within a new tex group, and the same group as the file content.

---

<code>\stex_smsmode_do:</code>	Starts gobbling tokens until one is encountered that is allowed in SMS mode.
--------------------------------	--

---

### 13.1.2 Imports and Inheritance

---

<code>\importmodule</code>	<code>\importmodule[&lt;archive-ID&gt;]{&lt;module-path&gt;}</code>
----------------------------	---

---

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

---

<code>\usemodule</code>	<code>\importmodule[&lt;archive-ID&gt;]{&lt;module-path&gt;}</code>
-------------------------	---

---

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

---

<code>\stex_import_module_uri:nn</code>	<code>\stex_import_module_uri:nn {&lt;archive-ID&gt;} {&lt;module-path&gt;}</code>
---	--

---

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.

---

<code>\l_stex_import_name_str</code>	stores the result in these four variables.
<code>\l_stex_import_archive_str</code>	
<code>\l_stex_import_path_str</code>	
<code>\l_stex_import_ns_str</code>	

---



---

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

---

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

Finally, activates that module by executing its `_code`-macro.

# Chapter 14

## TEX-Symbols

Code related to symbol declarations and notations

### 14.1 Macros and Environments

---

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

---

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

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



<hr/> <hr/> <code>\stex_symdecl_do:n</code>	<p>Implements the core functionality of <code>\symdecl</code>, and is called by <code>\symdecl</code> and <code>\symdef</code>.</p> <p>Ultimately stores the symbol <math>\langle URI \rangle</math> in the property list <code>\l_stex_symdecl_⟨URI⟩_prop</code> with fields:</p> <ul style="list-style-type: none"> <li>• <code>name</code> (string),</li> <li>• <code>module</code> (string),</li> <li>• <code>notations</code> (sequence of strings; initially empty),</li> <li>• <code>local</code> (boolean),</li> <li>• <code>type</code> (token list),</li> <li>• <code>args</code> (string of <code>is</code>, <code>as</code> and <code>bs</code>),</li> <li>• <code>arity</code> (integer string),</li> <li>• <code>assoc</code>s (integer string; number of associative arguments),</li> </ul>
<hr/> <hr/> <code>\l_stex_all_symbols_seq</code>	Stores full URIs for all modules currently in scope.
<hr/> <hr/> <code>\stex_get_symbol:n</code>	Computes the full URI of a symbol from a macro argument, e.g. the macro name, the macro itself, the full URI...
<hr/> <hr/> <code>\notation</code>	<p><code>\notation[⟨args⟩]{⟨symbol⟩}{⟨notations<sup>+</sup>⟩}</code></p> <p>Introduces a new notation for <math>\langle symbol \rangle</math>, see <code>\stex_notation_do:nn</code></p>
<hr/> <hr/> <code>\stex_notation_do:nn</code>	<p><code>\stex_notation_do:nn{⟨URI⟩}{⟨notations<sup>+</sup>⟩}</code></p> <p>Implements the core functionality of <code>\notation</code>, and is called by <code>\notation</code> and <code>\symdef</code>.</p> <p>Ultimately stores the notation in the property list <code>\g_stex_notation_⟨URI⟩#⟨variant⟩#⟨lang⟩_prop</code> with fields:</p> <ul style="list-style-type: none"> <li>• <code>symbol</code> (URI string),</li> <li>• <code>language</code> (string),</li> <li>• <code>variant</code> (string),</li> <li>• <code>opprec</code> (integer string),</li> <li>• <code>argprec</code>s (sequence of integer strings)</li> </ul>
<hr/> <hr/> <code>\symdef</code>	<p><code>\symdef[⟨args⟩]{⟨symbol⟩}{⟨notations<sup>+</sup>⟩}</code></p> <p>Combines <code>\symdecl</code> and <code>\notation</code> by introducing a new symbol and assigning a new notation for it.</p>

# 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> .

<hr/> <code>\infprec</code> <hr/> <code>\neginfprec</code> <hr/>	Maximal and minimal notation precedences.
<hr/> <code>\dobrackets</code> <hr/>	<code>\dobrackets {⟨body⟩}</code>  Puts $\langle body \rangle$ in parentheses; scaled if in display mode unscaled otherwise. Uses the current $\text{\SIX}$ brackets (by default ( and )), which can be changed temporarily using <code>\withbrackets</code> .
<hr/> <code>\withbrackets</code> <hr/>	<code>\withbrackets ⟨left⟩ ⟨right⟩ {⟨body⟩}</code>  Temporarily (i.e. within $\langle body \rangle$ ) sets the brackets used by $\text{\SIX}$ 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 <code>\left</code> and <code>\right</code> in display-mode.
<hr/> <code>\stex_term_custom:nn</code> <hr/>	<code>\stex_term_custom:nn{⟨URI⟩}{⟨args⟩}</code>  Implements custom one-time notation. Invoked by <code>\stex_invoke_symbol:n</code> in text mode, or if followed by <code>*</code> in math mode, or whenever followed by <code>!</code> .
<hr/> <code>\stex_highlight_term:nn</code> <hr/>	<code>\stex_highlight_term:nn{⟨URI⟩}{⟨args⟩}</code>  Establishes a context for <code>\comp</code> . Stores the URI in a variable so that <code>\comp</code> knows which symbol governs the current notation.
<hr/> <code>\comp</code> <code>\compemph</code> <code>\compemph@uri</code> <code>\defemph</code> <code>\defemph@uri</code> <code>\symrefemph</code> <code>\symrefemph@uri</code> <hr/>	<code>\comp{⟨args⟩}</code>  Marks $\langle args \rangle$ as a notation component of the current symbol for highlighting, linking, etc.  The precise behavior is governed by <code>\@comp</code> , which takes as additional argument the URI of the current symbol. By default, <code>\@comp</code> adds the URI as a PDF tooltip and colors the highlighted part in blue.  <code>\@defemph</code> behaves like <code>\@comp</code> , and can be similarly redefined, but marks an expression as <i>definiendum</i> (used by <code>\definiendum</code> )
<hr/> <code>\STEXinvisible</code> <hr/>	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.
<hr/> <code>\ellipses</code> <hr/>	TODO

## Chapter 16

# TeX-Structural Features

Code related to structural features

### 16.1 Macros and Environments

#### 16.1.1 Structures

`mathstructure` TODO

## Chapter 17

# sTeX-Statements

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

### 17.1 Macros and Environments

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

## Chapter 18

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

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

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

## Contents

## 18.1 Introduction

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

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

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

Example 1: A very explicit proof, marked up semantically

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

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



## 18.2 The User Interface

### 18.2.1 Package Options

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

### 18.2.2 Proofs and Proof steps

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

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

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

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

### 18.2.3 Justifications

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

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

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

<b>Proof:</b>	We prove that $\sum_{i=1}^n 2i - 1 = n^2$ by induction over $n$	
<b>P.1</b>	For the induction we have to consider the following cases:	
<b>P.1.1</b>	$n = 1$ : then we compute $1 = 1^2$	□
<b>P.1.1</b>	$n = 2$ : This case is not really necessary, but we do it for the fun of it (and to get more intuition). We compute $1 + 3 = 2^2 = 4$	□
<b>P.1.1</b>	$n > 1$ :	
<b>P.1.1.1</b>	Now, we assume that the assertion is true for a certain $k \geq 1$ , i.e. $\sum_{i=1}^k (2i - 1) = k^2$ .	
<b>P.1.1.1</b>	We have to show that we can derive the assertion for $n = k + 1$ from this assumption, i.e. $\sum_{i=1}^{k+1} (2i - 1) = (k + 1)^2$ .	
<b>P.1.1.1</b>	We obtain $\sum_{i=1}^{k+1} (2i - 1) = \sum_{i=1}^k (2i - 1) + 2(k + 1) - 1$ by splitting the sum	
<b>P.1.1.1</b>	Thus we have $\sum_{i=1}^{k+1} (2i - 1) = k^2 + 2k + 1$ by inductive hypothesis.	
<b>P.1.1.1</b>	We can simplify the right-hand side to $(k + 1)^2$ , which proves the assertion.	□
<b>P.1.1</b>	We have considered all the cases, so we have proven the assertion.	□

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

#### 18.2.4 Proof Structure

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

### 18.2.5 Proof End Markers

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

$$\backslash\text{proofend}$$

$$\backslash\text{sProofEndSymbol}$$

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

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

### 18.2.6 Configuration of the Presentation

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

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

Figure 1: Configuration Hooks for Semantic Proof Markup

$$\backslash\text{pstlabelstyle}$$

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

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

Figure 2: Configuration Proof Step Label Styles

## 18.3 Limitations

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

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

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

## Chapter 19

# sTeX-Metatheory

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

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

### 19.1 Symbols

**Part III**  
**Extensions**

## Chapter 20

# Tikzinput

### 20.1 Macros and Environments

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

## Chapter 21

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

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

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

### 21.1 Introduction

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

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

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



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

## 21.2 The User Interface

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

### 21.2.1 Package and Class Options

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

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

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

### 21.2.2 Document Structure

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

```

\begin{smodule}{foo}
\symdef{bar}{B^a_r}
...
\begin{omgroup}[id=sec.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[⟨URL⟩]{⟨label⟩}`, which expands to the previously stored content. If the `\STRlabel` macro was in a different file, then we can give a URL `⟨URL⟩` that lets  $\text{\LaTeX}$ ML generate the correct reference.

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

### 21.2.5 Global Variables

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

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

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

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

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

### 21.2.6 Colors

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

## 21.3 Limitations

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

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

## Chapter 22

# NotesSlides – Slides and Course Notes

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

### 22.1 Introduction

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

### 22.2 The User Interface

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

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

#### 22.2.1 Package Options

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

- |                     |   |
|---------------------|---|
| <code>slides</code> | • 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.



---

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

2022-02-20

Write the solutions to the sheet.

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

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

[illegible]

Example 8: A generated test heading.

Part IV

# Implementation

## Chapter 25

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

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

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

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

### 25.2 Preliminaries

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

Package options:

```

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

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

```

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

## 25.3 Messages and logging

```

42 <@@=stex_log>

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

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

```

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

Redirecting messages:

```

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

```

## 25.4 HTML Annotations

```

77 <@=stex_annotate>
78 \RequirePackage{rustex}

```

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

```

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

```

Conditionals for `LATeXML`:

`\if@latexml`

```

80 \ifcsname if@latexml\endcsname\else
81   \expandafter\newif\csname if@latexml\endcsname\@latexmlfalse
82 \fi

```

(End definition for `\if@latexml`. This function is documented on page 20.)

`\latexml_if_p:`

`\latexml_if:TF`

```

83 \prg_new_conditional:Nnn \latexml_if: {p, T, F, TF} {
84   \if@latexml
85     \prg_return_true:
86   \else:
87     \prg_return_false:
88   \fi:
89 }

```

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

`\l__stex_annotate_arg_tl`

`\c__stex_annotate_emptyarg_tl`

Used by annotation macros to ensure that the HTML output to annotate is not empty.

```

90 \tl_new:N \l__stex_annotate_arg_tl
91 \tl_const:Nx \c__stex_annotate_emptyarg_tl {
92   \rustex_if:TF {
93     \rustex_direct_HTML:n { \c_ampersand_str lrm; }
94   }{-}
95 }

```

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

`\_stex_annotate_checkempty:n`

```

96 \cs_new_protected:Nn \_stex_annotate_checkempty:n {
97   \tl_set:Nn \l__stex_annotate_arg_tl { #1 }
98   \tl_if_empty:NT \l__stex_annotate_arg_tl {
99     \tl_set_eq:NN \l__stex_annotate_arg_tl \c__stex_annotate_emptyarg_tl
100   }
101 }

```

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

`\stex_if_do_html_p:`

Whether to (locally) produce HTML output

`\stex_if_do_html:TF`

```

102 \bool_new:N \_stex_html_do_output_bool
103 \bool_set_true:N \_stex_html_do_output_bool
104
105 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
106   \bool_if:nTF \_stex_html_do_output_bool
107     \prg_return_true: \prg_return_false:
108 }

```

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

`\stex_suppress_html:n`

Whether to (locally) produce HTML output

```

109 \cs_new_protected:Nn \stex_suppress_html:n {
110   \exp_args:Nne \use:nn {
111     \bool_set_false:N \_stex_html_do_output_bool
112     #1
113   }{
114     \stex_if_do_html:T {
115       \bool_set_true:N \_stex_html_do_output_bool
116     }
117   }
118 }

```

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

`\stex_annotate:nnx`

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>X<sub>M</sub>L, 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>).

`\stex_annotate_invisible:n`

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>X<sub>M</sub>L-implementations resort to perl bindings.

`\stex_annotate_invisible:nnn`

```

119 \rustex_if:TF{
120   \cs_new_protected:Nn \stex_annotate:nnn {
121     \_stex_annotate_checkempty:n { #3 }
122     \rustex_annotate_HTML:nn {
123       property="stex:#1" ~
124       resource="#2"
125     } {
126       \mode_if_vertical:TF{
127         \tl_use:N \l__stex_annotate_arg_tl\par
128       }{
129         \tl_use:N \l__stex_annotate_arg_tl
130       }
131     }
132   }
133   \cs_new_protected:Nn \stex_annotate_invisible:n {

```

```

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

```

```

188         \tl_use:N \l__stex_annotate_arg_tl
189     }
190 } {
191     \cs:w latexml@invisible@text\cs_end:{
192         \tl_use:N \l__stex_annotate_arg_tl
193     }
194 }
195 }
196 \cs_new_protected:Nn \stex_annotate_invisible:nnn {
197     \__stex_annotate_checkempty:n { #3 }
198     \cs:w latexml@annotate@invisible\cs_end:{#1}{#2}{
199         \tl_use:N \l__stex_annotate_arg_tl
200     }
201 }
202 \NewDocumentEnvironment{stex_annotate_env} { m m } {
203     \par\begin{latexml@annotateenv}{#1}{#2}
204 }{
205     \par\end{latexml@annotateenv}
206 }
207 }{
208     \cs_new_protected:Nn \stex_annotate:nnn {#3}
209     \cs_new_protected:Nn \stex_annotate_invisible:n {}
210     \cs_new_protected:Nn \stex_annotate_invisible:nnn {}
211     \NewDocumentEnvironment{stex_annotate_env} { m m } {}{}
212 }
213 }

```

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

## 25.5 Babel Languages

```

214 <@@=stex_language>

```

`\c_stex_languages_prop` We store language abbreviations in two (mutually inverse) property lists:  
`\c_stex_language_abbrevs_prop`

```

215 \prop_const_from_keyval:Nn \c_stex_languages_prop {
216     en = english ,
217     de = ngerman ,
218     ar = arabic ,
219     bg = bulgarian ,
220     ru = russian ,
221     fi = finnish ,
222     ro = romanian ,
223     tr = turkish ,
224     fr = french
225 }
226
227 \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop {
228     english = en ,
229     ngerman = de ,
230     arabic = ar ,
231     bulgarian = bg ,
232     russian = ru ,
233     finnish = fi ,

```



```

234   romanian = ro ,
235   turkish  = tr ,
236   french   = fr
237 }
238 % todo: chinese simplified (zhs)
239 %       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:

```

240 \clist_if_empty:NF \c_stex_languages_clist {
241   \clist_clear:N \l_tmpa_clist
242   \clist_map_inline:Nn \c_stex_languages_clist {
243     \prop_get:NnNTF \c_stex_languages_prop { #1 } \l_tmpa_str {
244       \clist_put_right:No \l_tmpa_clist \l_tmpa_str
245     } {
246       \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
247     }
248   }
249   \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
250   \RequirePackage[\clist_use:Nn \l_tmpa_clist,]{babel}
251 }

```

## 25.6 Auxiliary Methods

`\stex_deactivate_macro:Nn`

```

252 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
253   \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
254   \def#1{
255     \msg_error:nnnn{stex}{error/deactivated-macro}{#1}{#2}
256   }
257 }

```

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

`\stex_reactivate_macro:N`

```

258 \cs_new_protected:Nn \stex_reactivate_macro:N {
259   \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname
260 }

```

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

`\ignorespacesandpars`

```

261 \protected\def\ignorespacesandpars{
262   \begingroup\catcode13=10\relax
263   \@ifnextchar\par{
264     \endgroup\expandafter\ignorespacesandpars\@gobble
265   }{
266     \endgroup
267   }
268 }
269 \</package>

```

(End definition for `\ignorespacesandpars`. This function is documented on page 21.)

## Chapter 26

# STEX -MathHub Implementation

```
270 <*package>
271
272 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
273
274 <@@=stex_path>
275
276 Warnings and error messages
277 \msg_new:nnn{stex}{error/norepository}{
278   No~archive~#1~found~in~#2
279 }
280 \msg_new:nnn{stex}{error/notinarchive}{
281   Not~currently~in~an~archive,~but~\detokenize{#1}~
282   needs~one!
283 }
284 \msg_new:nnn{stex}{error/nofile}{
285   \detokenize{#1}~could~not~find~file~#2
286 }
287 \msg_new:nnn{stex}{error/twofiles}{
288   \detokenize{#1}~found~two~candidates~for~#2
289 }
290 }
```

### 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`

```
288 \cs_new_protected:Nn \stex_path_from_string:Nn {
289   \str_set:Nx \l_tmpa_str { #2 }
290   \str_if_empty:NTF \l_tmpa_str {
291     \seq_clear:N #1
292   }{
293     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
294     \sys_if_platform_windows:T{
295       \seq_clear:N \l_tmpa_tl
```

```

296     \seq_map_inline:Nn #1 {
297       \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
298       \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
299     }
300     \seq_set_eq:NN #1 \l_tmpa_tl
301   }
302   \stex_path_canonicalize:N #1
303 }
304 }
305

```

(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`

```

306 \cs_new_protected:Nn \stex_path_to_string:NN {
307   \exp_args:Nne \str_set:Nn #2 { \seq_use:Nn #1 / }
308 }
309
310 \cs_new:Nn \stex_path_to_string:N {
311   \seq_use:Nn #1 /
312 }

```

(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`

```

313 \str_const:Nn \c__stex_path_dot_str {.}
314 \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.

```

315 \cs_new_protected:Nn \stex_path_canonicalize:N {
316   \seq_if_empty:NF #1 {
317     \seq_clear:N \l_tmpa_seq
318     \seq_get_left:NN #1 \l_tmpa_tl
319     \str_if_empty:NT \l_tmpa_tl {
320       \seq_put_right:Nn \l_tmpa_seq {}
321     }
322     \seq_map_inline:Nn #1 {
323       \str_set:Nn \l_tmpa_tl { ##1 }
324       \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_dot_str {} {
325         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
326           \seq_if_empty:NNTF \l_tmpa_seq {
327             \exp_args:Nno \seq_put_right:Nn \l_tmpa_seq {
328               \c__stex_path_up_str
329             }
330           }{
331             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
332             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
333               \exp_args:Nno \seq_put_right:Nn \l_tmpa_seq {
334                 \c__stex_path_up_str
335               }
336             }{

```

```

337         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
338     }
339 }
340 }{
341     \str_if_empty:NF \l_tmpa_tl {
342         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
343     }
344 }
345 }
346 }
347 \seq_gset_eq:NN #1 \l_tmpa_seq
348 }
349 }

```

(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 $\underline{TF}$`

```

350 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
351     \seq_if_empty:NTF #1 {
352         \prg_return_false:
353     }{
354         \seq_get_left:NN #1 \l_tmpa_tl
355         \sys_if_platform_windows:TF{
356             \str_if_in:NnTF \l_tmpa_tl {:}{
357                 \prg_return_true:
358             }{
359                 \prg_return_false:
360             }
361         }{
362             \str_if_empty:NTF \l_tmpa_tl {
363                 \prg_return_true:
364             }{
365                 \prg_return_false:
366             }
367         }
368     }
369 }

```

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

## 26.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

370 \str_new:N\l_stex_kpsewhich_return_str
371 \cs_new_protected:Nn \stex_kpsewhich:n {
372     \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
373     \exp_args:NNo \str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
374     \tl_trim_spaces:N \l_stex_kpsewhich_return_str
375 }

```

(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`

```

376 \sys_if_platform_windows:TF{
377   \begingroup\escapechar=-1\catcode'\=12
378   \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
379   \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
380   \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_
381   }}{
382     \stex_kpsewhich:n{-var-value~PWD}
383   }
384
385   \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
386   \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
387   \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

```

388 <@@=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

```

389 \seq_gclear_new:N\g__stex_files_stack

```

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`  
`\c_stex_mainfile_str`

```

390 \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
391 \stex_path_from_string:Nn \c_stex_mainfile_seq
392   \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`

```

393 \seq_gclear_new:N\g_stex_currentfile_seq

```

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

`\stex_filestack_push:n`

```

394 \cs_new_protected:Nn \stex_filestack_push:n {
395   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
396   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
397     \stex_path_from_string:Nn\g_stex_currentfile_seq{
398       \c_stex_pwd_str/#1
399     }
400   }
401   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
402   \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
403 }

```

(End definition for `\stex_filestack_push:n`. This function is documented on page 23.)

`\stex_filestack_pop:`

```

404 \cs_new_protected:Nn \stex_filestack_pop: {
405   \seq_if_empty:NF\g__stex_files_stack{
406     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
407   }
408   \seq_if_empty:NTF\g__stex_files_stack{
409     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
410   }{
411     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
412     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
413   }
414 }

```

(End definition for `\stex_filestack_pop:`. This function is documented on page 23.)

Hooks for the current file:

```

415 \AddToHook{file/before}{
416   \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
417 }
418 \AddToHook{file/after}{
419   \stex_filestack_pop:
420 }

```

## 26.4 MathHub Repositories

421 `<@=stex_mathhub>`

`\mathhub` The path to the mathhub directory. If the `\mathhub`-macro is not set, we query `\kpsewhich` for the MATHHUB system variable.

`\c_stex_mathhub_seq`  
`\c_stex_mathhub_str`

```

422 \str_if_empty:NTF\mathhub{
423   \sys_if_platform_windows:TF{
424     \begingroup\escapechar=-1\catcode'\=12
425     \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str MATHHUB\c_percent_str}
426     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
427     \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_kpsewhich_return_str}}
428   }{
429     \stex_kpsewhich:n{-var-value-MATHHUB}
430   }
431   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
432 }
433 \str_if_empty:NTF\c_stex_mathhub_str{
434   \msg_warning:nn{stex}{warning/nomathhub}
435 }{
436   \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
437   \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
438 }
439 }{
440   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
441   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
442     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
443       \c_stex_pwd_str/\mathhub
444     }
445   }
446 }

```

```

445 }
446 \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
447 \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
448 }

```

(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` Checks whether the manifest for archive #1 already exists, and if not, finds and parses the corresponding manifest file

```

449 \cs_new_protected:Nn \_stex_mathhub_do_manifest:n {
450   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
451     \str_set:Nx \l_tmpa_str { #1 }
452     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
453     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
454     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
455     \_stex_mathhub_find_manifest:N \l_tmpa_seq
456     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
457       \msg_error:nnxx{stex}{error/norepository}{#1}{
458         \stex_path_to_string:N \c_stex_mathhub_str
459       }
460     } {
461       \exp_args:No \_stex_mathhub_parse_manifest:n { \l_tmpa_str }
462     }
463   }
464 }

```

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

`\l__stex_mathhub_manifest_file_seq`

```

465 \seq_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`:

```

466 \cs_new_protected:Nn \_stex_mathhub_find_manifest:N {
467   \seq_set_eq:NN\l_tmpa_seq #1
468   \bool_set_true:N\l_tmpa_bool
469   \bool_while_do:Nn \l_tmpa_bool {
470     \seq_if_empty:NTF \l_tmpa_seq {
471       \bool_set_false:N\l_tmpa_bool
472     } {
473       \file_if_exist:nTF{
474         \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
475       } {
476         \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
477         \bool_set_false:N\l_tmpa_bool
478       } {
479         \file_if_exist:nTF{
480           \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
481         } {
482           \seq_put_right:Nn\l_tmpa_seq{META-INF}
483           \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}

```

```

484         \bool_set_false:N\l_tmpa_bool
485     }{
486         \file_if_exist:nTF{
487             \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
488         }{
489             \seq_put_right:Nn\l_tmpa_seq{meta-inf}
490             \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
491             \bool_set_false:N\l_tmpa_bool
492         }{
493             \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
494         }
495     }
496 }
497 }
498 }
499 \seq_set_eq:NN\l__stex_mathhub_manifest_file_seq\l_tmpa_seq
500 }

```

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

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

```
501 \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:

```

502 \cs_new_protected:Nn \__stex_mathhub_parse_manifest:n {
503     \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
504     \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
505     \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
506         \str_set:Nn \l_tmpa_str {##1}
507         \exp_args:NNoo \seq_set_split:Nnn
508             \l_tmpb_seq \c_colon_str \l_tmpa_str
509         \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
510             \exp_args:NNe \str_set:Nn \l_tmpb_tl {
511                 \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
512             }
513             \exp_args:No \str_case:nnTF \l_tmpa_tl {
514                 {id} {
515                     \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
516                     { id } \l_tmpb_tl
517                 }
518                 {narration-base} {
519                     \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
520                     { narr } \l_tmpb_tl
521                 }
522                 {url-base} {
523                     \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
524                     { docurl } \l_tmpb_tl
525                 }
526                 {source-base} {
527                     \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
528                     { ns } \l_tmpb_tl
529                 }

```



```

530     {ns} {
531         \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
532         { ns } \l_tmpb_tl
533     }
534     {dependencies} {
535         \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
536         { deps } \l_tmpb_tl
537     }
538     }{}{}
539 }{}
540 }
541 \ior_close:N \c__stex_mathhub_manifest_ior
542 }

```

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

`\stex_set_current_repository:n`

```

543 \cs_new_protected:Nn \stex_set_current_repository:n {
544     \stex_require_repository:n { #1 }
545     \prop_set_eq:Nc \l_stex_current_repository_prop {
546         c_stex_mathhub_#1_manifest_prop
547     }
548 }

```

(End definition for `\stex_set_current_repository:n`. This function is documented on page 23.)

`\stex_require_repository:n`

```

549 \cs_new_protected:Nn \stex_require_repository:n {
550     \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
551         \stex_debug:nn{mathhub}{Opening~archive:~#1}
552         \_stex_mathhub_do_manifest:n { #1 }
553     }
554 }

```

(End definition for `\stex_require_repository:n`. This function is documented on page 23.)

`\l_stex_current_repository_prop`

Current MathHub repository

```

555 %\prop_new:N \l_stex_current_repository_prop
556
557 \_stex_mathhub_find_manifest:N \c_stex_pwd_seq
558 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
559     \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
560 } {
561     \_stex_mathhub_parse_manifest:n { main }
562     \prop_get:Nn \c_stex_mathhub_main_manifest_prop {id}
563     \l_tmpa_str
564     \prop_set_eq:cN { c_stex_mathhub\_l_tmpa_str_manifest_prop }
565     \c_stex_mathhub_main_manifest_prop
566     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
567     \stex_debug:nn{mathhub}{Current~repository:~
568         \prop_item:Nn \l_stex_current_repository_prop {id}
569     }
570 }

```

(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.

```

571 \cs_new_protected:Nn \stex_in_repository:nn {
572   \str_set:Nx \l_tmpa_str { #1 }
573   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
574   \str_if_empty:NTF \l_tmpa_str {
575     \prop_if_exist:NTF \l_stex_current_repository_prop {
576       \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
577       \exp_args:Ne \l_tmpa_cs{
578         \prop_item:Nn \l_stex_current_repository_prop { id }
579       }
580     }{
581       \l_tmpa_cs{}
582     }
583   }{
584     \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}
585     \stex_require_repository:n \l_tmpa_str
586     \str_set:Nx \l_tmpa_str { #1 }
587     \exp_args:Nne \use:nn {
588       \stex_set_current_repository:n \l_tmpa_str
589       \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
590     }{
591       \stex_debug:nn{mathhub}{switching~back~to:~
592       \prop_if_exist:NTF \l_stex_current_repository_prop {
593         \prop_item:Nn \l_stex_current_repository_prop { id }::~
594       \meaning\l_stex_current_repository_prop
595       }{
596         no~repository
597       }
598     }
599     \prop_if_exist:NTF \l_stex_current_repository_prop {
600       \stex_set_current_repository:n {
601         \prop_item:Nn \l_stex_current_repository_prop { id }
602       }
603     }{
604       \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
605     }
606   }
607 }
608 }

```

(End definition for `\stex_in_repository:nn`. This function is documented on page [23](#).)

## 26.5 Using Content in Archives

`\mhpath`

```

609 \def \mhpath #1 #2 {
610   \exp_args:Ne \str_if_eq:nnTF{#1}{}{
611     \c_stex_mathhub_str /
612     \prop_item:Nn \l_stex_current_repository_prop { id }
613     / source / #2
614   }{
615     \c_stex_mathhub_str / #1 / source / #2

```

```

616 }
617 }

```

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

`\inputref`  
`\mhinput`

```

618 \newif \ifinputref \inputreffalse
619
620 \cs_new_protected:Nn \__stex_mathhub_mhinput:nn {
621   \stex_in_repository:nn {#1} {
622     \ifinputref
623       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
624     \else
625       \inputreftrue
626       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
627       \inputreffalse
628     \fi
629   }
630 }
631 \NewDocumentCommand \mhinput { 0{} m }{
632   \stex_mhinput:nn{ #1 }{ #2 }
633 }
634
635 \cs_new_protected:Nn \__stex_mathhub_inputref:nn {
636   \stex_in_repository:nn {#1} {
637     \bool_lazy_any:nTF {
638       {\rustex_if_p:}
639       {\latexml_if_p:}
640     } {
641       \str_clear:N \l_tmpa_str
642       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
643         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
644       }
645       \stex_annotate_invisible:nnn{inputref}{
646         \l_tmpa_str / #2
647       }{}
648     }{
649       \begingroup
650         \inputreftrue
651         \input{ \c_stex_mathhub_str / ##1 / source / #2 }
652       \endgroup
653     }
654   }
655 }
656 \NewDocumentCommand \inputref { 0{} m }{
657   \__stex_mathhub_inputref:nn{ #1 }{ #2 }
658 }

```

(End definition for `\inputref` and `\mhinput`. These functions are documented on page 24.)

`\addmhbibresource`

```

659 \cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {
660   \stex_in_repository:nn {#1} {
661     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
662   }

```

```

663 }
664 \newcommand\addmhbibresource[2][]{
665   \_stex_mathhub_mhbibresource:nn{ #1 }{ #2 }
666 }

```

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

### \libinput

```

667 \cs_new_protected:Npn \libinput #1 {
668   \prop_if_exist:NF \l_stex_current_repository_prop {
669     \msg_error:nnn{stex}{error/notinarchive}\libinput
670   }
671   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
672     \msg_error:nnn{stex}{error/notinarchive}\libinput
673   }
674   \seq_clear:N \l__stex_mathhub_libinput_files_seq
675   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
676   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
677
678   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
679     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}
680     \IfFileExists{ \l_tmpa_str }{
681       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
682     }{}
683     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
684     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
685   }
686
687   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
688   \IfFileExists{ \l_tmpa_str }{
689     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
690   }{}
691
692   \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
693     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
694   }{
695     \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
696       \input{ ##1 }
697     }
698   }
699 }

```

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

### \libusepackage

```

700 \NewDocumentCommand \libusepackage {0{} m} {
701   \prop_if_exist:NF \l_stex_current_repository_prop {
702     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
703   }
704   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
705     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
706   }
707   \tl_clear:N \l__stex_mathhub_libinput_files_seq
708   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
709   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str

```

```

710
711 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
712   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2.sty}
713   \IfFileExists{ \l_tmpa_str }{
714     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
715   }{}
716   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
717   \seq_put_right:No \l_tmpa_seq \l_tmpa_str
718 }
719
720 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2.sty}
721 \IfFileExists{ \l_tmpa_str }{
722   \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
723 }{}
724
725 \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
726   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
727 }{
728   \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
729     \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
730       \usepackage[#1]{ #1 }
731     }
732   }{
733     \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
734   }
735 }
736 }

```

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

`\mhgraphics`  
`\cmhgraphics`

```

737
738 \AddToHook{begindocument}{
739   \ltx@ifpackageloaded{graphicx}{
740     \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
741     \newcommand\mhgraphics[2][]{\%
742       \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
743       \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}}
744     \newcommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
745   }{}

```

(End definition for `\mhgraphics` and `\cmhgraphics`. These functions are documented on page 24.)

`\lstinputmhlisting`  
`\clstinputmhlisting`

```

746 \ltx@ifpackageloaded{listings}{
747   \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
748   \newcommand\lstinputmhlisting[2][]{\%
749     \def\lst@mhrepos{}\setkeys{lst}{#1}%
750     \lstinputlisting[#1]{\mhp\lst@mhrepos{#2}}}
751   \newcommand\clstinputmhlisting[2][]{\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
752 }{}
753 }
754
755 </package>

```

*(End definition for \lstinputmhlisting and \clstinputmhlisting. These functions are documented on page 24.)*

## Chapter 27

# STEX -References Implementation

```
756 <*package>
757
758 %%%%%%%%%%% references.dtx %%%%%%%%%%%
759
760 <@@=stex_refs>
```

Warnings and error messages

```
761
```

References are stored in the file `\jobname.sref`, to enable cross-referencing external documents.

```
762 \iow_new:N \c__stex_refs_refs_iow
763 \AddToHook{begindocument}{
764   \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
765 }
766 \AddToHook{enddocument}{
767   \iow_close:N \c__stex_refs_refs_iow
768 }
```

`\STEXreftitle`

```
769 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
770
771 \NewDocumentCommand \STEXreftitle { m } {
772   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
773 }
```

*(End definition for `\STEXreftitle`. This function is documented on page 25.)*

### 27.1 Document URIs and URLs

`\l_stex_current_docns_str`

```
774 \str_new:N \l_stex_current_docns_str
```

*(End definition for `\l_stex_current_docns_str`. This variable is documented on page 25.)*

`\stex_get_document_uri:`

```
775 \cs_new_protected:Nn \stex_get_document_uri: {  
776   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq  
777   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str  
778   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str  
779   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str  
780   \seq_put_right:No \l_tmpa_seq \l_tmpb_str  
781  
782   \str_clear:N \l_tmpa_str  
783   \prop_if_exist:NT \l_stex_current_repository_prop {  
784     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {  
785       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}  
786     }  
787   }  
788  
789   \str_if_empty:NTF \l_tmpa_str {  
790     \str_set:Nx \l_stex_current_docns_str {  
791       file:/\stex_path_to_string:N \l_tmpa_seq  
792     }  
793   }{  
794     \bool_set_true:N \l_tmpa_bool  
795     \bool_while_do:Nn \l_tmpa_bool {  
796       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str  
797       \exp_args:No \str_case:nnTF { \l_tmpb_str } {  
798         {source} { \bool_set_false:N \l_tmpa_bool }  
799       }{}{  
800         \seq_if_empty:NT \l_tmpa_seq {  
801           \bool_set_false:N \l_tmpa_bool  
802         }  
803       }  
804     }  
805  
806     \seq_if_empty:NTF \l_tmpa_seq {  
807       \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str  
808     }{  
809       \str_set:Nx \l_stex_current_docns_str {  
810         \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq  
811       }  
812     }  
813   }  
814 }
```

(End definition for `\stex_get_document_uri:`. This function is documented on page 25.)

`\l_stex_current_docurl_str`

```
815 \str_new:N \l_stex_current_docurl_str
```

(End definition for `\l_stex_current_docurl_str`. This variable is documented on page 25.)

`\stex_get_document_url:`

```
816 \cs_new_protected:Nn \stex_get_document_url: {  
817   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq  
818   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str  
819   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
```



```

820 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
821 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
822
823 \str_clear:N \l_tmpa_str
824 \prop_if_exist:NT \l_stex_current_repository_prop {
825   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
826     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
827       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
828     }
829   }
830 }
831
832 \str_if_empty:NTF \l_tmpa_str {
833   \str_set:Nx \l_stex_current_docurl_str {
834     file:/\stex_path_to_string:N \l_tmpa_seq
835   }
836 }{
837   \bool_set_true:N \l_tmpa_bool
838   \bool_while_do:Nn \l_tmpa_bool {
839     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
840     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
841       {source} { \bool_set_false:N \l_tmpa_bool }
842     }{}{
843       \seq_if_empty:NT \l_tmpa_seq {
844         \bool_set_false:N \l_tmpa_bool
845       }
846     }
847   }
848
849   \seq_if_empty:NTF \l_tmpa_seq {
850     \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
851   }{
852     \str_set:Nx \l_stex_current_docurl_str {
853       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
854     }
855   }
856 }
857 }

```

(End definition for `\stex_get_document_url`:. This function is documented on page 25.)

## 27.2 Setting Reference Targets

```

858 \str_const:Nn \c__stex_refs_url_str{URL}
859 \str_const:Nn \c__stex_refs_ref_str{REF}
860 \str_new:N \l__stex_refs_curr_label_str
861 % @currentlabel -> number
862 % @currentlabelname -> title
863 % @currentHref -> name.number <- id of some kind
864 % \theH# -> \arabic{section}
865 % \the# -> number
866 % \hyper@makecurrent{#}
867 \int_new:N \l__stex_refs_unnamed_counter_int

```

`\stex_ref_new_doc_target:n`

```

868 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
869   \stex_get_document_uri:
870   \str_clear:N \l__stex_refs_curr_label_str
871   \str_set:Nx \l_tmpa_str { #1 }
872   \str_if_empty:NT \l_tmpa_str {
873     \int_incr:N \l__stex_refs_unnamed_counter_int
874     \str_set:Nx \l_tmpa_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
875   }
876   \str_set:Nx \l__stex_refs_curr_label_str {
877     \l_stex_current_docns_str?\l_tmpa_str
878   }
879   \seq_if_exist:cF{g__stex_refs_labels_\l_tmpa_str_seq}{
880     \seq_new:c {g__stex_refs_labels_\l_tmpa_str_seq}
881   }
882   \seq_if_in:coF{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str {
883     \seq_gput_right:co{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str
884   }
885   \stex_if_smsmode:TF {
886     \stex_get_document_url:
887     \str_gset_eq:cN {sref_url_\l__stex_refs_curr_label_str_str}\l_stex_current_docurl_str
888     \str_gset_eq:cN {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_url_str
889   }{
890     \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~\expandafter\unexpanded\expandafter{
891       \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
892       \immediate\write\@auxout{\stexauxadddocref{\l_stex_current_docns_str}{\l_tmpa_str}}
893       \str_gset:cx {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_ref_str
894     }
895   }

```

(End definition for `\stex_ref_new_doc_target:n`. This function is documented on page 25.)

The following is used to set the necessary macros in the .aux-file.

```

896 \cs_new_protected:Npn \stexauxadddocref #1 #2 {
897   \str_set:Nn \l_tmpa_str {#1?#2}
898   \str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
899   \seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
900     \seq_new:c {g__stex_refs_labels_#2_seq}
901   }
902   \seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
903     \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
904   }
905 }

```

To avoid resetting the same macros when the .aux-file is read at the end of the document:

```

906 \AtEndDocument{
907   \def\stexauxadddocref#1 #2 {}{}
908 }

```

`\stex_ref_new_sym_target:n`

```

909 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
910   \stex_if_smsmode:TF {
911     \str_if_exist:cF{sref_sym_#1_type}{
912       \stex_get_document_url:
913       \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str

```

```

914     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
915   }
916 }{
917   \str_if_empty:NF \l__stex_refs_curr_label_str {
918     \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
919     \immediate\write\@auxout{
920       \exp_not:N\expandafter\def\exp_not:N\csname sref_sym_#1_label_str\exp_not:N\endcsname
921       \l__stex_refs_curr_label_str
922     }
923   }
924 }
925 }
926 }

```

(End definition for `\stex_ref_new_sym_target:n`. This function is documented on page 25.)

## 27.3 Using References

```

927 \str_new:N \l__stex_refs_indocument_str

```

**\sref** Optional arguments:

```

928
929 \keys_define:nn { stex / sref } {
930   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
931   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
932   pre           .tl_set:N = \l__stex_refs_pre_tl ,
933   post          .tl_set:N = \l__stex_refs_post_tl ,
934 }
935 \cs_new_protected:Nn \__stex_refs_args:n {
936   \tl_clear:N \l__stex_refs_linktext_tl
937   \tl_clear:N \l__stex_refs_fallback_tl
938   \tl_clear:N \l__stex_refs_pre_tl
939   \tl_clear:N \l__stex_refs_post_tl
940   \str_clear:N \l__stex_refs_repo_str
941   \keys_set:nn { stex / sref } { #1 }
942 }

```

The actual macro:

```

943 \NewDocumentCommand \sref { 0{} m}{
944   \__stex_refs_args:n { #1 }
945   \str_if_empty:NTF \l__stex_refs_indocument_str {
946     \str_set:Nx \l_tmpa_str { #2 }
947     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
948     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
949       \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
950         \seq_get_left:cNF {g__stex_refs_labels_\l_tmpa_str _seq} \l_tmpa_str {
951           \str_clear:N \l_tmpa_str
952         }
953       }{
954         \str_clear:N \l_tmpa_str
955       }
956     }{
957       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
958       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str

```

```

959 \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
960 \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str_seq}{
961   \str_set_eq:NN \l_tmpc_str \l_tmpa_str
962   \str_clear:N \l_tmpa_str
963   \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str_seq} {
964     \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
965       \str_range:nnn { ##1 }{ -\l_tmpa_int}{ -1 }
966     }{
967       \seq_map_break:n {
968         \str_set:Nn \l_tmpa_str { ##1 }
969       }
970     }
971   }
972 }{
973   \str_clear:N \l_tmpa_str
974 }
975 }
976 \str_if_empty:NTF \l_tmpa_str {
977   \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_lin
978 }{
979   \str_if_eq:cNTF {sref_\l_tmpa_str_type} \c__stex_refs_ref_str {
980     \tl_if_empty:NTF \l__stex_refs_linktext_tl {
981       \cs_if_exist:cTF{autoref}{
982         \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
983       }{
984         \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
985       }
986     }{
987       \ltx@ifpackageloaded{hyperref}{
988         \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
989       }{
990         \l__stex_refs_linktext_tl
991       }
992     }
993   }{
994     \ltx@ifpackageloaded{hyperref}{
995       \href{\use:c{sref_url_\l_tmpa_str_str}}{\tl_if_empty:NTF \l__stex_refs_linktext_t
996     }{
997       \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs
998     }
999   }
1000 }
1001 }{
1002   % TODO
1003 }
1004 }

```

(End definition for \sref. This function is documented on page 26.)

### \srefsym

```

1005 \NewDocumentCommand \srefsym { 0{} m }{
1006   \stex_get_symbol:n { #2 }
1007   \__stex_refs_sym_aux:nn{##1}{\l_stex_get_symbol_uri_str}
1008 }

```

```

1009
1010 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1011   \str_if_exist:cTF {sref_sym_#2 _label_str }{
1012     \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1013   }{
1014     \__stex_refs_args:n { #1 }
1015     \str_if_empty:NTF \l__stex_refs_indocument_str {
1016       \tl_if_exist:cTF{sref_sym_#2 _type}{
1017         % doc uri in \l_tmpb_str
1018         \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1019         \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1020           % reference
1021           \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1022             \cs_if_exist:cTF{autoref}{
1023               \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1024             }{
1025               \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1026             }
1027           }{
1028             \ltx@ifpackageloaded{hyperref}{
1029               \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1030             }{
1031               \l__stex_refs_linktext_tl
1032             }
1033           }
1034         }{
1035           % URL
1036           \ltx@ifpackageloaded{hyperref}{
1037             \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl \
1038           }{
1039             \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_re
1040           }
1041         }
1042       }{
1043         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_l
1044       }
1045     }{
1046       % TODO
1047     }
1048   }
1049 }

```

(End definition for \srefsym. This function is documented on page 26.)

**\srefsymuri**

```

1050 \cs_new_protected:Npn \srefsymuri #1 #2 {
1051   \__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1052 }

```

(End definition for \srefsymuri. This function is documented on page 26.)

```

1053 </package>

```

## Chapter 28

# STEX -Modules Implementation

```
1054 <*package>
1055
1056 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1057
1058 <@@=stex_modules>
1059
1060 Warnings and error messages
1061 \msg_new:nnn{stex}{error/unknownmodule}{
1062   No~module~#1~found
1063 }
1064 \msg_new:nnn{stex}{error/syntax}{
1065   Syntax~error:~#1
1066 }
1067 \msg_new:nnn{stex}{error/siglanguage}{
1068   Module~#1~declares~signature~#2,~but~does~not~
1069   declare~its~language
1070 }
1071 \msg_new:nnn{stex}{warning/deprecated}{
1072   #1~is~deprecated;~please~use~#2~instead!
1073 }
1074 \msg_new:nnn{stex}{error/conflictingmodules}{
1075   Conflicting~imports~for~module~#1
1076 }
1077
\l_stex_current_module_str The current module:
1078 \str_new:N \l_stex_current_module_str
1079
(End definition for \l_stex_current_module_str. This variable is documented on page 28.)
1080
\l_stex_all_modules_seq Stores all available modules
1081 \seq_new:N \l_stex_all_modules_seq
1082
(End definition for \l_stex_all_modules_seq. This variable is documented on page 28.)
```

```

\stex_if_in_module_p:
\stex_if_in_module:TF
1078 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1079   \str_if_empty:NTF \l_stex_current_module_str
1080   \prg_return_false: \prg_return_true:
1081 }

```

(End definition for \stex\_if\_in\_module:TF. This function is documented on page 28.)

```

\stex_if_module_exists_p:n
\stex_if_module_exists:nTF
1082 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1083   \prop_if_exist:cTF { c_stex_module_#1_prop }
1084   \prg_return_true: \prg_return_false:
1085 }

```

(End definition for \stex\_if\_module\_exists:nTF. This function is documented on page 28.)

\stex\_add\_to\_current\_module:n Only allowed within modules:

```

\STEXexport
1086 \cs_new_protected:Nn \stex_add_to_current_module:n {
1087   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1088 }
1089 \cs_new_protected:Npn \STEXexport {
1090   \begingroup
1091   \newlinechar=-1\relax
1092   \endlinechar=-1\relax
1093   %\catcode'\ = 9\relax
1094   \expandafter\endgroup\__stex_modules_export:n
1095 }
1096 \cs_new_protected:Nn \__stex_modules_export:n {
1097   \ignorespaces #1
1098   \stex_add_to_current_module:n { \ignorespaces #1 }
1099   \stex_smsmode_do:
1100 }
1101 \stex_deactivate_macro:Nn \STEXexport {module~environments}

```

(End definition for \stex\_add\_to\_current\_module:n and \STEXexport. These functions are documented on page 28.)

```

\stex_add_constant_to_current_module:n
1102 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1103   \str_set:Nx \l_tmpa_str { #1 }
1104   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1105 }

```

(End definition for \stex\_add\_constant\_to\_current\_module:n. This function is documented on page 28.)

```

\stex_add_import_to_current_module:n
1106 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1107   \str_set:Nx \l_tmpa_str { #1 }
1108   \exp_args:Nno
1109   \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1110     \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1111   }
1112 }

```

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

`\stex_collect_imports:n`

```

1113 \cs_new_protected:Nn \stex_collect_imports:n {
1114   \seq_clear:N \l_stex_collect_imports_seq
1115   \__stex_modules_collect_imports:n {#1}
1116 }
1117 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1118   \seq_map_inline:cn {c_stex_module_#1_imports} {
1119     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1120       \__stex_modules_collect_imports:n { ##1 }
1121     }
1122   }
1123   \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1124     \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1125   }
1126 }

```

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

`\stex_do_up_to_module:n`

```

1127 \int_new:N \l__stex_modules_group_depth_int
1128 \tl_new:N \l__stex_modules_aftergroup_tl
1129 \cs_new_protected:Nn \stex_do_up_to_module:n {
1130   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1131     #1
1132   }{
1133     #1
1134     \expandafter \tl_gset:Nn \expandafter \l__stex_modules_aftergroup_tl \expandafter { \l__
1135       \aftergroup\__stex_modules_aftergroup_do:
1136     }
1137   }
1138   \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1139     \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1140       \l__stex_modules_aftergroup_tl
1141       \tl_clear:N \l__stex_modules_aftergroup_tl
1142     }{
1143       \l__stex_modules_aftergroup_tl
1144       \aftergroup\__stex_modules_aftergroup_do:
1145     }
1146   }

```

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

`\stex_modules_compute_namespace:nN` Computes the appropriate namespace from the top-level namespace of a repository (#1) and a file path (#2).

1147

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

`\stex_modules_current_namespace:` Computes the current namespace based on the current MathHub repository (if existent) and the current file.

```

1148 \str_new:N \l_stex_modules_ns_str
1149 \str_new:N \l_stex_modules_subpath_str

```



```

1150 \cs_new_protected:Nn \__stex_modules_compute_namespace:nN {
1151   \str_set:Nx \l_tmpa_str { #1 }
1152   \seq_set_eq:NN \l_tmpa_seq #2
1153   % split off file extension
1154   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1155   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1156   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1157   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1158
1159   \bool_set_true:N \l_tmpa_bool
1160   \bool_while_do:Nn \l_tmpa_bool {
1161     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1162     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1163       {source} { \bool_set_false:N \l_tmpa_bool }
1164     }{}{
1165       \seq_if_empty:NT \l_tmpa_seq {
1166         \bool_set_false:N \l_tmpa_bool
1167       }
1168     }
1169   }
1170
1171   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
1172   \str_if_empty:NTF \l_stex_modules_subpath_str {
1173     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
1174   }{
1175     \str_set:Nx \l_stex_modules_ns_str {
1176       \l_tmpa_str/\l_stex_modules_subpath_str
1177     }
1178   }
1179 }
1180
1181 \cs_new_protected:Nn \stex_modules_current_namespace: {
1182   \str_clear:N \l_stex_modules_subpath_str
1183   \prop_if_exist:NTF \l_stex_current_repository_prop {
1184     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1185     \__stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1186   }{
1187     % split off file extension
1188     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1189     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1190     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1191     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1192     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1193     \str_set:Nx \l_stex_modules_ns_str {
1194       file:\stex_path_to_string:N \l_tmpa_seq
1195     }
1196   }
1197 }

```

(End definition for `\stex_modules_current_namespace:.` This function is documented on page 29.)

## 28.1 The smodule environment

smodule arguments:

```

1198 \keys_define:nn { stex / module } {
1199   title      .tl_set:N      = \smodulename ,
1200   type       .str_set_x:N   = \smoduletype ,
1201   id         .str_set_x:N   = \smoduleid ,
1202   deprecate  .str_set_x:N   = \l_stex_module_deprecate_str ,
1203   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1204   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1205   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1206   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1207   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1208   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1209   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1210 }
1211
1212 \cs_new_protected:Nn \__stex_modules_args:n {
1213   \str_clear:N \smodulename
1214   \str_clear:N \smoduletype
1215   \str_clear:N \smoduleid
1216   \str_clear:N \l_stex_module_ns_str
1217   \str_clear:N \l_stex_module_deprecate_str
1218   \str_clear:N \l_stex_module_lang_str
1219   \str_clear:N \l_stex_module_sig_str
1220   \str_clear:N \l_stex_module_creators_str
1221   \str_clear:N \l_stex_module_contributors_str
1222   \str_clear:N \l_stex_module_meta_str
1223   \str_clear:N \l_stex_module_srccite_str
1224   \keys_set:nn { stex / module } { #1 }
1225 }
1226
1227 % module parameters here? In the body?
1228

```

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

```

1229 \cs_new_protected:Nn \stex_module_setup:nn {
1230   \str_set:Nx \l_stex_module_name_str { #2 }
1231   \__stex_modules_args:n { #1 }
1232
1233   First, we set up the name and namespace of the module.
1234   Are we in a nested module?
1235
1236   \stex_if_in_module:TF {
1237     % Nested module
1238     \prop_get:cnN {c_stex_module_\l_stex_current_module_str _prop}
1239     { ns } \l_stex_module_ns_str
1240     \str_set:Nx \l_stex_module_name_str {
1241       \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
1242       { name } / \l_stex_module_name_str
1243     }
1244   }{
1245     % not nested:
1246     \str_if_empty:NT \l_stex_module_ns_str {
1247       \stex_modules_current_namespace:
1248     }
1249   }
1250 }

```

```

1244 \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1245 \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1246 / {\l_stex_module_ns_str}
1247 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1248 \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1249 \str_set:Nx \l_stex_module_ns_str {
1250 \stex_path_to_string:N \l_tmpa_seq
1251 }
1252 }
1253 }
1254 }

```

Next, we determine the language of the module:

```

1255 \str_if_empty:NT \l_stex_module_lang_str {
1256 \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1257 \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1258 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1259 \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1260 \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1261 \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1262 inferred~from~file~name}
1263 \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1264 }
1265 }
1266
1267 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
1268 \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1269 \l_tmpa_str {
1270 \ltx@ifpackageloaded{babel}{
1271 \exp_args:Nx \selectlanguage { \l_tmpa_str }
1272 }{}
1273 } {
1274 \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
1275 }
1276 }}

```

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

```

1277 \str_if_empty:NTF \l_stex_module_sig_str {
1278 \exp_args:Nnx \prop_gset_from_keyval:cn {
1279 c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1280 } {
1281 name = \l_stex_module_name_str ,
1282 ns = \l_stex_module_ns_str ,
1283 file = \exp_not:o { \g_stex_currentfile_seq } ,
1284 lang = \l_stex_module_lang_str ,
1285 sig = \l_stex_module_sig_str ,
1286 deprecate = \l_stex_module_deprecate_str ,
1287 meta = \l_stex_module_meta_str
1288 }
1289 \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1290 \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1291 \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1292 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1293 \str_if_empty:NT \l_stex_module_meta_str {
1294   \str_set:Nx \l_stex_module_meta_str {
1295     \c_stex_metatheory_ns_str ? Metatheory
1296   }
1297 }
1298 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1299   \bool_set_true:N \l_stex_in_meta_bool
1300   \exp_args:Nx \stex_add_to_current_module:n {
1301     \bool_set_true:N \l_stex_in_meta_bool
1302     \stex_activate_module:n {\l_stex_module_meta_str}
1303     \bool_set_false:N \l_stex_in_meta_bool
1304   }
1305   \stex_activate_module:n {\l_stex_module_meta_str}
1306   \bool_set_false:N \l_stex_in_meta_bool
1307 }
1308 }{
1309   \str_if_empty:NT \l_stex_module_lang_str {
1310     \msg_error:nnxx{stex}{error/siglanguage}{
1311       \l_stex_module_ns_str?\l_stex_module_name_str
1312     }{\l_stex_module_sig_str}
1313   }
1314
1315   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1316   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1317   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1318   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1319   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1320   \str_set:Nx \l_tmpa_str {
1321     \stex_path_to_string:N \l_tmpa_seq /
1322     \l_tmpa_str . \l_stex_module_sig_str .tex
1323   }
1324   \IfFileExists \l_tmpa_str {
1325     \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1326       \str_clear:N \l_stex_current_module_str
1327       \seq_clear:N \l_stex_all_modules_seq
1328       \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1329     }
1330   }{
1331     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1332   }
1333   \stex_if_smsmode:F {
1334     \stex_activate_module:n {
1335       \l_stex_module_ns_str ? \l_stex_module_name_str
1336     }
1337   }
1338   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1339 }
1340 \str_if_empty:NF \l_stex_module_deprecate_str {
1341   \msg_warning:nnxx{stex}{warning/deprecated}{
1342     Module~\l_stex_current_module_str
1343   }{
1344     \l_stex_module_deprecate_str
1345   }

```

```

1346 }
1347 }

```

(End definition for `\stex_module_setup:nn`. This function is documented on page 29.)

**smodule** The module environment.

`\_stex_modules_begin_module:` implements `\begin{smodule}`

```

1348 \cs_new_protected:Nn \_stex_modules_begin_module: {
1349   \stex_reactivate_macro:N \STEXexport
1350   \stex_reactivate_macro:N \importmodule
1351   \stex_reactivate_macro:N \symdecl
1352   \stex_reactivate_macro:N \notation
1353   \stex_reactivate_macro:N \symdef
1354
1355   \stex_debug:nn{modules}{
1356     New~module:\\
1357     Namespace:~\l_stex_module_ns_str\\
1358     Name:~\l_stex_module_name_str\\
1359     Language:~\l_stex_module_lang_str\\
1360     Signature:~\l_stex_module_sig_str\\
1361     Metatheory:~\l_stex_module_meta_str\\
1362     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1363   }
1364
1365   \seq_put_right:Nx \l_stex_all_modules_seq {
1366     \l_stex_module_ns_str ? \l_stex_module_name_str
1367   }
1368
1369   \stex_if_smsmode:F{
1370     \begin{stex_annotate_env} {theory} {
1371       \l_stex_module_ns_str ? \l_stex_module_name_str
1372     }
1373
1374     \stex_annotate_invisible:nnn{header}{} {
1375       \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1376       \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1377       \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1378         \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1379       }
1380       \str_if_empty:NF \smoduletype {
1381         \stex_annotate:nnn{type}{\smoduletype}{}
1382       }
1383     }
1384   }
1385   \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1386   % TODO: Inherit metatheory for nested modules?
1387 }
1388 \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}`

```

1389 \cs_new_protected:Nn \_stex_modules_end_module: {

```

```

1390 \stex_debug:nn{modules}{Closing-module~\prop_item:cn {c_stex_module_\l_stex_current_module
1391 }

```

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

The core environment

```

1392 \iffalse \begin{stex_annotate_env} \fi %%^A make syntax highlighting work again
1393 \NewDocumentEnvironment { smodule } { 0{} m } {
1394   \stex_module_setup:nn{#1}{#2}
1395   \par
1396   \stex_if_smsmode:F{
1397     \tl_clear:N \l_tmpa_tl
1398     \clist_map_inline:Nn \smodulotype {
1399       \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1400         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1401       }
1402     }
1403     \tl_if_empty:NTF \l_tmpa_tl {
1404       \__stex_modules_smodule_start:
1405     }{
1406       \l_tmpa_tl
1407     }
1408   }
1409   \__stex_modules_begin_module:
1410   \str_if_empty:NF \smoduleid {
1411     \stex_ref_new_doc_target:n \smoduleid
1412   }
1413   \stex_smsmode_do:
1414 } {
1415   \__stex_modules_end_module:
1416   \stex_if_smsmode:F {
1417     \end{stex_annotate_env}
1418     \clist_set:Nn \l_tmpa_clist \smodulotype
1419     \tl_clear:N \l_tmpa_tl
1420     \clist_map_inline:Nn \l_tmpa_clist {
1421       \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1422         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1423       }
1424     }
1425     \tl_if_empty:NTF \l_tmpa_tl {
1426       \__stex_modules_smodule_end:
1427     }{
1428       \l_tmpa_tl
1429     }
1430   }
1431 }

```

**\stexpatchmodule**

```

1432 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1433 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1434
1435 \newcommand\stexpatchmodule[3] [] {
1436   \str_set:Nx \l_tmpa_str{ #1 }
1437   \str_if_empty:NTF \l_tmpa_str {
1438     \tl_set:Nn \__stex_modules_smodule_start: { #2 }

```

```

1439     \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1440   }{
1441     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1442     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1443   }
1444 }

```

(End definition for `\stexpatchmodule`. This function is documented on page 29.)

## 28.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n
1445 \NewDocumentCommand \STEXModule { m } {
1446   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1447   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1448   \tl_set:Nn \l_tmpa_tl {
1449     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1450   }
1451   \seq_map_inline:Nn \l_stex_all_modules_seq {
1452     \str_set:Nn \l_tmpb_str { ##1 }
1453     \str_if_eq:eeT { \l_tmpa_str } {
1454       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1455     } {
1456       \seq_map_break:n {
1457         \tl_set:Nn \l_tmpa_tl {
1458           \stex_invoke_module:n { ##1 }
1459         }
1460       }
1461     }
1462   }
1463   \l_tmpa_tl
1464 }
1465
1466 \cs_new_protected:Nn \stex_invoke_module:n {
1467   \stex_debug:nn{modules}{Invoking~module~#1}
1468   \peek_charcode_remove:NTF ! {
1469     \__stex_modules_invoke_uri:nN { #1 }
1470   } {
1471     \peek_charcode_remove:NTF ? {
1472       \__stex_modules_invoke_symbol:nn { #1 }
1473     } {
1474       \msg_error:nnx{stex}{error/syntax}{
1475         ?~or~!~expected~after~
1476         \c_backslash_str STEXModule{#1}
1477       }
1478     }
1479   }
1480 }
1481
1482 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1483   \str_set:Nn #2 { #1 }
1484 }
1485

```

```

1486 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1487   \stex_invoke_symbol:n{#1?#2}
1488 }

```

*(End definition for \STEXModule and \stex\_invoke\_module:n. These functions are documented on page 29.)*

**\stex\_activate\_module:n**

```

1489 \bool_new:N \l_stex_in_meta_bool
1490 \bool_set_false:N \l_stex_in_meta_bool
1491 \cs_new_protected:Nn \stex_activate_module:n {
1492   \stex_debug:nn{modules}{Activating~module~#1}
1493   \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1494     \msg_error:nnn{stex}{error/conflictingmodules}{ #1 }
1495   }
1496   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1497     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1498     \use:c{ c_stex_module_#1_code }
1499   }
1500 }

```

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

```

1501 \</package>

```



## Chapter 29

# STEX -Module Inheritance Implementation

```
1502 <*package>
1503
1504 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1505
```

### 29.1 SMS Mode

```
1506 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1507 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1508 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1509 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1510
1511 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1512   \makeatletter
1513   \makeatother
1514   \ExplSyntaxOn
1515   \ExplSyntaxOff
1516   \rustexBREAK
1517 }
1518
1519 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1520   \symdef
1521   \importmodule
1522   \notation
1523   \symdecl
1524   \STEXexport
1525   \inlineass
1526   \inlinedef
1527   \inlineex
1528   \endinput
1529   \setnotation
```

```

1530 \copynotation
1531 }
1532
1533 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1534   \tl_to_str:n {
1535     smodule,
1536     copymodule,
1537     interpretmodule,
1538     sdefinition,
1539     sexample,
1540     sassertion,
1541     sparagraph
1542   }
1543 }

```

(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`

```

1544 \bool_new:N \g__stex_smsmode_bool
1545 \bool_set_false:N \g__stex_smsmode_bool
1546 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1547   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1548 }

```

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

`\_stex_smsmode_in_smsmode:nn`

```

1549 \cs_new_protected:Nn \_stex_smsmode_in_smsmode:nn {
1550   \vbox_set:Nn \l_tmpa_box {
1551     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1552     \bool_gset_true:N \g__stex_smsmode_bool
1553     #2
1554     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1555   }
1556   \box_clear:N \l_tmpa_box
1557 }

```

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

`\stex_file_in_smsmode:nn`

```

1558 \quark_new:N \q__stex_smsmode_break
1559
1560 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1561   \stex_filestack_push:n{#1}
1562   \_stex_smsmode_in_smsmode:nn{#1} {
1563     #2
1564     \everyeof{\q__stex_smsmode_break\noexpand}
1565     \expandafter\expandafter\expandafter
1566     \stex_smsmode_do:
1567     \csname @ @ input\endcsname "#1"\relax
1568   }
1569   \stex_filestack_pop:
1570 }

```

(End definition for `\stex_file_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:

```

1571 \cs_new_protected:Npn \stex_smsmode_do: {
1572   \stex_if_smsmode:T {
1573     \__stex_smsmode_do:w
1574   }
1575 }
1576 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1577   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
1578     \expandafter\if\expandafter\relax\noexpand#1
1579     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1580   \else\expandafter\__stex_smsmode_do:w\fi
1581   }{
1582     \__stex_smsmode_do:w % #1
1583   }
1584 }
1585 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1586   \cs_if_eq:NNTF #1 \q__stex_smsmode_break {
1587     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1588       #1\__stex_smsmode_do:w
1589     }{
1590       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1591         #1
1592       }{
1593         \cs_if_eq:NNTF \begin #1 {
1594           \__stex_smsmode_check_begin:n
1595         }{
1596           \cs_if_eq:NNTF \end #1 {
1597             \__stex_smsmode_check_end:n
1598           }{
1599             \__stex_smsmode_do:w
1600           }
1601         }
1602       }
1603     }
1604   }
1605 }
1606
1607 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1608   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1609     \begin{#1}
1610   }{
1611     \__stex_smsmode_do:w
1612   }
1613 }
1614 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1615   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1616     \end{#1}\__stex_smsmode_do:w
1617   }{
1618     \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1619   }
1620 }

```

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

## 29.2 Inheritance

```

1621 <@@=stex_importmodule>

\stex_import_module_uri:nn

1622 \cs_new_protected:Nn \stex_import_module_uri:nn {
1623   \str_set:Nx \l_stex_import_archive_str { #1 }
1624   \str_set:Nn \l_stex_import_path_str { #2 }
1625
1626   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1627   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1628   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1629
1630   \stex_modules_current_namespace:
1631   \bool_lazy_all:nTF {
1632     {\str_if_empty_p:N \l_stex_import_archive_str}
1633     {\str_if_empty_p:N \l_stex_import_path_str}
1634     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1635   }{
1636     \str_set_eq:NN \l_stex_import_path_str \l_stex_modules_subpath_str
1637     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1638   }{
1639     \str_if_empty:NT \l_stex_import_archive_str {
1640       \prop_if_exist:NT \l_stex_current_repository_prop {
1641         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1642       }
1643     }
1644     \str_if_empty:NTF \l_stex_import_archive_str {
1645       \str_if_empty:NF \l_stex_import_path_str {
1646         \str_set:Nx \l_stex_import_ns_str {
1647           \l_stex_module_ns_str / \l_stex_import_path_str
1648         }
1649       }
1650     }{
1651       \stex_require_repository:n \l_stex_import_archive_str
1652       \prop_get:cnN { c_stex_mathhub \l_stex_import_archive_str _manifest_prop } { ns }
1653       \l_stex_import_ns_str
1654       \str_if_empty:NF \l_stex_import_path_str {
1655         \str_set:Nx \l_stex_import_ns_str {
1656           \l_stex_import_ns_str / \l_stex_import_path_str
1657         }
1658       }
1659     }
1660   }
1661 }
```

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

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

```
1665 \str_new:N \l_stex_import_ns_str
```

(End definition for `\l_stex_import_name_str` and others. These variables are documented on page 33.)

```
\stex_import_require_module:nnnn
    {\ns} {\<archive-ID>} {\<path>} {\<name>}
1666 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1667   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1668
1669     % archive
1670     \str_set:Nx \l_tmpa_str { #2 }
1671     \str_if_empty:NTF \l_tmpa_str {
1672       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1673     } {
1674       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1675       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1676       \seq_put_right:Nn \l_tmpa_seq { source }
1677     }
1678
1679     % path
1680     \str_set:Nx \l_tmpb_str { #3 }
1681     \str_if_empty:NTF \l_tmpb_str {
1682       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1683
1684       \ltx@ifpackageloaded{babel} {
1685         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1686           { \language } \l_tmpb_str {
1687           \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1688         }
1689       } {
1690         \str_clear:N \l_tmpb_str
1691       }
1692
1693       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1694       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1695         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1696       }{
1697         \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1698         \IfFileExists{ \l_tmpa_str.tex }{
1699           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1700         }{
1701           % try english as default
1702           \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1703           \IfFileExists{ \l_tmpa_str.en.tex }{
1704             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1705           }{
1706             \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1707           }
1708         }
1709       }
1710
1711     } {
1712       \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1713       \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1714
```

```

1715 \ltx@ifpackageloaded{babel} {
1716   \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1717   { \language } \l_tmpb_str {
1718     \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1719   }
1720 } {
1721   \str_clear:N \l_tmpb_str
1722 }
1723
1724 \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1725
1726 \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1727 \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1728   \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1729 }{
1730   \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1731   \IfFileExists{ \l_tmpa_str/#4.tex }{
1732     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1733   }{
1734     % try english as default
1735     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1736     \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1737       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1738     }{
1739       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1740       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1741         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1742       }{
1743         \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1744         \IfFileExists{ \l_tmpa_str.tex }{
1745           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1746         }{
1747           % try english as default
1748           \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1749           \IfFileExists{ \l_tmpa_str.en.tex }{
1750             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1751           }{
1752             \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1753           }
1754         }
1755       }
1756     }
1757   }
1758 }
1759 }
1760
1761 \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
1762   \seq_clear:N \l_stex_all_modules_seq
1763   \str_clear:N \l_stex_current_module_str
1764   \str_set:Nx \l_tmpb_str { #2 }
1765   \str_if_empty:NF \l_tmpb_str {
1766     \stex_set_current_repository:n { #2 }
1767   }
1768   \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}

```

```

1769     }
1770
1771     \stex_if_module_exists:nF { #1 ? #4 } {
1772       \msg_error:nnx{stex}{error/unknownmodule}{
1773         #1?#4~(in~file~\g__stex_importmodule_file_str)
1774       }
1775     }
1776   }
1777   \stex_activate_module:n { #1 ? #4 }
1778 }

```

(End definition for `\stex_import_require_module:nnnn`. This function is documented on page 33.)

### `\importmodule`

```

1779 \NewDocumentCommand \importmodule { 0{ } m } {
1780   \stex_import_module_uri:nn { #1 } { #2 }
1781   \stex_debug:nn{modules}{Importing~module:~
1782     \l_stex_import_ns_str ? \l_stex_import_name_str
1783   }
1784   \stex_if_smsmode:F {
1785     \stex_import_require_module:nnnn
1786     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1787     { \l_stex_import_path_str } { \l_stex_import_name_str }
1788     \stex_annotate_invisible:nnn
1789     {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1790   }
1791   \exp_args:Nx \stex_add_to_current_module:n {
1792     \stex_import_require_module:nnnn
1793     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1794     { \l_stex_import_path_str } { \l_stex_import_name_str }
1795   }
1796   \exp_args:Nx \stex_add_import_to_current_module:n {
1797     \l_stex_import_ns_str ? \l_stex_import_name_str
1798   }
1799   \stex_smsmode_do:
1800   \ignorespacesandpars
1801 }
1802 \stex_deactivate_macro:Nn \importmodule {module-environments}

```

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

### `\usemodule`

```

1803 \NewDocumentCommand \usemodule { 0{ } m } {
1804   \stex_if_smsmode:F {
1805     \stex_import_module_uri:nn { #1 } { #2 }
1806     \stex_import_require_module:nnnn
1807     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1808     { \l_stex_import_path_str } { \l_stex_import_name_str }
1809     \stex_annotate_invisible:nnn
1810     {usemodule} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1811   }
1812   \stex_smsmode_do:
1813   \ignorespacesandpars
1814 }

```

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

1815 `\endpackage`



## Chapter 30

# STEX -Symbols Implementation

```
1816 <*package>
1817
1818 %%%%%%%%%%% symbols.dtx %%%%%%%%%%%
1819
    Warnings and error messages
1820 \msg_new:nnn{stex}{error/wrongargs}{
1821   args~value~in~symbol~declaration~for~#1~
1822   needs~to~be~i,~a,~b~or~B,~but~#2~given
1823 }
```

### 30.1 Symbol Declarations

```
1824 <@@=stex_symdecl>
\l_stex_all_symbols_seq Stores all available symbols
1825 \seq_new:N \l_stex_all_symbols_seq
(End definition for \l_stex_all_symbols_seq. This variable is documented on page 35.)
```

**\STEXsymbol**

```
1826 \NewDocumentCommand \STEXsymbol { m } {
1827   \stex_get_symbol:n { #1 }
1828   \exp_args:No
1829   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1830 }
```

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

symdecl arguments:

```
1831 \keys_define:nn { stex / symdecl } {
1832   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1833   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1834   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1835   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
1836   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,
1837   align     .str_set:N   = \l_stex_symdecl_align_str , % TODO(?)
```

```

1838 gfc .str_set:N = \l_stex_symdecl_gfc_str , % TODO(?)
1839 specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
1840 def .tl_set:N = \l_stex_symdecl_definiens_tl ,
1841 assoc .choices:nn =
1842 {bin,binl,binr,pre,conj,pwconj}
1843 {\str_set:Nx \l_stex_symdecl_astype_str {\l_keys_choice_tl}}
1844 }
1845
1846 \bool_new:N \l_stex_symdecl_make_macro_bool
1847
1848 \cs_new_protected:Nn \__stex_symdecl_args:n {
1849 \str_clear:N \l_stex_symdecl_name_str
1850 \str_clear:N \l_stex_symdecl_args_str
1851 \str_clear:N \l_stex_symdecl_deprecate_str
1852 \str_clear:N \l_stex_symdecl_astype_str
1853 \bool_set_false:N \l_stex_symdecl_local_bool
1854 \tl_clear:N \l_stex_symdecl_type_tl
1855 \tl_clear:N \l_stex_symdecl_definiens_tl
1856
1857 \keys_set:nn { stex / symdecl } { #1 }
1858 }

```

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

```

1859
1860 \NewDocumentCommand \symdecl { s O{} m } {
1861 \__stex_symdecl_args:n { #2 }
1862 \IfBooleanTF #1 {
1863 \bool_set_false:N \l_stex_symdecl_make_macro_bool
1864 } {
1865 \bool_set_true:N \l_stex_symdecl_make_macro_bool
1866 }
1867 \stex_symdecl_do:n { #3 }
1868 \stex_smsmode_do:
1869 }
1870
1871 \cs_new_protected:Nn \stex_symdecl_do:nn {
1872 \__stex_symdecl_args:n{#1}
1873 \bool_set_false:N \l_stex_symdecl_make_macro_bool
1874 \stex_symdecl_do:n{#2}
1875 }
1876
1877 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

(End definition for `\symdecl`. This function is documented on page 34.)

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

```

1878 \cs_new_protected:Nn \stex_symdecl_do:n {
1879 \stex_if_in_module:F {
1880 % TODO throw error? some default namespace?
1881 }
1882
1883 \str_if_empty:NT \l_stex_symdecl_name_str {
1884 \str_set:Nx \l_stex_symdecl_name_str { #1 }

```

```

1885 }
1886
1887 \prop_if_exist:cT { \l_stex_symdecl_
1888   \l_stex_current_module_str ?
1889   \l_stex_symdecl_name_str
1890   _prop
1891 }{
1892   % TODO throw error (beware of circular dependencies)
1893 }
1894
1895 \prop_clear:N \l_tmpa_prop
1896 \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
1897 \seq_clear:N \l_tmpa_seq
1898 \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1899 \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1900
1901 \str_if_empty:NT \l_stex_symdecl_deprecate_str {
1902   \str_if_empty:NF \l_stex_module_deprecate_str {
1903     \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
1904   }
1905 }
1906 \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
1907
1908 \exp_args:No \stex_add_constant_to_current_module:n {
1909   \l_stex_symdecl_name_str
1910 }
1911
1912 % arity/args
1913 \int_zero:N \l_tmpb_int
1914
1915 \bool_set_true:N \l_tmpa_bool
1916 \str_map_inline:Nn \l_stex_symdecl_args_str {
1917   \token_case_meaning:NnF ##1 {
1918     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1919     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1920     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1921     {\tl_to_str:n a} {
1922       \bool_set_false:N \l_tmpa_bool
1923       \int_incr:N \l_tmpb_int
1924     }
1925     {\tl_to_str:n B} {
1926       \bool_set_false:N \l_tmpa_bool
1927       \int_incr:N \l_tmpb_int
1928     }
1929   }{
1930     \msg_error:nnxx{stex}{error/wrongargs}{
1931       \l_stex_current_module_str ?
1932       \l_stex_symdecl_name_str
1933     }{##1}
1934   }
1935 }
1936 \bool_if:NTF \l_tmpa_bool {
1937   % possibly numeric
1938   \str_if_empty:NTF \l_stex_symdecl_args_str {

```

```

1939     \prop_put:Nnn \l_tmpa_prop { args } {}
1940     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1941   }{
1942     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1943     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1944     \str_clear:N \l_tmpa_str
1945     \int_step_inline:nn \l_tmpa_int {
1946       \str_put_right:Nn \l_tmpa_str i
1947     }
1948     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1949   }
1950 } {
1951   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1952   \prop_put:Nnx \l_tmpa_prop { arity }
1953     { \str_count:N \l_stex_symdecl_args_str }
1954 }
1955 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
1956
1957
1958 % semantic macro
1959
1960 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1961   \exp_args:Nx \stex_do_up_to_module:n {
1962     \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1963       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1964     }}
1965   }
1966
1967   \bool_if:NF \l_stex_symdecl_local_bool {
1968     \exp_args:Nx \stex_add_to_current_module:n {
1969       \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1970         \l_stex_current_module_str ? \l_stex_symdecl_name_str
1971       } }
1972     }
1973   }
1974 }
1975
1976 % add to all symbols
1977
1978 \bool_if:NF \l_stex_symdecl_local_bool {
1979   \exp_args:Nx \stex_add_to_current_module:n {
1980     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1981       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1982     }
1983   }
1984   % \exp_args:Nx \stex_add_field_to_current_module:n {
1985   %   \l_stex_current_module_str ? \l_stex_symdecl_name_str
1986   % }
1987 }
1988
1989 \stex_debug:nn{symbols}{New~symbol:~
1990   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
1991   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
1992   Args:~\prop_item:Nn \l_tmpa_prop { args }

```

```

1993 }
1994
1995 % circular dependencies require this:
1996
1997 \prop_if_exist:cF {
1998   l_stex_symdecl_
1999   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2000   _prop
2001 } {
2002   \prop_set_eq:cN {
2003     l_stex_symdecl_
2004     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2005     _prop
2006   } \l_tmpa_prop
2007 }
2008
2009 \seq_clear:c {
2010   l_stex_symdecl_
2011   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2012   _notations
2013 }
2014
2015 \bool_if:NF \l_stex_symdecl_local_bool {
2016   \exp_args:Nx
2017   \stex_add_to_current_module:n {
2018     \seq_clear:c {
2019       l_stex_symdecl_
2020       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2021       _notations
2022     }
2023     \prop_set_from_keyval:cn {
2024       l_stex_symdecl_
2025       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2026       _prop
2027     } {
2028       name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2029       module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2030       type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2031       args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2032       arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2033       assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
2034     }
2035   }
2036 }
2037
2038 \stex_if_smsmode:F {
2039   \exp_args:Nx \stex_do_up_to_module:n {
2040     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2041       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2042     }
2043   }
2044   \stex_if_do_html:T {
2045     \stex_annotate_invisible:nnn {symdecl} {
2046       \l_stex_current_module_str ? \l_stex_symdecl_name_str

```

```

2047 } {
2048   \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}{}}{${\l_st
2049   \stex_annotate_invisible:nnn{args}{}}{
2050     \prop_item:Nn \l_tmpa_prop { args }
2051   }
2052   \stex_annotate_invisible:nnn{macroname}{#1}{}
2053   \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2054     \stex_annotate_invisible:nnn{definiens}{}
2055     {${\l_stex_symdecl_definiens_tl$}
2056   }
2057   \str_if_empty:NF \l_stex_symdecl_assocotype_str {
2058     \stex_annotate_invisible:nnn{assocotype}{\l_stex_symdecl_assocotype_str}{}
2059   }
2060 }
2061 }
2062 }
2063 }

```

(End definition for `\stex_symdecl_do:n`. This function is documented on page 35.)

`\stex_get_symbol:n`

```

2064 \str_new:N \l_stex_get_symbol_uri_str
2065
2066 \cs_new_protected:Nn \stex_get_symbol:n {
2067   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2068     \__stex_symdecl_get_symbol_from_cs:n { #1 }
2069   }{
2070     % argument is a string
2071     % is it a command name?
2072     \cs_if_exist:cTF { #1 }{
2073       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2074       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2075       \str_if_empty:NTF \l_tmpa_str {
2076         \exp_args:Nx \cs_if_eq:NNTF {
2077           \tl_head:N \l_tmpa_tl
2078         } \stex_invoke_symbol:n {
2079           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
2080         }{
2081           \__stex_symdecl_get_symbol_from_string:n { #1 }
2082         }
2083       } {
2084         \__stex_symdecl_get_symbol_from_string:n { #1 }
2085       }
2086     }{
2087       % argument is not a command name
2088       \__stex_symdecl_get_symbol_from_string:n { #1 }
2089       % \l_stex_all_symbols_seq
2090     }
2091   }
2092   \str_if_eq:eeF {
2093     \prop_item:cn {
2094       \l_stex_symdecl\l_stex_get_symbol_uri_str _prop
2095     }{ deprecate }
2096   }{}{

```

```

2097 \msg_warning:nnxx{stex}{warning/deprecated}{
2098 Symbol~\l_stex_get_symbol_uri_str
2099 }{
2100 \prop_item:cn {l_stex_symdecl_l_stex_get_symbol_uri_str _prop}{ deprecate }
2101 }
2102 }
2103 }
2104
2105 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2106 \str_set:Nn \l_tmpa_str { #1 }
2107 \bool_set_false:N \l_tmpa_bool
2108 \stex_if_in_module:T {
2109 \exp_args:Nno \seq_if_in:cnT {c_stex_module_l_stex_current_module_str _constants} { \l_
2110 \bool_set_true:N \l_tmpa_bool
2111 \str_set:Nx \l_stex_get_symbol_uri_str {
2112 \l_stex_current_module_str ? #1
2113 }
2114 }
2115 }
2116 \bool_if:NF \l_tmpa_bool {
2117 \tl_set:Nn \l_tmpa_tl {
2118 \msg_set:nnn{stex}{error/unknownsymbol}{
2119 No~symbol~#1~found!
2120 }
2121 \msg_error:nn{stex}{error/unknownsymbol}
2122 }
2123 \str_set:Nn \l_tmpa_str { #1 }
2124 \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2125 \seq_map_inline:Nn \l_stex_all_symbols_seq {
2126 \str_set:Nn \l_tmpb_str { ##1 }
2127 \str_if_eq:eeT { \l_tmpa_str } {
2128 \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
2129 } {
2130 \seq_map_break:n {
2131 \tl_set:Nn \l_tmpa_tl {
2132 \str_set:Nn \l_stex_get_symbol_uri_str {
2133 ##1
2134 }
2135 }
2136 }
2137 }
2138 }
2139 \l_tmpa_tl
2140 }
2141 }
2142
2143 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
2144 \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2145 { \tl_tail:N \l_tmpa_tl }
2146 \tl_if_single:NTF \l_tmpa_tl {
2147 \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2148 \exp_after:wN \str_set:Nn \exp_after:wN
2149 \l_stex_get_symbol_uri_str \l_tmpa_tl
2150 }{

```

```

2151     % TODO
2152     % tail is not a single group
2153   }
2154 }{
2155   % TODO
2156   % tail is not a single group
2157 }
2158 }

```

(End definition for `\stex_get_symbol:n`. This function is documented on page 35.)

## 30.2 Notations

```

2159 <@@=stex_notation>
      notation arguments:
2160 \keys_define:nn { stex / notation } {
2161   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2162   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2163   prec .str_set_x:N = \l__stex_notation_prec_str ,
2164   op .tl_set:N = \l__stex_notation_op_tl ,
2165   primary .bool_set:N = \l__stex_notation_primary_bool ,
2166   primary .default:n = {true} ,
2167   unknown .code:n = \str_set:Nx
2168     \l__stex_notation_variant_str \l_keys_key_str
2169 }
2170
2171 \cs_new_protected:Nn \_stex_notation_args:n {
2172   \str_clear:N \l__stex_notation_lang_str
2173   \str_clear:N \l__stex_notation_variant_str
2174   \str_clear:N \l__stex_notation_prec_str
2175   \tl_clear:N \l__stex_notation_op_tl
2176   \bool_set_false:N \l__stex_notation_primary_bool
2177
2178   \keys_set:nn { stex / notation } { #1 }
2179 }

```

**\notation**

```

2180 \NewDocumentCommand \notation { s O{} m } {
2181   \_stex_notation_args:n { #2 }
2182   \tl_clear:N \l_stex_symdecl_definiens_tl
2183   \stex_get_symbol:n { #3 }
2184   \tl_set:Nn \l_stex_notation_after_do_tl {
2185     \__stex_notation_final:
2186     \IfBooleanTF#1{
2187       \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2188     }{}
2189     \stex_smsmode_do:
2190   }
2191   \stex_notation_do:nnnn
2192   { \prop_item:cn {l_stex_symdecl\_l_stex_get_symbol_uri_str_prop } { args } }
2193   { \prop_item:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str_prop } { arity } }
2194   { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2195 }
2196 \stex_deactivate_macro:Nn \notation {module~environments}

```



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

\stex\_notation\_do:nnnn

```

2197 \seq_new:N \l__stex_notation_precedences_seq
2198 \tl_new:N \l__stex_notation_opprec_tl
2199 \int_new:N \l__stex_notation_currarg_int
2200 \tl_new:N \stex_symbol_after_invokation_tl
2201
2202 \cs_new_protected:Nn \stex_notation_do:nnnn {
2203   \let\l_stex_current_symbol_str\relax
2204   \seq_clear:N \l__stex_notation_precedences_seq
2205   \tl_clear:N \l__stex_notation_opprec_tl
2206   \str_set:Nx \l__stex_notation_args_str { #1 }
2207   \str_set:Nx \l__stex_notation_arity_str { #2 }
2208   \str_set:Nx \l__stex_notation_suffix_str { #3 }
2209
2210   % precedences
2211   \str_if_empty:NTF \l__stex_notation_prec_str {
2212     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2213       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2214     }{
2215       \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2216     }
2217   } {
2218     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2219       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2220       \int_step_inline:nn { \l__stex_notation_arity_str } {
2221         \exp_args:NNo
2222         \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2223       }
2224     }{
2225       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2226       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2227         \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2228         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2229           \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2230             \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2231           \seq_map_inline:Nn \l_tmpa_seq {
2232             \seq_put_right:Nn \l_tmpb_seq { ##1 }
2233           }
2234         }
2235       }{
2236         \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2237           \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2238         }{
2239           \tl_set:No \l__stex_notation_opprec_tl { 0 }
2240         }
2241       }
2242     }
2243   }
2244
2245   \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2246   \int_step_inline:nn { \l__stex_notation_arity_str } {
2247     \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {

```

```

2248     \exp_args:NNo
2249     \seq_put_right:No \l__stex_notation_precedences_seq {
2250         \l__stex_notation_opprec_tl
2251     }
2252 }
2253 }
2254 \tl_clear:N \l_stex_notation_dummyargs_tl
2255
2256 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2257     \exp_args:NNe
2258     \cs_set:Npn \l_stex_notation_macrocode_cs {
2259         \_stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2260         { \_stex_notation_suffix_str }
2261         { \l__stex_notation_opprec_tl }
2262         { \exp_not:n { #4 } }
2263     }
2264     \l_stex_notation_after_do_tl
2265 }{
2266     \str_if_in:NnTF \l__stex_notation_args_str b {
2267         \exp_args:Nne \use:nn
2268         {
2269             \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2270             \cs_set:Npn \l__stex_notation_arity_str } { {
2271                 \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2272                 { \_stex_notation_suffix_str }
2273                 { \l__stex_notation_opprec_tl }
2274                 { \exp_not:n { #4 } }
2275             } }
2276 }{
2277     \str_if_in:NnTF \l__stex_notation_args_str B {
2278         \exp_args:Nne \use:nn
2279         {
2280             \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2281             \cs_set:Npn \l__stex_notation_arity_str } { {
2282                 \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2283                 { \_stex_notation_suffix_str }
2284                 { \l__stex_notation_opprec_tl }
2285                 { \exp_not:n { #4 } }
2286             } }
2287 }{
2288     \exp_args:Nne \use:nn
2289     {
2290         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2291         \cs_set:Npn \l__stex_notation_arity_str } { {
2292             \_stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2293             { \_stex_notation_suffix_str }
2294             { \l__stex_notation_opprec_tl }
2295             { \exp_not:n { #4 } }
2296         } }
2297     }
2298 }
2299
2300 \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2301 \int_zero:N \l__stex_notation_currarg_int

```

```

2302     \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2303     \__stex_notation_arguments:
2304   }
2305 }

```

(End definition for \stex\_notation\_do:nnnn. This function is documented on page ??.)

\\_\_stex\_notation\_arguments: Takes care of annotating the arguments in a notation macro

```

2306 \cs_new_protected:Nn \__stex_notation_arguments: {
2307   \int_incr:N \l__stex_notation_currarg_int
2308   \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2309     \l_stex_notation_after_do_tl
2310   }{
2311     \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2312     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2313     \str_if_eq:VnTF \l_tmpa_str a {
2314       \__stex_notation_argument_assoc:n
2315     }{
2316       \str_if_eq:VnTF \l_tmpa_str B {
2317         \__stex_notation_argument_assoc:n
2318       }{
2319         \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2320         \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2321           { \stex_term_math_arg:nnn
2322             { \int_use:N \l__stex_notation_currarg_int }
2323             { \l_tmpa_str }
2324             { ####\int_use:N \l__stex_notation_currarg_int }
2325           }
2326         }
2327         \__stex_notation_arguments:
2328       }
2329     }
2330   }
2331 }

```

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

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

```

2332 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
2333
2334   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2335     {\l__stex_notation_arity_str}{
2336     #1
2337   }
2338   \int_zero:N \l_tmpa_int
2339   \tl_clear:N \l_tmpa_tl
2340   \str_map_inline:Nn \l__stex_notation_args_str {
2341     \int_incr:N \l_tmpa_int
2342     \tl_put_right:Nx \l_tmpa_tl {
2343       \str_if_eq:nnTF {##1}{a}{ {} }{
2344         \str_if_eq:nnTF {##1}{B}{ {} }{
2345           {\stex_term_arg:nn{\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa_in
2346         }
2347     }

```

```

2348     }
2349   }
2350   \exp_after:wN\exp_after:wN\exp_after:wN \def
2351   \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2352   \exp_after:wN\exp_after:wN\exp_after:wN ##
2353   \exp_after:wN\exp_after:wN\exp_after:wN 1
2354   \exp_after:wN\exp_after:wN\exp_after:wN ##
2355   \exp_after:wN\exp_after:wN\exp_after:wN 2
2356   \exp_after:wN\exp_after:wN\exp_after:wN {
2357     \exp_after:wN \exp_after:wN \exp_after:wN
2358     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2359       \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2360     }
2361   }
2362
2363   \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2364   \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
2365     \stex_term_math_assoc_arg:nnnn
2366     { \int_use:N \l__stex_notation_currarg_int }
2367     { \l_tmpa_str }
2368     { ####\int_use:N \l__stex_notation_currarg_int }
2369     { \l_tmpa_cs {####1} {####2} }
2370   } }
2371   \__stex_notation_arguments:
2372 }

```

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

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

```

2373 \cs_new_protected:Nn \__stex_notation_final: {
2374   \exp_args:Nne \use:nn
2375   {
2376     \cs_generate_from_arg_count:cNnn {
2377       stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2378       \__stex_notation_suffix_str
2379       _cs
2380     }
2381     \cs_set:Npn \l__stex_notation_arity_str { { {
2382       \exp_after:wN \exp_after:wN \exp_after:wN
2383       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2384       { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_symb
2385     } }
2386
2387     \tl_if_empty:NF \l__stex_notation_op_tl {
2388       \cs_set:cpx {
2389         stex_op_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2390         \__stex_notation_suffix_str
2391         _cs
2392       } {
2393         \stex_term_oms:nnn {
2394           \l_stex_get_symbol_uri_str \c_hash_str \__stex_notation_suffix_str
2395         }{
2396           \l_stex_get_symbol_uri_str
2397         }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } } }

```

```

2398     }
2399 }
2400
2401 \exp_args:Ne
2402 \stex_add_to_current_module:n {
2403   \cs_generate_from_arg_count:cNnn {
2404     stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2405     \__stex_notation_suffix_str
2406     _cs
2407   } \cs_set:Npn {\l__stex_notation_arity_str} {
2408     \exp_after:wN \exp_after:wN \exp_after:wN
2409     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2410     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2411   }
2412   \tl_if_empty:NF \l__stex_notation_op_tl {
2413     \cs_set:cpn {
2414       stex_op_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2415       \__stex_notation_suffix_str
2416       _cs
2417     } {
2418       \_stex_term_oms:nnn {
2419         \l_stex_get_symbol_uri_str \c_hash_str \__stex_notation_suffix_str
2420       } {
2421         \l_stex_get_symbol_uri_str
2422       } { \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } } }
2423     }
2424   }
2425 }
2426 %\exp_args:Nx
2427 % \stex_do_up_to_module:n {
2428   \seq_put_right:cx {
2429     l_stex_symdecl_ \l_stex_get_symbol_uri_str
2430     _notations
2431   } {
2432     \__stex_notation_suffix_str
2433   }
2434 % }
2435
2436 \stex_debug:nn{symbols}{
2437   Notation~\__stex_notation_suffix_str
2438   ~for~\l_stex_get_symbol_uri_str^^J
2439   Operator~precedence:~\l__stex_notation_opprec_tl^^J
2440   Argument~precedences:~
2441   \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2442   Notation: \cs_meaning:c {
2443     stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2444     \__stex_notation_suffix_str
2445     _cs
2446   }
2447 }
2448
2449 \exp_args:Ne
2450 \stex_add_to_current_module:n {
2451   \seq_put_right:cn {

```

```

2452     \l_stex_symdecl \l_stex_get_symbol_uri_str
2453     _notations
2454   } { \_stex_notation_suffix_str }
2455 }
2456
2457 \stex_if_smsmode:F {
2458
2459   % HTML annotations
2460   \stex_if_do_html:T {
2461     \stex_annotate_invisible:nnn { notation }
2462     { \l_stex_get_symbol_uri_str } {
2463       \stex_annotate_invisible:nnn { notationfragment }
2464       { \_stex_notation_suffix_str }{}
2465       \stex_annotate_invisible:nnn { precedence }
2466       { \l_stex_notation_prec_str }{}
2467
2468       \int_zero:N \l_tmpa_int
2469       \str_set_eq:NN \l_stex_notation_remaining_args_str \l_stex_notation_args_str
2470       \tl_clear:N \l_tmpa_tl
2471       \int_step_inline:nn { \l_stex_notation_arity_str }{
2472         \int_incr:N \l_tmpa_int
2473         \str_set:Nx \l_tmpb_str { \str_head:N \l_stex_notation_remaining_args_str }
2474         \str_set:Nx \l_stex_notation_remaining_args_str { \str_tail:N \l_stex_notation_r
2475         \str_if_eq:VnTF \l_tmpb_str a {
2476           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2477             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2478             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2479           } }
2480         }{
2481           \str_if_eq:VnTF \l_tmpb_str B {
2482             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2483               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2484               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2485             } }
2486           }{
2487             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2488               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2489             } }
2490           }
2491         }
2492       }
2493       \stex_annotate_invisible:nnn { notationcomp }{}{
2494         \str_set:Nx \l_stex_current_symbol_str { \l_stex_get_symbol_uri_str }
2495         $ \exp_args:Nno \use:nn { \use:c {
2496           stex_notation_ \l_stex_current_symbol_str
2497           \c_hash_str \_stex_notation_suffix_str _cs
2498         } } { \l_tmpa_tl } $
2499       }
2500     }
2501   }
2502 }
2503 }

```

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

\setnotation

```

2504 \keys_define:nn { stex / setnotation } {
2505   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2506   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2507   unknown .code:n = \str_set:Nx
2508     \l__stex_notation_variant_str \l_keys_key_str
2509 }
2510
2511 \cs_new_protected:Nn \stex_setnotation_args:n {
2512   \str_clear:N \l__stex_notation_lang_str
2513   \str_clear:N \l__stex_notation_variant_str
2514   \keys_set:nn { stex / setnotation } { #1 }
2515 }
2516
2517 \cs_new_protected:Nn \stex_setnotation:n {
2518   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1 _notations }
2519     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2520     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2521       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2522     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2523       { \c_hash_str }
2524     \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_#1 _notations }
2525       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2526     \exp_args:Nx \stex_add_to_current_module:n {
2527       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2528         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2529       \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_#1 _notations }
2530         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2531       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2532         { \c_hash_str }
2533     }
2534     \stex_debug:nn {notations}{
2535       Setting~default~notation~
2536       {\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str}~for~
2537       #1 \\
2538       \expandafter\meaning\csname
2539         l_stex_symdecl_#1 _notations\endcsname
2540     }
2541   }{
2542     % todo throw error
2543   }
2544 }
2545
2546 \NewDocumentCommand \setnotation {m m} {
2547   \stex_get_symbol:n { #1 }
2548   \stex_setnotation_args:n { #2 }
2549   \stex_setnotation:n{\l_stex_get_symbol_uri_str}
2550   \stex_smsmode_do:
2551 }
2552
2553 \cs_new_protected:Nn \stex_copy_notations:nn {
2554   \stex_debug:nn {notations}{
2555     Copying~notations~from~#2~to~#1\\
2556     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}

```

```

2557 }
2558 \tl_clear:N \l_tmpa_tl
2559 \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2560   \tl_put_right:Nn \l_tmpa_tl { {## ##1} }
2561 }
2562 \seq_map_inline:cn {l_stex_symdecl_#2_notations}{
2563   \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2564   \edef \l_tmpa_tl {
2565     \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2566     \exp_after:wN\exp_after:wN\exp_after:wN {
2567       \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2568     }
2569   }
2570   \exp_args:Nx
2571   \stex_do_up_to_module:n {
2572     \seq_put_right:cn{l_stex_symdecl_#1_notations}{##1}
2573     \cs_generate_from_arg_count:cNnn {
2574       stex_notation_ #1 \c_hash_str ##1 _cs
2575     } \cs_set:Npn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }{
2576       \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl}
2577     }
2578   }
2579 }
2580 }
2581
2582 \NewDocumentCommand \copynotation {m m} {
2583   \stex_get_symbol:n { #1 }
2584   \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2585   \stex_get_symbol:n { #2 }
2586   \exp_args:Noo
2587   \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2588   \exp_args:Nx \stex_add_import_to_current_module:n{
2589     \stex_copy_notations:nn {\l_tmpa_str} {\l_stex_get_symbol_uri_str}
2590   }
2591   \stex_smsmode_do:
2592 }
2593

```

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

**\symdef**

```

2594 \keys_define:nn { stex / symdef } {
2595   name .str_set_x:N = \l_stex_symdecl_name_str ,
2596   local .bool_set:N = \l_stex_symdecl_local_bool ,
2597   args .str_set_x:N = \l_stex_symdecl_args_str ,
2598   type .tl_set:N = \l_stex_symdecl_type_tl ,
2599   def .tl_set:N = \l_stex_symdecl_definiens_tl ,
2600   op .tl_set:N = \l_stex_notation_op_tl ,
2601   lang .str_set_x:N = \l_stex_notation_lang_str ,
2602   variant .str_set_x:N = \l_stex_notation_variant_str ,
2603   prec .str_set_x:N = \l_stex_notation_prec_str ,
2604   assoc .choices:nn =
2605     {bin,binl,binr,pre,conj,pwconj}
2606     {\str_set:Nx \l_stex_symdecl_assoc_type_str {\l_keys_choice_tl}},

```



```

2607   unknown .code:n      = \str_set:Nx
2608       \l__stex_notation_variant_str \l_keys_key_str
2609 }
2610
2611 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2612   \str_clear:N \l_stex_symdecl_name_str
2613   \str_clear:N \l_stex_symdecl_args_str
2614   \str_clear:N \l_stex_symdecl_assoctype_str
2615   \bool_set_false:N \l_stex_symdecl_local_bool
2616   \tl_clear:N \l_stex_symdecl_type_tl
2617   \tl_clear:N \l_stex_symdecl_definiens_tl
2618   \str_clear:N \l__stex_notation_lang_str
2619   \str_clear:N \l__stex_notation_variant_str
2620   \str_clear:N \l__stex_notation_prec_str
2621   \tl_clear:N \l__stex_notation_op_tl
2622
2623   \keys_set:nn { stex / symdef } { #1 }
2624 }
2625
2626 \NewDocumentCommand \symdef { O{} m } {
2627   \__stex_notation_symdef_args:n { #1 }
2628   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2629   \stex_symdecl_do:n { #2 }
2630   \tl_set:Nn \l_stex_notation_after_do_tl {
2631     \__stex_notation_final:
2632     \stex_smsmode_do:
2633   }
2634   \str_set:Nx \l_stex_get_symbol_uri_str {
2635     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2636   }
2637   \exp_args:Nx \stex_notation_do:nnnn
2638     { \prop_item:cn { \l_stex_symdecl \l_stex_get_symbol_uri_str _prop } { args } }
2639     { \prop_item:cn { \l_stex_symdecl \l_stex_get_symbol_uri_str _prop } { arity } }
2640     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2641   }
2642   \stex_deactivate_macro:Nn \symdef {module~environments}

```

(End definition for `\symdef`. This function is documented on page [35](#).)

### 30.3 Variables

```

2643 <@@=stex_variables>
2644
2645 \keys_define:nn { stex / vardef } {
2646   name .str_set_x:N = \l__stex_variables_name_str ,
2647   args .str_set_x:N = \l__stex_variables_args_str ,
2648   type .tl_set:N    = \l__stex_variables_type_tl ,
2649   def  .tl_set:N    = \l__stex_variables_def_tl ,
2650   op   .tl_set:N    = \l__stex_variables_op_tl ,
2651   prec .str_set_x:N = \l__stex_variables_prec_str ,
2652   assoc .choices:nn =
2653     {bin,binl,binr,pre,conj,pwconj}
2654     {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
2655   bind .choices:nn =

```

```

2656     {forall,exists}
2657     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2658   }
2659
2660   \cs_new_protected:Nn \__stex_variables_args:n {
2661     \str_clear:N \l__stex_variables_name_str
2662     \str_clear:N \l__stex_variables_args_str
2663     \str_clear:N \l__stex_variables_prec_str
2664     \str_clear:N \l__stex_variables_assoctype_str
2665     \str_clear:N \l__stex_variables_bind_str
2666     \tl_clear:N \l__stex_variables_type_tl
2667     \tl_clear:N \l__stex_variables_def_tl
2668     \tl_clear:N \l__stex_variables_op_tl
2669
2670     \keys_set:nn { stex / vardef } { #1 }
2671   }
2672
2673   \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{} } {
2674     \__stex_variables_args:n {#2}
2675     \str_if_empty:NT \l__stex_variables_name_str {
2676       \str_set:Nx \l__stex_variables_name_str { #1 }
2677     }
2678     \prop_clear:N \l_tmpa_prop
2679     \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
2680
2681     \int_zero:N \l_tmpb_int
2682     \bool_set_true:N \l_tmpa_bool
2683     \str_map_inline:Nn \l__stex_variables_args_str {
2684       \token_case_meaning:NnF ##1 {
2685         0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2686         {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2687         {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2688         {\tl_to_str:n a} {
2689           \bool_set_false:N \l_tmpa_bool
2690           \int_incr:N \l_tmpb_int
2691         }
2692         {\tl_to_str:n B} {
2693           \bool_set_false:N \l_tmpa_bool
2694           \int_incr:N \l_tmpb_int
2695         }
2696       }{
2697         \msg_error:nnxx{stex}{error/wrongargs}{
2698           variable~\l__stex_variables_name_str
2699         }{##1}
2700       }
2701     }
2702     \bool_if:NTF \l_tmpa_bool {
2703       % possibly numeric
2704       \str_if_empty:NTF \l__stex_variables_args_str {
2705         \prop_put:Nnn \l_tmpa_prop { args } {}
2706         \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2707       }{
2708         \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
2709         \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }

```

```

2710 \str_clear:N \l_tmpa_str
2711 \int_step_inline:nn \l_tmpa_int {
2712   \str_put_right:Nn \l_tmpa_str i
2713 }
2714 \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
2715 \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2716 }
2717 } {
2718   \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2719   \prop_put:Nnx \l_tmpa_prop { arity }
2720   { \str_count:N \l__stex_variables_args_str }
2721 }
2722 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
2723 \tl_set:cx { #1 } { \stex_invoke_variable:n { \l__stex_variables_name_str } }
2724
2725 \prop_set_eq:cN { l_stex_variable_\l__stex_variables_name_str _prop } \l_tmpa_prop
2726
2727 \tl_if_empty:NF \l__stex_variables_op_tl {
2728   \cs_set:cpx {
2729     stex_var_op_notation_ \l__stex_variables_name_str _cs
2730   } {
2731     \stex_term_omv:nn {
2732       var://\l__stex_variables_name_str
2733     } { \comp { \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
2734   }
2735 }
2736
2737 \tl_set:Nn \l_stex_notation_after_do_tl {
2738   \exp_args:Nne \use:nn {
2739     \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str _cs }
2740     \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
2741   } { {
2742     \exp_after:wN \exp_after:wN \exp_after:wN
2743     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2744     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_symbol
2745   } }
2746 \stex_if_do_html:T {
2747   \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
2748     \stex_annotate_invisible:nnn { precedence }
2749     { \l__stex_variables_prec_str } { }
2750   \tl_if_empty:NF \l__stex_variables_type_tl { \stex_annotate_invisible:nnn {type} { } { $\l
2751   \stex_annotate_invisible:nnn {args} { } { \l__stex_variables_args_str }
2752   \stex_annotate_invisible:nnn {macroname} { #1 } { }
2753   \tl_if_empty:NF \l__stex_variables_def_tl {
2754     \stex_annotate_invisible:nnn {definiens} { }
2755     { $\l__stex_variables_def_tl $ }
2756   }
2757   \str_if_empty:NF \l__stex_variables_assoc_type_str {
2758     \stex_annotate_invisible:nnn {assoc_type} { \l__stex_variables_assoc_type_str } { }
2759   }
2760   \int_zero:N \l_tmpa_int
2761   \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
2762   \tl_clear:N \l_tmpa_tl
2763   \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } } {

```

```

2764 \int_incr:N \l_tmpa_int
2765 \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
2766 \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables
2767 \str_if_eq:VnTF \l_tmpb_str a {
2768 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2769 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2770 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2771 } }
2772 }{
2773 \str_if_eq:VnTF \l_tmpb_str B {
2774 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2775 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2776 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2777 } }
2778 }{
2779 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2780 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2781 } }
2782 }
2783 }
2784 }
2785 \stex_annotate_invisible:nnn { notationcomp }{}{
2786 \str_set:Nx \l_stex_current_symbol_str {var://\l__stex_variables_name_str }
2787 $ \exp_args:Nno \use:nn { \use:c {
2788 stex_var_notation_\l__stex_variables_name_str_cs
2789 } } { \l_tmpa_tl } $
2790 }
2791 }
2792 }
2793 }
2794
2795 \stex_notation_do:nnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { arit
2796 }
2797
2798 \cs_new:Nn \__stex_variables_reset:N {
2799 \tl_if_exist:NTF #1 {
2800 \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
2801 }{
2802 \let \exp_not:N #1 \exp_not:N \undefined
2803 }
2804 }
2805
2806 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
2807 \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
2808 \exp_args:Nnx \use:nn {
2809 % TODO
2810 \stex_annotate_invisible:nnn {vardecls}{\clist_use:Nn\l__stex_variables_names,}{
2811 #2
2812 }
2813 }{
2814 \__stex_variables_reset:N \varnot
2815 \__stex_variables_reset:N \vartype
2816 \__stex_variables_reset:N \vardef
2817 }

```

```

2818 }
2819
2820 \NewDocumentCommand \vardef { s } {
2821   \IfBooleanTF#1 {
2822     \__stex_variables_do_complex:nn
2823   }{
2824     \__stex_variables_do_simple:nnn
2825   }
2826 }
2827
2828
2829
2830
2831 \end{package}

```

## Chapter 31

# STEX -Terms Implementation

```
2832 <*package>
2833
2834 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2835
2836 <@@=stex_terms>
2837
2838 Warnings and error messages
2839 \msg_new:nnn{stex}{error/nonotation}{
2840   Symbol~#1~invoked,~but~has~no~notation~#2!
2841 }
2842 \msg_new:nnn{stex}{error/notationarg}{
2843   Error~in~parsing~notation~#1
2844 }
2845 \msg_new:nnn{stex}{error/noop}{
2846   Symbol~#1~has~no~operator~notation~for~notation~#2
2847 }
2848 \msg_new:nnn{stex}{error/notallowed}{
2849   Symbol~invocation~#1~not~allowed~in~notation~component~of~#2
2850 }
```

### 31.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
2850 \keys_define:nn { stex / terms } {
2851   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2852   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2853   unknown .code:n = \str_set:Nx
2854     \l__stex_terms_variant_str \l_keys_key_str
2855 }
2856
2857 \cs_new_protected:Nn \__stex_terms_args:n {
2858   \str_clear:N \l__stex_terms_lang_str
2859   \str_clear:N \l__stex_terms_variant_str
2860 }
```

```

2861 \keys_set:nn { stex / terms } { #1 }
2862 }
2863
2864 \cs_new:Nn \__stex_terms_reset:N {
2865   \tl_if_exist:NTF #1 {
2866     \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
2867   }{
2868     \let \exp_not:N #1 \exp_not:N \undefined
2869   }
2870 }
2871
2872 \bool_new:N \l__stex_terms_allow_semantic_bool
2873 \bool_set_true:N \l__stex_terms_allow_semantic_bool
2874
2875 \cs_new_protected:Nn \stex_invoke_symbol:n {
2876   \bool_if:NTF \l__stex_terms_allow_semantic_bool {
2877     \str_if_eq:eeF {
2878       \prop_item:cn {
2879         l_stex_symdecl_#1_prop
2880       }{ deprecate }
2881     }{}{
2882       \msg_warning:nxxx{stex}{warning/deprecated}{
2883         Symbol~#1
2884       }{
2885         \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
2886       }
2887     }
2888     \if_mode_math:
2889       \exp_after:wN \__stex_terms_invoke_math:n
2890     \else:
2891       \exp_after:wN \__stex_terms_invoke_text:n
2892     \fi: { #1 }
2893   }{
2894     \msg_error:nxxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
2895   }
2896 }
2897
2898 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
2899   \peek_charcode_remove:NTF ! {
2900     \__stex_terms_invoke_op_custom:nn {#1}
2901   }{
2902     \__stex_terms_invoke_custom:nn {#1}
2903   }
2904 }
2905
2906 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2907   \peek_charcode_remove:NTF ! {
2908     % operator
2909     \peek_charcode_remove:NTF * {
2910       % custom op
2911       \__stex_terms_invoke_op_custom:nn {#1}
2912     }{
2913       % op notation
2914       \peek_charcode:NTF [ {

```

```

2915         \__stex_terms_invoke_op_notation:nw {#1}
2916     }{
2917         \__stex_terms_invoke_op_notation:nw {#1}[]
2918     }
2919 }
2920 }{
2921     \peek_charcode_remove:NTF * {
2922         \__stex_terms_invoke_custom:nn {#1}
2923         % custom
2924     }{
2925         % normal
2926         \peek_charcode:NTF [ {
2927             \__stex_terms_invoke_notation:nw {#1}
2928         }{
2929             \__stex_terms_invoke_notation:nw {#1}[]
2930         }
2931     }
2932 }
2933 }
2934
2935
2936 \cs_new_protected:Nn \__stex_terms_invoke_op_custom:nn {
2937     \exp_args:Nnx \use:nn {
2938         \str_set:Nn \l_stex_current_symbol_str { #1 }
2939         \bool_set_false:N \l__stex_terms_allow_semantic_bool
2940         \stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
2941             \comp{ #2 }
2942         }
2943     }{
2944         \__stex_terms_reset:N \l_stex_current_symbol_str
2945         \bool_set_true:N \l__stex_terms_allow_semantic_bool
2946     }
2947 }
2948
2949 \cs_new_protected:Nn \__stex_terms_find_notation:nn {
2950     \str_set:Nn \l_stex_current_symbol_str { #1 }
2951     \__stex_terms_args:n { #2 }
2952     \seq_if_empty:cTF {
2953         l_stex_symdecl_ #1 _notations
2954     } {
2955         \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
2956     } {
2957         \bool_lazy_all:nTF {
2958             {\str_if_empty_p:N \l__stex_terms_variant_str}
2959             {\str_if_empty_p:N \l__stex_terms_lang_str}
2960         }{
2961             \seq_get_left:cN {l_stex_symdecl_#1_notations}\l__stex_terms_variant_str
2962         }{
2963             \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
2964                 \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2965             }{
2966                 \str_set:Nx \l__stex_terms_variant_str { \l__stex_terms_variant_str \c_hash_str \l__
2967             }{
2968                 \msg_error:nnxx{stex}{error/nonotation}{#1}{

```



```

2969         ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2970     }
2971 }
2972 }
2973 }
2974 }
2975
2976 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
2977     \__stex_terms_find_notation:nn { #1 }{ #2 }
2978     \bool_set_false:N \l__stex_terms_allow_semantic_bool
2979     \cs_if_exist:cTF {
2980         stex_op_notation_ #1 \c_hash_str \l__stex_terms_variant_str _cs
2981     }{
2982         \use:c{stex_op_notation_ #1 \c_hash_str \l__stex_terms_variant_str _cs}
2983     }{
2984         \msg_error:nnxx{stex}{error/noop}{#1}{\l__stex_terms_variant_str}
2985     }
2986     \bool_set_true:N \l__stex_terms_allow_semantic_bool
2987 }
2988
2989 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {
2990     \__stex_terms_find_notation:nn { #1 }{ #2 }
2991     \cs_if_exist:cTF {
2992         stex_notation_ #1 \c_hash_str \l__stex_terms_variant_str _cs
2993     }{
2994         \tl_set:Nx \stex_symbol_after_invokation_tl {
2995             \__stex_terms_reset:N \stex_symbol_after_invokation_tl
2996             \__stex_terms_reset:N \l_stex_current_symbol_str
2997             \bool_set_true:N \l__stex_terms_allow_semantic_bool
2998         }
2999         \bool_set_false:N \l__stex_terms_allow_semantic_bool
3000         \use:c{stex_notation_ #1 \c_hash_str \l__stex_terms_variant_str _cs}
3001     }{
3002         \msg_error:nnxx{stex}{error/nonotation}{#1}{
3003             ~\l__stex_terms_variant_str
3004         }
3005     }
3006 }
3007
3008 \prop_new:N \l__stex_terms_custom_args_prop
3009
3010 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3011     \exp_args:Nnx \use:nn {
3012         \bool_set_false:N \l__stex_terms_allow_semantic_bool
3013         \str_set:Nn \l_stex_current_symbol_str { #1 }
3014         \prop_clear:N \l__stex_terms_custom_args_prop
3015         \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3016         \prop_put:Nnx \l__stex_terms_custom_args_prop {args} {
3017             \prop_item:cn {
3018                 l_stex_symdecl_l\l_stex_get_symbol_uri_str _prop
3019             }{ args }
3020         }
3021         \tl_set:Nn \arg { \__stex_terms_arg: }
3022         #2

```

```

3023     % TODO check that all arguments exist
3024 }{
3025   \__stex_terms_reset:N \l_stex_current_symbol_str
3026   \__stex_terms_reset:N \arg
3027   \__stex_terms_reset:N \l__stex_terms_custom_args_prop
3028   \bool_set_true:N \l__stex_terms_allow_semantic_bool
3029 }
3030 }
3031
3032 \NewDocumentCommand \__stex_terms_arg: { s O{} m}{
3033   \str_if_eq:nnTF {#2}{ }{
3034     \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
3035     \bool_set_true:N \l_tmpa_bool
3036     \bool_do_while:Nn \l_tmpa_bool {
3037       \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int}
3038       \int_incr:N \l_tmpa_int
3039     }{
3040       \bool_set_false:N \l_tmpa_bool
3041     }
3042   }
3043 }{
3044   \int_set:Nn \l_tmpa_int { #2 }
3045   \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
3046     % TODO throw error
3047   }
3048 }
3049 \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
3050 \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3051   % TODO throw error
3052 }
3053 \IfBooleanTF#1{
3054   \stex_annotate_invisible:n {
3055     \exp_args:No \_stex_term_arg:nn {\l_stex_current_symbol_str}{#3}
3056   }
3057 }{
3058   \exp_args:No \_stex_term_arg:nn {\l_stex_current_symbol_str}{#3}
3059 }
3060 }
3061
3062
3063 \cs_new_protected:Nn \_stex_term_arg:nn {
3064   \exp_args:Nnx \use:nn {
3065     \bool_set_true:N \l__stex_terms_allow_semantic_bool
3066     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3067   }{
3068     \bool_set_false:N \l__stex_terms_allow_semantic_bool
3069   }
3070 }
3071
3072 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
3073   \exp_args:Nnx \use:nn
3074   { \int_set:Nn \l__stex_terms_downprec { #2 }
3075     \_stex_term_arg:nn { #1 }{ #3 }
3076   }

```

```

3077 { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3078 }
3079
3080

```

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

## 31.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
3081 \tl_const:Nx \infprec {\int_use:N \c_max_int}
3082 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
3083 \int_new:N \l__stex_terms_downprec
3084 \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 37.)

Bracketing:

```

\l__stex_terms_left_bracket_str
\l__stex_terms_right_bracket_str
3085 \tl_set:Nn \l__stex_terms_left_bracket_str (
3086 \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

```

3087 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
3088   \bool_if:NTF \l__stex_terms_brackets_done_bool {
3089     \bool_set_false:N \l__stex_terms_brackets_done_bool
3090     #2
3091   } {
3092     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
3093       \bool_if:NTF \l__stex_inarray_bool { #2 }{
3094         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
3095         \dobrackets { #2 }
3096       }
3097     }{ #2 }
3098   }
3099 }

```

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

`\dobrackets`

```

3100 \bool_new:N \l__stex_terms_brackets_done_bool
3101 %\RequirePackage{scalerel}
3102 \cs_new_protected:Npn \dobrackets #1 {
3103   %\ThisStyle{\if D\m@switch
3104   %   \exp_args:Nnx \use:nn
3105   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
3106   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
3107   % \else
3108   \exp_args:Nnx \use:nn

```

```

3109     {
3110       \bool_set_true:N \l__stex_terms_brackets_done_bool
3111       \int_set:Nn \l__stex_terms_downprec \infprec
3112       \l__stex_terms_left_bracket_str
3113       #1
3114     }
3115     {
3116       \bool_set_false:N \l__stex_terms_brackets_done_bool
3117       \l__stex_terms_right_bracket_str
3118       \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3119     }
3120     %\fi}
3121   }

```

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

### `\withbrackets`

```

3122 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
3123   \exp_args:Nnx \use:nn
3124   {
3125     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
3126     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
3127     #3
3128   }
3129   {
3130     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
3131     {\l__stex_terms_left_bracket_str}
3132     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
3133     {\l__stex_terms_right_bracket_str}
3134   }
3135 }

```

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

### `\STEXinvisible`

```

3136 \cs_new_protected:Npn \STEXinvisible #1 {
3137   \stex_annotate_invisible:n { #1 }
3138 }

```

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

OMDoc terms:

### `\_stex_term_math_oms:nnnn`

```

3139 \cs_new_protected:Nn \_stex_term_oms:nnn {
3140   \stex_annotate:nnn{ OMID }{ #2 }{
3141     \stex_highlight_term:nn { #1 } { #3 }
3142   }
3143 }
3144
3145 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
3146   \_stex_terms_maybe_brackets:nn { #3 }{
3147     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3148   }
3149 }

```

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

`\_stex_term_math_omv:nn`

```

3150 \cs_new_protected:Nn \_stex_term_omv:nn {
3151   \stex_annotate:nnn{ OMID }{ #1 }{
3152     \stex_highlight_term:nn { #1 } { #2 }
3153   }
3154 }
```

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

`\_stex_term_math_oma:nnnn`

```

3155 \cs_new_protected:Nn \_stex_term_oma:nnn {
3156   \stex_annotate:nnn{ OMA }{ #2 }{
3157     \stex_highlight_term:nn { #1 } { #3 }
3158   }
3159 }
3160
3161 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
3162   \_stex_terms_maybe_brackets:nn { #3 }{
3163     \stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3164   }
3165 }
```

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

`\_stex_term_math_omb:nnnn`

```

3166 \cs_new_protected:Nn \_stex_term_ombind:nnn {
3167   \stex_annotate:nnn{ OMBIND }{ #2 }{
3168     \stex_highlight_term:nn { #1 } { #3 }
3169   }
3170 }
3171
3172 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
3173   \_stex_terms_maybe_brackets:nn { #3 }{
3174     \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3175   }
3176 }
```

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

`\_stex_term_math_assoc_arg:nnnn`

```

3177 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
3178   % TODO sequences
3179   \clist_set:Nn \l_tmpa_clist{ #3 }
3180   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3181     \tl_set:Nn \l_tmpa_tl { #3 }
3182   }{
3183     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3184     \clist_reverse:N \l_tmpa_clist
3185     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3186
3187     \clist_map_inline:Nn \l_tmpa_clist {
3188       \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3189         \exp_args:Nno
```

```

3190         \l_tmpa_cs { ##1 } \l_tmpa_tl
3191     }
3192 }
3193 }
3194 \exp_args:Nnno
3195   \stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
3196 }

```

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

`\stex_term_custom:nn`

```

3197 \cs_new_protected:Nn \stex_term_custom:nn {
3198   \str_set:Nn \l__stex_terms_custom_uri { #1 }
3199   \str_set:Nn \l_tmpa_str { #2 }
3200   \tl_clear:N \l_tmpa_tl
3201   \int_zero:N \l_tmpa_int
3202   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
3203   \__stex_terms_custom_loop:
3204 }

```

(End definition for `\stex_term_custom:nn`. This function is documented on page 37.)

`\__stex_terms_custom_loop:`

```

3205 \cs_new_protected:Nn \__stex_terms_custom_loop: {
3206   \bool_set_false:N \l_tmpa_bool
3207   \bool_while_do:nn {
3208     \str_if_eq_p:ee X {
3209       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
3210     }
3211   }{
3212     \int_incr:N \l_tmpa_int
3213   }
3214
3215   \peek_charcode:NTF [ {
3216     % notation/text component
3217     \__stex_terms_custom_component:w
3218   } {
3219     \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
3220       % all arguments read => finish
3221       \__stex_terms_custom_final:
3222     } {
3223       % arguments missing
3224       \peek_charcode_remove:NTF * {
3225         % invisible, specific argument position or both
3226         \peek_charcode:NTF [ {
3227           % visible specific argument position
3228           \__stex_terms_custom_arg:wn
3229         } {
3230           % invisible
3231           \peek_charcode_remove:NTF * {
3232             % invisible specific argument position
3233             \__stex_terms_custom_arg_inv:wn
3234           } {
3235             % invisible next argument
3236             \__stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]

```

```

3237     }
3238   }
3239   } {
3240     % next normal argument
3241     \__stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
3242   }
3243 }
3244 }
3245 }

```

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

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

```

3246 \cs_new_protected:Npn \__stex_terms_custom_arg_inv:wn [ #1 ] #2 {
3247   \bool_set_true:N \l_tmpa_bool
3248   \__stex_terms_custom_arg:wn [ #1 ] { #2 }
3249 }

```

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

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

```

3250 \cs_new_protected:Npn \__stex_terms_custom_arg:wn [ #1 ] #2 {
3251   \str_set:Nx \l_tmpb_str {
3252     \str_item:Nn \l_tmpa_str { #1 }
3253   }
3254   \str_case:VnTF \l_tmpb_str {
3255     { X } {
3256       \msg_error:nnx{stex}{error/notationarg}{\l__stex_terms_custom_uri}
3257     }
3258     { i } { \__stex_terms_custom_set_X:n { #1 } }
3259     { b } { \__stex_terms_custom_set_X:n { #1 } }
3260     { a } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
3261     { B } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
3262   }{}{
3263     \msg_error:nnx{stex}{error/notationarg}{\l__stex_terms_custom_uri}
3264   }
3265
3266   \bool_if:nTF \l_tmpa_bool {
3267     \tl_put_right:Nx \l_tmpa_tl {
3268       \stex_annotate_invisible:n {
3269         \stex_term_arg:nn { \int_eval:n { #1 } }
3270         \exp_not:n { { #2 } }
3271       }
3272     }
3273   } {
3274     \tl_put_right:Nx \l_tmpa_tl {
3275       \stex_term_arg:nn { \int_eval:n { #1 } }
3276       \exp_not:n { { #2 } }
3277     }
3278   }
3279
3280   \__stex_terms_custom_loop:
3281 }

```

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

`\_stex_terms_custom_set_X:n`

```

3282 \cs_new_protected:Nn \_stex_terms_custom_set_X:n {
3283   \str_set:Nx \l_tmpa_str {
3284     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
3285     X
3286     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
3287   }
3288 }

```

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

`\_stex_terms_custom_component:`

```

3289 \cs_new_protected:Npn \_stex_terms_custom_component:w [ #1 ] {
3290   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
3291   \_stex_terms_custom_loop:
3292 }

```

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

`\_stex_terms_custom_final:`

```

3293 \cs_new_protected:Nn \_stex_terms_custom_final: {
3294   \int_compare:nNnTF \l_tmpb_int = 0 {
3295     \exp_args:Nnno \_stex_term_oms:nnn
3296   }{
3297     \str_if_in:NnTF \l_tmpa_str {b} {
3298       \exp_args:Nnno \_stex_term_ombind:nnn
3299     } {
3300       \exp_args:Nnno \_stex_term_oma:nnn
3301     }
3302   }
3303   { \l_stex_terms_custom_uri } { \l_stex_terms_custom_uri } { \l_tmpa_tl }
3304 }

```

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

`\symref`

`\symname`

```

3305 \NewDocumentCommand \symref { m m }{
3306   \let\compemph_uri_prev:\compemph@uri
3307   \let\compemph@uri\symrefemph@uri
3308   \STEXsymbol{#1}![#2]
3309   \let\compemph@uri\compemph_uri_prev:
3310 }
3311
3312 \keys_define:nn { stex / symname } {
3313   post .str_set_x:N = \l_stex_symname_post_str
3314 }
3315
3316 \cs_new_protected:Nn \stex_symname_args:n {
3317   \str_clear:N \l_stex_symname_post_str
3318   \keys_set:nn { stex / symname } { #1 }
3319 }
3320
3321 \NewDocumentCommand \symname { 0{} m }{
3322   \stex_symname_args:n { #1 }

```



```

3323 \stex_get_symbol:n { #2 }
3324 \str_set:Nx \l_tmpa_str {
3325   \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3326 }
3327 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3328
3329 \let\compemph_uri_prev:\compemph@uri
3330 \let\compemph@uri\symrefemph@uri
3331 \exp_args:NNx \use:nn
3332 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }![
3333   \l_tmpa_str \l_stex_symname_post_str
3334 ] }
3335 \let\compemph@uri\compemph_uri_prev:
3336 }

```

(End definition for `\symref` and `\symname`. These functions are documented on page 36.)

### 31.3 Notation Components

```

3337 <@@=stex_notationcomps>

```

`\stex_highlight_term:nn`

```

3338
3339 \str_new:N \l_stex_current_symbol_str
3340 \cs_new_protected:Nn \stex_highlight_term:nn {
3341   \exp_args:Nnx
3342   \use:nn {
3343     \str_set:Nx \l_stex_current_symbol_str { #1 }
3344     #2
3345   } {
3346     \str_set:Nx \exp_not:N \l_stex_current_symbol_str
3347     { \l_stex_current_symbol_str }
3348   }
3349 }
3350
3351 \cs_new_protected:Nn \stex_unhighlight_term:n {
3352   % \latexml_if:TF {
3353   %   #1
3354   % } {
3355   %   \rustex_if:TF {
3356   %     #1
3357   %   } {
3358     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
3359   % }
3360   % }
3361 }

```

(End definition for `\stex_highlight_term:nn`. This function is documented on page 37.)

```

\comp
\compemph@uri
\compemph
\defemph
\defemph@uri
\symrefemph
\symrefemph@uri
3362 \cs_new_protected:Npn \comp #1 {
3363   \str_if_empty:NF \l_stex_current_symbol_str {
3364     \rustex_if:TF {
3365       \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }

```

```

3366     }{
3367       \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3368     }
3369   }
3370 }
3371
3372 \cs_new_protected:Npn \compemph@uri #1 #2 {
3373   \compemph{ #1 }
3374 }
3375
3376
3377 \cs_new_protected:Npn \compemph #1 {
3378   #1
3379 }
3380
3381 \cs_new_protected:Npn \defemph@uri #1 #2 {
3382   \defemph{#1}
3383 }
3384
3385 \cs_new_protected:Npn \defemph #1 {
3386   \textbf{#1}
3387 }
3388
3389 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3390   \symrefemph{#1}
3391 }
3392
3393 \cs_new_protected:Npn \symrefemph #1 {
3394   \textbf{#1}
3395 }

```

(End definition for `\comp` and others. These functions are documented on page 37.)

### `\ellipses`

```

3396 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix 3397 \bool_new:N \l_stex_inparray_bool
\parrayline 3398 \bool_set_false:N \l_stex_inparray_bool
\parraylineh 3399 \NewDocumentCommand \parray { m m } {
\parraycell 3400   \begin{group}
3401     \bool_set_true:N \l_stex_inparray_bool
3402     \begin{array}{#1}
3403       #2
3404     \end{array}
3405   \end{group}
3406 }
3407
3408 \NewDocumentCommand \prmatrix { m } {
3409   \begin{group}
3410     \bool_set_true:N \l_stex_inparray_bool
3411     \begin{matrix}
3412       #1

```

```

3413 \end{matrix}
3414 \endgroup
3415 }
3416
3417 \def \maybepline {
3418   \bool_if:NT \l_stex_inarray_bool {\hline}
3419 }
3420
3421 \def \parrayline #1 #2 {
3422   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3423 }
3424
3425 \def \pmrow #1 { \parrayline{}{ #1 } }
3426
3427 \def \parraylineh #1 #2 {
3428   #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3429 }
3430
3431 \def \parraycell #1 {
3432   #1 \bool_if:NT \l_stex_inarray_bool {&}
3433 }

```

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

## 31.4 Variables

```

3434 \@@=stex_variables)

```

\stex\_invoke\_variable:n Invokes a variable

```

3435 \cs_new_protected:Nn \stex_invoke_variable:n {
3436   \if_mode_math:
3437     \exp_after:wN \__stex_variables_invoke_math:n
3438   \else:
3439     \exp_after:wN \__stex_variables_invoke_text:n
3440   \fi: {#1}
3441 }
3442
3443 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
3444   %TODO
3445 }
3446
3447
3448 \cs_new_protected:Nn \__stex_variables_invoke_math:n {
3449   \peek_charcode_remove:NTF ! {
3450     \peek_charcode_remove:NTF ! {
3451       \peek_charcode:NTF [ {
3452         \__stex_variables_invoke_op_custom:nw
3453       }{
3454         % TODO throw error
3455       }
3456     }{
3457       \__stex_variables_invoke_op:n { #1 }
3458     }
3459   }{

```

```

3460 \peek_charcode_remove:NTF * {
3461   \__stex_variables_invoke_text:n { #1 }
3462 }{
3463   \__stex_variables_invoke_math_ii:n { #1 }
3464 }
3465 }
3466 }
3467
3468 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
3469   \cs_if_exist:cTF {
3470     stex_var_op_notation_ #1 _cs
3471   }{
3472     \use:c{stex_var_op_notation_ #1 _cs }
3473   }{
3474     \msg_error:nxxx{stex}{error/noop}{variable~#1}{}
3475   }
3476 }
3477
3478 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n #1 {
3479   \cs_if_exist:cTF {
3480     stex_var_notation_#1_cs
3481   }{
3482     \str_set:Nn \l_stex_current_symbol_str { #1 }
3483     \use:c{stex_var_notation_#1_cs}
3484   }{
3485     \msg_error:nxxx{stex}{error/nonotation}{variable~#1}{s}
3486   }
3487 }

```

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

```

3488 </package>

```

## Chapter 32

# STEX -Structural Features Implementation

```
3489 <*package>
3490
3491 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3492
3493 <@@=stex_features>
3494
3495   Warnings and error messages
3496 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3497   Symbol~#1~can~not~be~assigned~in~copymodule~#2
3498 }
3499 \msg_new:nnn{stex}{error/interpretmodule/noddefinies}{
3500   Symbol~#1~not~assigned~in~interpretmodule~#2
3501 }
```

### 32.1 Imports with modification

```
3501 \cs_new_protected:Nn \stex_get_symbol_in_copymodule:n {
3502   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3503     \__stex_features_get_symbol_from_cs:n { #1 }
3504   }{
3505     % argument is a string
3506     % is it a command name?
3507     \cs_if_exist:cTF { #1 }{
3508       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3509       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3510       \str_if_empty:NNTF \l_tmpa_str {
3511         \exp_args:Nx \cs_if_eq:NNTF {
3512           \tl_head:N \l_tmpa_tl
3513         } \stex_invoke_symbol:n {
3514           \exp_args:No \__stex_features_get_symbol_from_cs:n { \use:c { #1 } }
3515         }{
3516           \__stex_features_get_symbol_from_string:n { #1 }
```

```

3517     }
3518   } {
3519     \__stex_features_get_symbol_from_string:n { #1 }
3520   }
3521   ){
3522     % argument is not a command name
3523     \__stex_features_get_symbol_from_string:n { #1 }
3524     % \l_stex_all_symbols_seq
3525   }
3526 }
3527 }
3528
3529 \cs_new_protected:Nn \__stex_features_get_symbol_from_string:n {
3530   \str_set:Nn \l_tmpa_str { #1 }
3531   \bool_set_false:N \l_tmpa_bool
3532   \bool_if:NF \l_tmpa_bool {
3533     \tl_set:Nn \l_tmpa_tl {
3534       \msg_set:nnn{stex}{error/unknownsymbol}{
3535         No~symbol~#1~found!
3536       }
3537       \msg_error:nn{stex}{error/unknownsymbol}
3538     }
3539     \str_set:Nn \l_tmpa_str { #1 }
3540     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3541     \seq_map_inline:Nn \l__stex_features_copymodule_fields_seq {
3542       \str_set:Nn \l_tmpb_str { ##1 }
3543       \str_if_eq:eeT { \l_tmpa_str } {
3544         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3545       } {
3546         \seq_map_break:n {
3547           \tl_set:Nn \l_tmpa_tl {
3548             \str_set:Nn \l_stex_get_symbol_uri_str {
3549               ##1
3550             }
3551             \__stex_features_get_symbol_check:
3552           }
3553         }
3554       }
3555     }
3556     \l_tmpa_tl
3557   }
3558 }
3559
3560 \cs_new_protected:Nn \__stex_features_get_symbol_from_cs:n {
3561   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3562   { \tl_tail:N \l_tmpa_tl }
3563   \tl_if_single:NTF \l_tmpa_tl {
3564     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3565       \exp_after:wN \str_set:Nn \exp_after:wN
3566       \l_stex_get_symbol_uri_str \l_tmpa_tl
3567       \__stex_features_get_symbol_check:
3568     }{
3569       % TODO
3570       % tail is not a single group

```

```

3571     }
3572   }{
3573     % TODO
3574     % tail is not a single group
3575   }
3576 }
3577
3578 \cs_new_protected:Nn \__stex_features_get_symbol_check: {
3579   \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq {?} \l_stex_get_symbol_uri_str
3580   \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 3 {
3581     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
3582     \str_set:Nx \l_tmpa_str {\seq_use:Nn \l_tmpa_seq ?}
3583     \seq_if_in:NoF \l__stex_features_copymodule_modules_seq \l_tmpa_str {
3584       \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3585         \l_stex_current_copymodule_name_str\Allowed:~\seq_use:Nn \l__stex_features_copymodule_modules_seq \l_tmpa_str
3586       }
3587     }
3588   }{
3589     \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3590       \l_stex_current_copymodule_name_str~(inexplicably)
3591     }
3592   }
3593 }
3594
3595 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
3596   \stex_import_module_uri:nn { #1 } { #2 }
3597   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3598   \stex_import_require_module:nnnn
3599     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3600     { \l_stex_import_path_str } { \l_stex_import_name_str }
3601   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3602   \seq_set_eq:NN \l__stex_features_copymodule_modules_seq \l_stex_collect_imports_seq
3603   \seq_clear:N \l__stex_features_copymodule_fields_seq
3604   \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3605     \seq_map_inline:cn {c_stex_module_###1_constants}{
3606       \exp_args:NNx \seq_put_right:Nn \l__stex_features_copymodule_fields_seq {
3607         ###1 ? ####1
3608       }
3609     }
3610   }
3611   \seq_clear:N \l_tmpa_seq
3612   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3613     name      = \l_stex_current_copymodule_name_str ,
3614     module    = \l_stex_current_module_str ,
3615     from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3616     includes  = \l_tmpa_seq ,
3617     fields    = \l_tmpa_seq
3618   }
3619   \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3620     as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3621   \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_features_copymodule_modules_seq
3622     \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_features_copymodule_fields_seq {,~}
3623   \stex_if_smsmode:F {
3624     \begin{stex_annotate_env} {#4} {

```

```

3625     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3626   }
3627   \stex_annotate_invisible:nnn{from}{\l_stex_import_ns_str ?\l_stex_import_name_str}{\}
3628 }
3629 \bool_set_eq:NN \l__stex_features_oldhtml_bool \stex_html_do_output_bool
3630 \bool_set_false:N \stex_html_do_output_bool
3631 }
3632 \cs_new_protected:Nn \stex_copymodule_end:n {
3633   \def \l_tmpa_cs ##1 ##2 {#1}
3634   \bool_set_eq:NN \stex_html_do_output_bool \l__stex_features_oldhtml_bool
3635   \tl_clear:N \l_tmpa_tl
3636   \tl_clear:N \l_tmpb_tl
3637   \prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3638   \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3639     \seq_map_inline:cn {c_stex_module_##1_constants}{
3640       \tl_clear:N \l_tmpc_tl
3641       \l_tmpa_cs{##1}{####1}
3642       \str_if_exist:cTF {\l__stex_features_copymodule_##1?####1_name_str} {
3643         \tl_put_right:Nx \l_tmpa_tl {
3644           \prop_set_from_keyval:cn {
3645             l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_#
3646           }{
3647             \exp_after:wN \prop_to_keyval:N \csname
3648               l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_#
3649             \endcsname
3650           }
3651           \seq_clear:c {
3652             l_stex_symdecl_
3653             \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_name_s
3654             _notations
3655           }
3656         }
3657         \tl_put_right:Nx \l_tmpc_tl {
3658           \stex_copy_notations:nn {\l_stex_current_module_str ? \use:c{l__stex_features_copy
3659           \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_features_copymodule_##1?####1_n
3660         }
3661         \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{l__stex_features_
3662         \str_if_exist:cT {\l_stex_features_copymodule_##1?####1_macroname_str} {
3663           \tl_put_right:Nx \l_tmpc_tl {
3664             \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?#
3665           }
3666           \tl_put_right:Nx \l_tmpa_tl {
3667             \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3668             \stex_invoke_symbol:n {
3669               \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_na
3670             }
3671           }
3672         }
3673       }
3674     }{
3675       \tl_put_right:Nx \l_tmpc_tl {
3676         \stex_copy_notations:nn {\l_stex_current_module_str ? \l_stex_current_copymodule_n
3677       }
3678       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1_prop}

```



```

3679 \prop_put:Nnx \l_tmpa_prop { name }{ \l_stex_current_copymodule_name_str / ####1 }
3680 \prop_put:Nnx \l_tmpa_prop { module }{ \l_stex_current_module_str }
3681 \tl_put_right:Nx \l_tmpa_tl {
3682   \prop_set_from_keyval:cn {
3683     l_stex_symdecl \l_stex_current_module_str ? \l_stex_current_copymodule_name_str
3684   }{
3685     \prop_to_keyval:N \l_tmpa_prop
3686   }
3687   \seq_clear:c {
3688     l_stex_symdecl_
3689     \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3690     _notations
3691   }
3692 }
3693 \seq_put_right:Nx \l_tmpa_seq { \l_stex_current_module_str ? \l_stex_current_copymodule
3694 \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3695   \tl_put_right:Nx \l_tmpc_tl {
3696     \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?#
3697   }
3698   \tl_put_right:Nx \l_tmpa_tl {
3699     \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3700       \stex_invoke_symbol:n {
3701         \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3702       }
3703     }
3704   }
3705 }
3706 }
3707 \tl_if_exist:cT {l__stex_features_copymodule_##1?####1_def_tl}{
3708   \tl_put_right:Nx \l_tmpc_tl {
3709     \stex_annotate_invisible:nnn{definiens}{\use:c{l__stex_features_copymodule_##1?#
3710   }
3711 }
3712 \tl_put_right:Nx \l_tmpb_tl {
3713   \stex_annotate:nnn{assignment} {##1?####1} { \l_tmpc_tl }
3714 }
3715 }
3716 }
3717 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3718 \tl_put_left:Nx \l_tmpa_tl {
3719   \prop_set_from_keyval:cn {
3720     l_stex_copymodule_ \l_stex_current_module_str ? \l_stex_current_copymodule_name_str _pro
3721   }{
3722     \prop_to_keyval:N \l_stex_current_copymodule_prop
3723   }
3724 }
3725 \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl
3726 \stex_debug:nn{copymodule}{result:\meaning \l_tmpa_tl}
3727 \exp_args:Nx \stex_do_up_to_module:n {
3728   \exp_args:No \exp_not:n \l_tmpa_tl
3729 }
3730 \l_tmpb_tl
3731 \stex_if_smsmode:F {
3732   \end{stex_annotate_env}

```

```

3733 }
3734 }
3735
3736 \NewDocumentEnvironment {copymodule} { 0{} m m}{
3737   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ structure }
3738   \stex_deactivate_macro:Nn \symdecl {module~environments}
3739   \stex_deactivate_macro:Nn \symdef {module~environments}
3740   \stex_deactivate_macro:Nn \notation {module~environments}
3741   \stex_reactivate_macro:N \assign
3742   \stex_reactivate_macro:N \renamedec1
3743   \stex_reactivate_macro:N \donotcopy
3744   \stex_smsmode_do:
3745 }{
3746   \stex_copymodule_end:n {}
3747 }
3748
3749 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
3750   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ realization }
3751   \stex_deactivate_macro:Nn \symdecl {module~environments}
3752   \stex_deactivate_macro:Nn \symdef {module~environments}
3753   \stex_deactivate_macro:Nn \notation {module~environments}
3754   \stex_reactivate_macro:N \assign
3755   \stex_reactivate_macro:N \renamedec1
3756   \stex_reactivate_macro:N \donotcopy
3757   \stex_smsmode_do:
3758 }{
3759   \stex_copymodule_end:n {
3760     \tl_if_exist:cF {
3761       l__stex_features_copymodule_##1?##2_def_tl
3762     }{
3763       \msg_error:nnxx{stex}{error/interpretmodule/nodéfiniens}{
3764         ##1?##2
3765       }{\l_stex_current_copymodule_name_str}
3766     }
3767   }
3768 }
3769
3770 \NewDocumentCommand \donotcopy { 0{} m}{
3771   \stex_import_module_uri:nn { #1 } { #2 }
3772   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3773   \seq_map_inline:Nn \l_stex_collect_imports_seq {
3774     \seq_remove_all:Nn \l_stex_features_copymodule_modules_seq { ##1 }
3775     \seq_map_inline:cn {c_stex_module_##1_constants}{
3776       \seq_remove_all:Nn \l_stex_features_copymodule_fields_seq { ##1 ? #####1 }
3777       \bool_lazy_any_p:nT {
3778         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_name_str}}
3779         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_macroname_str}}
3780         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_def_tl}}
3781       }{
3782         % TODO throw error
3783       }
3784     }
3785   }
3786 }

```

```

3787 \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
3788 \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_ns_str ?\l_stex_import_name_str }
3789 \prop_put:Nnx \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
3790 }
3791
3792 \NewDocumentCommand \assign { m m }{
3793   \stex_get_symbol_in_copymodule:n {#1}
3794   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
3795   \tl_set:cn {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _def_tl}{#2}
3796 }
3797
3798 \keys_define:nn { stex / renamedec1 } {
3799   name .str_set_x:N = \l_stex_renamedec1_name_str
3800 }
3801 \cs_new_protected:Nn \__stex_features_renamedec1_args:n {
3802   \str_clear:N \l_stex_renamedec1_name_str
3803
3804   \keys_set:nn { stex / renamedec1 } { #1 }
3805 }
3806
3807 \NewDocumentCommand \renamedec1 { O{} m m }{
3808   \__stex_features_renamedec1_args:n { #1 }
3809   \stex_get_symbol_in_copymodule:n {#2}
3810   \stex_debug:nn{renamedec1}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
3811   \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
3812   \str_if_empty:NTF \l_stex_renamedec1_name_str {
3813     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3814       \l_stex_get_symbol_uri_str
3815     } }
3816   } {
3817     \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex_r
3818     \stex_debug:nn{renamedec1}{@~\l_stex_current_module_str ? \l_stex_renamedec1_name_str}
3819     \prop_set_eq:cc {l_stex_symdec1_
3820       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3821       _prop
3822     }{l_stex_symdec1_ \l_stex_get_symbol_uri_str _prop}
3823     \seq_set_eq:cc {l_stex_symdec1_
3824       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3825       _notations
3826     }{l_stex_symdec1_ \l_stex_get_symbol_uri_str _notations}
3827     \prop_put:cnx {l_stex_symdec1_
3828       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3829       _prop
3830     }{ name }{ \l_stex_renamedec1_name_str }
3831     \prop_put:cnx {l_stex_symdec1_
3832       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3833       _prop
3834     }{ module }{ \l_stex_current_module_str }
3835     \exp_args:NNx \seq_put_left:Nn \l__stex_features_copymodule_fields_seq {
3836       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3837     }
3838     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3839       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3840     } }

```

```

3841 }
3842 }
3843 %\NewDocumentCommand \notation_in_copymodules: { 0{} m } {
3844 % \_stex_notation_args:n { #1 }
3845 % \tl_clear:N \l_stex_symdecl_definiens_tl
3846 % \stex_get_symbol_in_copymodule:n { #2 }
3847 % \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
3848 % % todo
3849 %}
3850 \stex_deactivate_macro:Nn \assign {copymodules}
3851 \stex_deactivate_macro:Nn \renamedekl {copymodules}
3852 \stex_deactivate_macro:Nn \donotcopy {copymodules}
3853
3854
3855 \seq_new:N \l_stex_implicit_morphisms_seq
3856 \NewDocumentCommand \implicitmorphism { 0{} m m }{
3857 \stex_import_module_uri:nn { #1 } { #2 }
3858 \stex_debug:nn{implicits}{
3859 Implicit~morphism:~
3860 \l_stex_module_ns_str ? \l__stex_features_name_str
3861 }
3862 \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
3863 \l_stex_module_ns_str ? \l__stex_features_name_str
3864 }{
3865 \msg_error:nnn{stex}{error/conflictingmodules}{
3866 \l_stex_module_ns_str ? \l__stex_features_name_str
3867 }
3868 }
3869
3870 % TODO
3871
3872
3873
3874 \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
3875 \l_stex_module_ns_str ? \l__stex_features_name_str
3876 }
3877 }
3878

```

## 32.2 The feature environment

structural@feature

```

3879
3880 \NewDocumentEnvironment{structural@feature}{ m m m }{
3881 \stex_if_in_module:F {
3882 \msg_set:nnn{stex}{error/nomodule}{
3883 Structural~Feature~has~to~occur~in~a~module:\\
3884 Feature~#2~of~type~#1\\
3885 In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
3886 }
3887 \msg_error:nn{stex}{error/nomodule}
3888 }
3889

```

```

3890 \str_set:Nx \l_stex_module_name_str {
3891   \prop_item:Nn \l_stex_current_module_prop
3892     { name } / #2 - feature
3893 }
3894
3895 \str_set:Nx \l_stex_module_ns_str {
3896   \prop_item:Nn \l_stex_current_module_prop
3897     { ns }
3898 }
3899
3900
3901 \str_clear:N \l_tmpa_str
3902 \seq_clear:N \l_tmpa_seq
3903 \tl_clear:N \l_tmpa_tl
3904 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
3905   origname = #2,
3906   name     = \l_stex_module_name_str ,
3907   ns       = \l_stex_module_ns_str ,
3908   imports  = \exp_not:o { \l_tmpa_seq } ,
3909   constants = \exp_not:o { \l_tmpa_seq } ,
3910   content  = \exp_not:o { \l_tmpa_tl } ,
3911   file     = \exp_not:o { \g_stex_currentfile_seq } ,
3912   lang     = \l_stex_module_lang_str ,
3913   sig      = \l_tmpa_str ,
3914   meta     = \l_tmpa_str ,
3915   feature  = #1 ,
3916 }
3917
3918 \stex_if_smsmode:F {
3919   \begin{stex_annotate_env}{ feature:#1 }{}
3920   \stex_annotate_invisible:nnn{header}{}{ #3 }
3921 }
3922 }{
3923   \str_set:Nx \l_tmpa_str {
3924     c_stex_feature_
3925     \prop_item:Nn \l_stex_current_module_prop { ns } ?
3926     \prop_item:Nn \l_stex_current_module_prop { name }
3927     _prop
3928   }
3929   \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
3930   \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
3931   \stex_if_smsmode:F {
3932     \end{stex_annotate_env}
3933   }
3934 }
3935

```

## 32.3 Features

structure

```

3936
3937 \prop_new:N \l_stex_all_structures_prop
3938

```

```

3939 \keys_define:nn { stex / features / structure } {
3940   name          .str_set_x:N = \l__stex_features_structure_name_str ,
3941 }
3942
3943 \cs_new_protected:Nn \__stex_features_structure_args:n {
3944   \str_clear:N \l__stex_features_structure_name_str
3945   \keys_set:nn { stex / features / structure } { #1 }
3946 }
3947
3948 %\stex_new_feature:nnnn { structure } { 0{ } m } {
3949 %   \__stex_features_structure_args:n { ##1 }
3950 %   \str_if_empty:NT \l__stex_features_structure_name_str {
3951 %     \str_set:Nx \l__stex_features_structure_name_str { ##2 }
3952 %   }
3953 % } {
3954 %
3955 %}
3956
3957 \NewDocumentEnvironment{mathstructure}{ 0{ } m }{
3958   \__stex_features_structure_args:n { #1 }
3959   \str_if_empty:NT \l__stex_features_structure_name_str {
3960     \str_set:Nx \l__stex_features_structure_name_str { #2 }
3961   }
3962   \exp_args:Nnnx
3963   \begin{structural@feature}{ structure }
3964     { \l__stex_features_structure_name_str }{}
3965     \seq_clear:N \l_tmpa_seq
3966     \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
3967     \stex_smsmode_do:
3968   }{
3969     \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
3970     \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
3971     \str_set:Nx \l_tmpa_str {
3972       \prop_item:Nn \l_stex_current_module_prop { ns } ?
3973       \prop_item:Nn \l_stex_current_module_prop { name }
3974     }
3975     \seq_map_inline:Nn \l_tmpa_seq {
3976       \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
3977     }
3978     \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
3979     \exp_args:Nnx
3980     \AddToHookNext { env / mathstructure / after }{
3981       \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}}{
3982         \stex_term_math_oms:nnnn { \l_tmpa_str }{}{0}{}}
3983       }, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
3984       \STEXexport {
3985         \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3986           {\prop_item:Nn \l_stex_current_module_prop { origname }}
3987           {\l_tmpa_str}
3988         \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3989           {#2}{\l_tmpa_str}
3990       %   \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3991       %     \prop_item:Nn \l_stex_current_module_prop { origname },
3992       %     \l_tmpa_str

```

```

3993 %      }
3994 %      \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3995 %          #2,\l_tmpa_str
3996 %      }
3997 %      \tl_set:cx { #2 } {
3998 %          \stex_invoke_structure:n { \l_tmpa_str }
3999 %      }
4000 %  }
4001
4002 \end{structural@feature}
4003 % \g_stex_last_feature_prop
4004 }

```

\instantiate

```

4005 \seq_new:N \l__stex_features_structure_field_seq
4006 \str_new:N \l__stex_features_structure_field_str
4007 \str_new:N \l__stex_features_structure_def_tl
4008 \prop_new:N \l__stex_features_structure_prop
4009 \NewDocumentCommand \instantiate { m O{} m }{
4010     \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
4011     \prop_set_eq:Nc \l__stex_features_structure_prop {
4012         c_stex_feature_\l_tmpa_str _prop
4013     }
4014     \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
4015     \seq_map_inline:Nn \l__stex_features_structure_field_seq {
4016         \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
4017         \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
4018             \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
4019             \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
4020                 {!} \l_tmpa_tl
4021             \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
4022                 \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
4023                 \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
4024                 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
4025             }{
4026                 \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
4027                 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
4028                 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
4029                     \l_tmpa_tl
4030                 \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
4031                     \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
4032                     \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
4033                 }{
4034                     \tl_clear:N \l_tmpb_tl
4035                 }
4036             }
4037         }{
4038             \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
4039             \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
4040                 \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
4041                 \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
4042                 \tl_clear:N \l_tmpa_tl
4043             }{
4044                 % TODO throw error

```

```

4045     }
4046   }
4047   % \l_tmpa_str: name
4048   % \l_tmpa_tl: definiens
4049   % \l_tmpb_tl: notation
4050   \tl_if_empty:NT \l__stex_features_structure_field_str {
4051     % TODO throw error
4052   }
4053   \str_clear:N \l_tmpb_str
4054
4055   \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
4056   \seq_map_inline:Nn \l_tmpa_seq {
4057     \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
4058     \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
4059     \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
4060       \seq_map_break:n {
4061         \str_set:Nn \l_tmpb_str { ####1 }
4062       }
4063     }
4064   }
4065   \prop_get:cnN { l_stex_symdecl_ \l_tmpb_str _prop } {args}
4066   \l_tmpb_str
4067
4068   \tl_if_empty:NTF \l_tmpb_tl {
4069     \tl_if_empty:NF \l_tmpa_tl {
4070       \exp_args:Nx \use:n {
4071         \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fe
4072       }
4073     }
4074   }{
4075     \tl_if_empty:NTF \l_tmpa_tl {
4076       \exp_args:Nx \use:n {
4077         \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\
4078       }
4079     }{
4080       \exp_args:Nx \use:n {
4081         \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
4082         \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
4083       }
4084     }
4085   }
4086 }
4087 % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
4088 % \prop_item:Nn \l_stex_current_module_prop {name} ?
4089 % #3/\l__stex_features_structure_field_str
4090 % \par
4091 % \expandafter\present\csname
4092 %   l_stex_symdecl_
4093 %   \prop_item:Nn \l_stex_current_module_prop {ns} ?
4094 %   \prop_item:Nn \l_stex_current_module_prop {name} ?
4095 %   #3/\l__stex_features_structure_field_str
4096 %   _prop
4097 % \endcsname
4098 }

```



```

4099
4100 \tl_clear:N \l__stex_features_structure_def_tl
4101
4102 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
4103 \seq_map_inline:Nn \l_tmpa_seq {
4104   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
4105   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
4106   \exp_args:Nx \use:n {
4107     \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
4108
4109     }
4110   }
4111
4112   \prop_if_exist:cF {
4113     l_stex_symdecl_
4114     \prop_item:Nn \l_stex_current_module_prop {ns} ?
4115     \prop_item:Nn \l_stex_current_module_prop {name} ?
4116     #3/\l_tmpa_str
4117     _prop
4118   }{
4119     \prop_get:cnN { l_stex_symdecl_ ##1 _prop } {args}
4120     \l_tmpb_str
4121     \exp_args:Nx \use:n {
4122       \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
4123     }
4124   }
4125 }
4126
4127 \symdecl*[type={\STEXsymbol{module-type}}{
4128   \_stex_term_math_oms:nnnn {
4129     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
4130     \prop_item:Nn \l__stex_features_structure_prop {name}
4131     }{}{0}{}
4132   }}{#3}
4133
4134 % TODO: -> sms file
4135
4136 \tl_set:cx{ #3 }{
4137   \stex_invoke_structure:nnn {
4138     \prop_item:Nn \l_stex_current_module_prop {ns} ?
4139     \prop_item:Nn \l_stex_current_module_prop {name} ? #3
4140   } {
4141     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
4142     \prop_item:Nn \l__stex_features_structure_prop {name}
4143   }
4144 }
4145 \stex_smsmode_do:
4146 }

```

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

\stex\_invoke\_structure:nnn

```

4147 % #1: URI of the instance
4148 % #2: URI of the instantiated module

```

```

4149 \cs_new_protected:Nn \stex_invoke_structure:nnn {
4150   \tl_if_empty:nTF{ #3 }{
4151     \prop_set_eq:Nc \l__stex_features_structure_prop {
4152       c_stex_feature_ #2 _prop
4153     }
4154     \tl_clear:N \l_tmpa_tl
4155     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
4156     \seq_map_inline:Nn \l_tmpa_seq {
4157       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
4158       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
4159       \cs_if_exist:cT {
4160         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
4161       }{
4162         \tl_if_empty:NF \l_tmpa_tl {
4163           \tl_put_right:Nn \l_tmpa_tl {,}
4164         }
4165         \tl_put_right:Nx \l_tmpa_tl {
4166           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
4167         }
4168       }
4169     }
4170     \exp_args:No \mathstrut \l_tmpa_tl
4171   }{
4172     \stex_invoke_symbol:n{#1/#3}
4173   }
4174 }

```

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

```

4175 </package>

```

## Chapter 33

# STEX -Statements Implementation

```
4176 <*package>
4177
4178 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4179
4180 <@@=stex_statements>
    Warnings and error messages
4181
\titleemph
4182 \def\titleemph#1{\textbf{#1}}
    (End definition for \titleemph. This function is documented on page ??.)
```

### 33.1 Definitions

```
definiendum
4183 \keys_define:nn {stex / definiendum }{
4184   post      .tl_set:N      = \l__stex_statements_definiendum_post_tl,
4185   root      .str_set_x:N   = \l__stex_statements_definiendum_root_str,
4186   gfa       .str_set_x:N   = \l__stex_statements_definiendum_gfa_str
4187 }
4188 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
4189   \str_clear:N \l__stex_statements_definiendum_root_str
4190   \tl_clear:N \l__stex_statements_definiendum_post_tl
4191   \str_clear:N \l__stex_statements_definiendum_gfa_str
4192   \keys_set:nn { stex / definiendum }{ #1 }
4193 }
4194 \NewDocumentCommand \definiendum { 0{} m m } {
4195   \__stex_statements_definiendum_args:n { #1 }
4196   \stex_get_symbol:n { #2 }
4197   \stex_ref_new_sym_target:n \l__stex_get_symbol_uri_str
4198   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
4199     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
4200       \tl_set:Nn \l_tmpa_tl { #3 }

```

```

4201 } {
4202   \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
4203   \tl_set:Nn \l_tmpa_tl {
4204     \l__stex_statements_definiendum_root_str\l__stex_statements_definiendum_post_tl
4205   }
4206 }
4207 } {
4208   \tl_set:Nn \l_tmpa_tl { #3 }
4209 }
4210
4211 % TODO root
4212 \rustex_if:TF {
4213   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
4214 } {
4215   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
4216 }
4217 }
4218 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

**definame**

```

4219
4220 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
4221
4222 \NewDocumentCommand \definame { 0{ } m } {
4223   \__stex_statements_definiendum_args:n { #1 }
4224   % TODO: root
4225   \stex_get_symbol:n { #2 }
4226   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4227   \str_set:Nx \l_tmpa_str {
4228     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4229   }
4230   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4231   \rustex_if:TF {
4232     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4233       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4234     }
4235   } {
4236     \defemph@uri {
4237       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4238     } { \l_stex_get_symbol_uri_str }
4239   }
4240 }
4241 \stex_deactivate_macro:Nn \definame {definition~environments}
4242
4243 \NewDocumentCommand \Definame { 0{ } m } {
4244   \__stex_statements_definiendum_args:n { #1 }
4245   \stex_get_symbol:n { #2 }
4246   \str_set:Nx \l_tmpa_str {
4247     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4248   }
4249   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4250   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str

```

```

4251 \rustex_if:TF {
4252   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4253     \l_tmpa_str\l__stex_statements_definiendum_post_tl
4254   }
4255 } {
4256   \defemph@uri {
4257     \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
4258   } { \l_stex_get_symbol_uri_str }
4259 }
4260 }
4261 \stex_deactivate_macro:Nn \Definame {definition-environments}
4262
4263 \NewDocumentCommand \Symname { 0{ } m }{
4264   \stex_symname_args:n { #1 }
4265   \stex_get_symbol:n { #2 }
4266   \str_set:Nx \l_tmpa_str {
4267     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4268   }
4269   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4270   \let\compemph_uri_prev:\compemph@uri
4271   \let\compemph@uri\symrefemph@uri
4272   \exp_args:NNx \use:nn
4273   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }![
4274     \exp_after:wN \stex_capitalize:n \l_tmpa_str
4275     \l_stex_symname_post_str
4276   ] }
4277   \let\compemph@uri\compemph_uri_prev:
4278 }

```

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

#### sdefinition

```

4279
4280 \keys_define:nn {stex / sdefinition }{
4281   type      .str_set_x:N = \sdefinitiontype,
4282   id        .str_set_x:N = \sdefinitionid,
4283   name      .str_set_x:N = \sdefinitionname,
4284   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4285   title     .tl_set:N     = \sdefinitiontitle
4286 }
4287 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
4288   \str_clear:N \sdefinitiontype
4289   \str_clear:N \sdefinitionid
4290   \str_clear:N \sdefinitionname
4291   \clist_clear:N \l__stex_statements_sdefinition_for_clist
4292   \tl_clear:N \sdefinitiontitle
4293   \keys_set:nn { stex / sdefinition }{ #1 }
4294 }
4295
4296 \NewDocumentEnvironment{sdefinition}{0{}}{
4297   \__stex_statements_sdefinition_args:n{ #1 }
4298   \stex_reactivate_macro:N \definiendum
4299   \stex_reactivate_macro:N \definame
4300   \stex_reactivate_macro:N \Definame

```

```

4301 \stex_if_smsmode:F{
4302   \seq_clear:N \l_tmpa_seq
4303   \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4304     \str_if_eq:nnF{ ##1 }{}{
4305       \stex_get_symbol:n { ##1 }
4306       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4307         \l_stex_get_symbol_uri_str
4308       }
4309     }
4310   }
4311   \exp_args:Nnnx
4312   \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {,}}
4313   \str_if_empty:NF \sdefinitiontype {
4314     \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{}
4315   }
4316   \clist_set:No \l_tmpa_clist \sdefinitiontype
4317   \tl_clear:N \l_tmpa_tl
4318   \clist_map_inline:Nn \l_tmpa_clist {
4319     \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
4320       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
4321     }
4322   }
4323   \tl_if_empty:NTF \l_tmpa_tl {
4324     \__stex_statements_sdefinition_start:
4325   }{
4326     \l_tmpa_tl
4327   }
4328 }
4329 \stex_ref_new_doc_target:n \sdefinitionid
4330 \stex_smsmode_do:
4331 }{
4332   \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
4333   \stex_if_smsmode:F {
4334     \clist_set:No \l_tmpa_clist \sdefinitiontype
4335     \tl_clear:N \l_tmpa_tl
4336     \clist_map_inline:Nn \l_tmpa_clist {
4337       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
4338         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
4339       }
4340     }
4341     \tl_if_empty:NTF \l_tmpa_tl {
4342       \__stex_statements_sdefinition_end:
4343     }{
4344       \l_tmpa_tl
4345     }
4346     \end{stex_annotate_env}
4347   }
4348 }

```

\stexpatchdefinition

```

4349 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
4350   \par\noindent\titllemph{Definition\tl_if_empty:NF \sdefinitiontitle {
4351     ~(\sdefinitiontitle)
4352   }~}

```

```

4353 }
4354 \cs_new_protected:Nn \__stex_statements_sdefinition_end: { \par \medskip }
4355
4356 \newcommand\stexpatchdefinition[3] [] {
4357   \str_set:Nx \l_tmpa_str{ #1 }
4358   \str_if_empty:NTF \l_tmpa_str {
4359     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
4360     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
4361   }{
4362     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
4363     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
4364   }
4365 }

```

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

\inlinedef inline:

```

4366 \keys_define:nn {stex / inlinedef }{
4367   type      .str_set_x:N = \sdefinitiontype,
4368   id        .str_set_x:N = \sdefinitionid,
4369   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4370   name      .str_set_x:N = \sdefinitionname
4371 }
4372 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
4373   \str_clear:N \sdefinitiontype
4374   \str_clear:N \sdefinitionid
4375   \str_clear:N \sdefinitionname
4376   \clist_clear:N \l__stex_statements_sdefinition_for_clist
4377   \keys_set:nn { stex / inlinedef }{ #1 }
4378 }
4379 \NewDocumentCommand \inlinedef { 0{} m } {
4380   \begin{group}
4381     \__stex_statements_inlinedef_args:n{ #1 }
4382     \stex_reactivate_macro:N \definiendum
4383     \stex_reactivate_macro:N \definame
4384     \stex_reactivate_macro:N \Definame
4385     \stex_ref_new_doc_target:n \sdefinitionid
4386     \stex_if_smsmode:TF{
4387       \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
4388     }{
4389       \seq_clear:N \l_tmpa_seq
4390       \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4391         \str_if_eq:nnF{ ##1 }{}{
4392           \stex_get_symbol:n { ##1 }
4393           \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4394             \l_stex_get_symbol_uri_str
4395           }
4396         }
4397       }
4398       \exp_args:Nnx
4399       \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
4400         \str_if_empty:NF \sdefinitiontype {
4401           \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{
4402

```

```

4403     #2
4404     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{ }\sdefinitionname } }
4405   }
4406 }
4407 \endgroup
4408 \stex_smsmode_do:
4409 }

```

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

## 33.2 Assertions

sassertion

```

4410
4411 \keys_define:nn {stex / sassertion }{
4412   type      .str_set_x:N = \sassertiontype,
4413   id        .str_set_x:N = \sassertionid,
4414   title     .tl_set:N    = \sassertiontitle ,
4415   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4416   name      .str_set_x:N = \sassertionname
4417 }
4418 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
4419   \str_clear:N \sassertiontype
4420   \str_clear:N \sassertionid
4421   \str_clear:N \sassertionname
4422   \clist_clear:N \l__stex_statements_sassertion_for_clist
4423   \tl_clear:N \sassertiontitle
4424   \keys_set:nn { stex / sassertion }{ #1 }
4425 }
4426
4427 %\tl_new:N \g__stex_statements_aftergroup_tl
4428
4429 \NewDocumentEnvironment{sassertion}{0{}}{
4430   \__stex_statements_sassertion_args:n{ #1 }
4431   \stex_if_smsmode:F {
4432     \seq_clear:N \l_tmpa_seq
4433     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4434       \str_if_eq:nnF{ ##1 }{}{
4435         \stex_get_symbol:n { ##1 }
4436         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4437           \l_stex_get_symbol_uri_str
4438         }
4439       }
4440     }
4441     \exp_args:Nnnx
4442     \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}
4443     \str_if_empty:NF \sassertiontype {
4444       \stex_annotate_invisible:nnn{type}{\sassertiontype}{ }
4445     }
4446     \clist_set:N \l_tmpa_clist \sassertiontype
4447     \tl_clear:N \l_tmpa_tl
4448     \clist_map_inline:Nn \l_tmpa_clist {
4449       \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{

```



```

4450         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
4451     }
4452 }
4453 \tl_if_empty:NTF \l_tmpa_tl {
4454     \__stex_statements_sassertion_start:
4455 }{
4456     \l_tmpa_tl
4457 }
4458 }
4459 \str_if_empty:NTF \sassertionid {
4460     \str_if_empty:NF \sassertionname {
4461         \stex_ref_new_doc_target:n {}
4462     }
4463 } {
4464     \stex_ref_new_doc_target:n \sassertionid
4465 }
4466 \stex_smsmode_do:
4467 }{
4468     \str_if_empty:NF \sassertionname {
4469         \stex_symdecl_do:nn{ }\sassertionname}
4470         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4471     }
4472     \stex_if_smsmode:F {
4473         \clist_set:No \l_tmpa_clist \sassertiontype
4474         \tl_clear:N \l_tmpa_tl
4475         \clist_map_inline:Nn \l_tmpa_clist {
4476             \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
4477                 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
4478             }
4479         }
4480         \tl_if_empty:NTF \l_tmpa_tl {
4481             \__stex_statements_sassertion_end:
4482         }{
4483             \l_tmpa_tl
4484         }
4485         \end{stex_annotate_env}
4486     }
4487 }

```

## \stexpatchassertion

```

4488
4489 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
4490     \par\noindent\titllemph{Assertion~\tl_if_empty:NF \sassertiontitle {
4491         (\sassertiontitle)
4492     }~}
4493 }
4494 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
4495
4496 \newcommand\stexpatchassertion[3] [] {
4497     \str_set:Nx \l_tmpa_str{ #1 }
4498     \str_if_empty:NTF \l_tmpa_str {
4499         \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
4500         \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
4501     }{

```

```

4502     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
4503     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
4504   }
4505 }

```

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

`\inlineass` inline:

```

4506 \keys_define:nn {stex / inlineass }{
4507   type      .str_set_x:N = \sassertiontype,
4508   id        .str_set_x:N = \sassertionid,
4509   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4510   name      .str_set_x:N = \sassertionname
4511 }
4512 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
4513   \str_clear:N \sassertiontype
4514   \str_clear:N \sassertionid
4515   \str_clear:N \sassertionname
4516   \clist_clear:N \l__stex_statements_sassertion_for_clist
4517   \keys_set:nn { stex / inlineass }{ #1 }
4518 }
4519 \NewDocumentCommand \inlineass { 0{} m } {
4520   \begingroup
4521   \__stex_statements_inlineass_args:n{ #1 }
4522   \str_if_empty:NTF \sassertionid {
4523     \str_if_empty:NF \sassertionname {
4524       \stex_ref_new_doc_target:n {}
4525     }
4526   } {
4527     \stex_ref_new_doc_target:n \sassertionid
4528   }
4529
4530   \stex_if_smsmode:TF{
4531     \str_if_empty:NF \sassertionname {
4532       \stex_symdecl_do:nn{}{\sassertionname}
4533       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4534     }
4535   }{
4536     \seq_clear:N \l_tmpa_seq
4537     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4538       \str_if_eq:nnF{ ##1 }{}{
4539         \stex_get_symbol:n { ##1 }
4540         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4541           \l_stex_get_symbol_uri_str
4542         }
4543       }
4544     }
4545     \exp_args:Nnx
4546     \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{
4547       \str_if_empty:NF \sassertiontype {
4548         \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
4549       }
4550       #2
4551       \str_if_empty:NF \sassertionname {

```

```

4552         \stex_symdecl_do:nn{}{\sassertionname}
4553         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4554     }
4555 }
4556 }
4557 \endgroup
4558 \stex_smsmode_do:
4559 }

```

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

### 33.3 Examples

`sexample`

```

4560
4561 \keys_define:nn {stex / sexample }{
4562   type      .str_set_x:N = \exampletype,
4563   id        .str_set_x:N = \sexampleid,
4564   title     .tl_set:N     = \sexamplename,
4565   for       .clist_set:N  = \l__stex_statements_sexample_for_clist,
4566 }
4567 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
4568   \str_clear:N \sexampletype
4569   \str_clear:N \sexampleid
4570   \tl_clear:N \sexamplename
4571   \clist_clear:N \l__stex_statements_sexample_for_clist
4572   \keys_set:nn { stex / sexample }{ #1 }
4573 }
4574
4575 \NewDocumentEnvironment{sexample}{0{}}{
4576   \__stex_statements_sexample_args:n{ #1 }
4577   \stex_if_smsmode:F {
4578     \seq_clear:N \l_tmpa_seq
4579     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4580       \str_if_eq:nnF{ ##1 }{ }{
4581         \stex_get_symbol:n { ##1 }
4582         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4583           \l_stex_get_symbol_uri_str
4584         }
4585       }
4586     }
4587     \exp_args:Nnnx
4588     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
4589     \str_if_empty:NF \sexampletype {
4590       \stex_annotate_invisible:nnn{type}{\sexampletype}{ }
4591     }
4592     \clist_set:Nn \l_tmpa_clist \sexampletype
4593     \tl_clear:N \l_tmpa_tl
4594     \clist_map_inline:Nn \l_tmpa_clist {
4595       \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
4596         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
4597       }
4598     }

```

```

4599 \tl_if_empty:NTF \l_tmpa_tl {
4600   \__stex_statements_sexample_start:
4601 }{
4602   \l_tmpa_tl
4603 }
4604 }
4605 \str_if_empty:NF \sexampleid {
4606   \stex_ref_new_doc_target:n \sexampleid
4607 }
4608 \stex_smsmode_do:
4609 }{
4610   \str_if_empty:NF \sexamplename { \stex_symdecl_do:nn}{\sexamplename} }
4611   \stex_if_smsmode:F {
4612     \clist_set:No \l_tmpa_clist \sexamplotype
4613     \tl_clear:N \l_tmpa_tl
4614     \clist_map_inline:Nn \l_tmpa_clist {
4615       \tl_if_exist:cT {\__stex_statements_sexample_##1_end:}{
4616         \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_statements_sexample_##1_end:}}
4617       }
4618     }
4619     \tl_if_empty:NTF \l_tmpa_tl {
4620       \__stex_statements_sexample_end:
4621     }{
4622       \l_tmpa_tl
4623     }
4624     \end{stex_annotate_env}
4625   }
4626 }

```

\stexpatchexample

```

4627
4628 \cs_new_protected:Nn \__stex_statements_sexample_start: {
4629   \par\noindent\titllemph{Example~\tl_if_empty:NF \sexamplename {
4630     (\sexamplename)
4631   }~}
4632 }
4633 \cs_new_protected:Nn \__stex_statements_sexample_end: {\par\medskip}
4634
4635 \newcommand\stexpatchexample[3]{} {
4636   \str_set:Nx \l_tmpa_str{ #1 }
4637   \str_if_empty:NTF \l_tmpa_str {
4638     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
4639     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
4640   }{
4641     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
4642     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
4643   }
4644 }

```

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

\inlineex inline:

```

4645 \keys_define:nn {stex / inlineex }{
4646   type .str_set_x:N = \sexamplotype,

```

```

4647 id      .str_set_x:N = \sexampleid,
4648 for      .clist_set:N = \l__stex_statements_sexample_for_clist ,
4649 name     .str_set_x:N = \sexamplename
4650 }
4651 \cs_new_protected:Nn \__stex_statements_inlinex_args:n {
4652   \str_clear:N \sexamplotype
4653   \str_clear:N \sexampleid
4654   \str_clear:N \sexamplename
4655   \clist_clear:N \l__stex_statements_sexample_for_clist
4656   \keys_set:nn { stex / inlinex }{ #1 }
4657 }
4658 \NewDocumentCommand \inlinex { 0{} m } {
4659   \begingroup
4660   \__stex_statements_inlinex_args:n{ #1 }
4661   \str_if_empty:NF \sexampleid {
4662     \stex_ref_new_doc_target:n \sexampleid
4663   }
4664   \stex_if_smsmode:TF{
4665     \str_if_empty:NF \sexamplename { \stex_symdecl_do:nn{ }\sexamplename } }
4666   ){
4667     \seq_clear:N \l_tmpa_seq
4668     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4669       \str_if_eq:nnF{ ##1 }{ }{
4670         \stex_get_symbol:n { ##1 }
4671         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4672           \l_stex_get_symbol_uri_str
4673         }
4674       }
4675     }
4676     \exp_args:Nnx
4677     \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpa_seq {,}}{
4678       \str_if_empty:NF \sexamplotype {
4679         \stex_annotate_invisible:nnn{type}{\sexamplotype}{ }
4680       }
4681       #2
4682       \str_if_empty:NF \sexamplename { \stex_symdecl_do:nn{ }\sexamplename } }
4683   }
4684 }
4685 \endgroup
4686 \stex_smsmode_do:
4687 }

```

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

## 33.4 Logical Paragraphs

sparagraph

```

4688 \keys_define:nn { stex / sparagraph } {
4689   id      .str_set_x:N = \sparagraphid ,
4690   title   .tl_set:N    = \l_stex_sparagraph_title_tl ,
4691   type    .str_set_x:N = \sparagraphtype ,
4692   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
4693   from    .tl_set:N    = \sparagraphfrom ,

```

```

4694 to .tl_set:N = \sparagraphto ,
4695 start .tl_set:N = \l_stex_sparagraph_start_tl ,
4696 name .str_set:N = \sparagraphname
4697 }
4698
4699 \cs_new_protected:Nn \stex_sparagraph_args:n {
4700 \tl_clear:N \l_stex_sparagraph_title_tl
4701 \tl_clear:N \sparagraphfrom
4702 \tl_clear:N \sparagraphto
4703 \tl_clear:N \l_stex_sparagraph_start_tl
4704 \str_clear:N \sparagraphid
4705 \str_clear:N \sparagraphtype
4706 \clist_clear:N \l__stex_statements_sparagraph_for_clist
4707 \str_clear:N \sparagraphname
4708 \keys_set:nn { stex / sparagraph }{ #1 }
4709 }
4710 \newif\if@in@omtext\@in@omtextfalse
4711
4712 \NewDocumentEnvironment {sparagraph} { 0{} } {
4713 \stex_sparagraph_args:n { #1 }
4714 \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4715 \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
4716 }{
4717 \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
4718 }
4719 \@in@omtexttrue
4720 \stex_if_smsmode:F {
4721 \seq_clear:N \l_tmpa_seq
4722 \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
4723 \str_if_eq:nnF{ ##1 }{}{
4724 \stex_get_symbol:n { ##1 }
4725 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4726 \l_stex_get_symbol_uri_str
4727 }
4728 }
4729 }
4730 \exp_args:Nnnx
4731 \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
4732 \str_if_empty:NF \sparagraphtype {
4733 \stex_annotate_invisible:nnn{type}{\sparagraphtype}{ }
4734 }
4735 \str_if_empty:NF \sparagraphfrom {
4736 \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{ }
4737 }
4738 \str_if_empty:NF \sparagraphto {
4739 \stex_annotate_invisible:nnn{to}{\sparagraphto}{ }
4740 }
4741 \clist_set:N \l_tmpa_clist \sparagraphtype
4742 \tl_clear:N \l_tmpa_tl
4743 \clist_map_inline:Nn \sparagraphtype {
4744 \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
4745 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
4746 }
4747 }

```

```

4748 \tl_if_empty:NTF \l_tmpa_tl {
4749   \__stex_statements_sparagraph_start:
4750 }{
4751   \l_tmpa_tl
4752 }
4753 }
4754 \clist_set:No \l_tmpa_clist \sparagraphtype
4755 \str_if_empty:NTF \sparagraphid {
4756   \str_if_empty:NTF \sparagraphname {
4757     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
4758       \stex_ref_new_doc_target:n {}
4759     }
4760   } {
4761     \stex_ref_new_doc_target:n {}
4762   }
4763 } {
4764   \stex_ref_new_doc_target:n \sparagraphid
4765 }
4766 \exp_args:NNx
4767 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
4768   \clist_map_inline:Nn \__stex_statements_sparagraph_for_clist {
4769     \str_if_eq:nnF{ ##1 }{ }{
4770       \stex_get_symbol:n { ##1 }
4771       \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4772     }
4773   }
4774 }
4775 \stex_smsmode_do:
4776 \ignorespacesandpars
4777 }{
4778   \str_if_empty:NF \sparagraphname {
4779     \stex_symdecl_do:nn{ }\sparagraphname}
4780   \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
4781 }
4782 \stex_if_smsmode:F {
4783   \clist_set:No \l_tmpa_clist \sparagraphtype
4784   \tl_clear:N \l_tmpa_tl
4785   \clist_map_inline:Nn \l_tmpa_clist {
4786     \tl_if_exist:cT {\__stex_statements_sparagraph_##1_end:}{
4787       \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_statements_sparagraph_##1_end:}}
4788     }
4789   }
4790   \tl_if_empty:NTF \l_tmpa_tl {
4791     \__stex_statements_sparagraph_end:
4792   }{
4793     \l_tmpa_tl
4794   }
4795   \end{stex_annotate_env}
4796 }
4797 }

```

\stexpatchparagraph

```

4798
4799 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {

```

```

4800 \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4801   \tl_if_empty:NF \l_stex_sparagraph_title_tl {
4802     \titleemph{\l_stex_sparagraph_title_tl}:~
4803   }
4804 }{
4805   \titleemph{\l_stex_sparagraph_start_tl}~
4806 }
4807 }
4808 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
4809
4810 \newcommand\stexpatchparagraph[3] [] {
4811   \str_set:Nx \l_tmpa_str{ #1 }
4812   \str_if_empty:NTF \l_tmpa_str {
4813     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
4814     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
4815   }{
4816     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2 }
4817     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
4818   }
4819 }
4820
4821 \keys_define:nn { stex / inlinepara } {
4822   id      .str_set:N = \sparagraphid ,
4823   type    .str_set:N = \sparagraphtype ,
4824   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
4825   from    .tl_set:N = \sparagraphfrom ,
4826   to      .tl_set:N = \sparagraphto ,
4827   name    .str_set:N = \sparagraphname
4828 }
4829 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
4830   \tl_clear:N \sparagraphfrom
4831   \tl_clear:N \sparagraphto
4832   \str_clear:N \sparagraphid
4833   \str_clear:N \sparagraphtype
4834   \clist_clear:N \l__stex_statements_sparagraph_for_clist
4835   \str_clear:N \sparagraphname
4836   \keys_set:nn { stex / inlinepara }{ #1 }
4837 }
4838 \NewDocumentCommand \inlinepara { O{} m } {
4839   \begingroup
4840   \__stex_statements_inlinepara_args:n{ #1 }
4841   \clist_set:No \l_tmpa_clist \sparagraphtype
4842   \str_if_empty:NTF \sparagraphid {
4843     \str_if_empty:NTF \sparagraphname {
4844       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
4845         \stex_ref_new_doc_target:n {}
4846       }
4847     } {
4848       \stex_ref_new_doc_target:n {}
4849     }
4850   } {
4851     \stex_ref_new_doc_target:n \sparagraphid
4852   }
4853   \stex_if_smsmode:TF{

```



```

4854 \str_if_empty:NF \sparagraphname {
4855   \stex_symdecl_do:nn{}{\sparagraphname}
4856   \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
4857 }
4858 }{
4859   \seq_clear:N \l_tmpa_seq
4860   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
4861     \str_if_eq:nnF{ ##1 }{}{
4862       \stex_get_symbol:n { ##1 }
4863       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4864         \l_stex_get_symbol_uri_str
4865       }
4866     }
4867   }
4868   \exp_args:Nnx
4869   \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
4870     \str_if_empty:NF \sparagraphtype {
4871       \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}
4872     }
4873     \str_if_empty:NF \sparagraphfrom {
4874       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
4875     }
4876     \str_if_empty:NF \sparagraphto {
4877       \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
4878     }
4879     \str_if_empty:NF \sparagraphname {
4880       \stex_symdecl_do:nn{}{\sparagraphname}
4881       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
4882     }
4883     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
4884       \clist_map_inline:Nn \l_tmpa_seq {
4885         \stex_ref_new_sym_target:n {##1}
4886       }
4887     }
4888     #2
4889   }
4890 }
4891 \endgroup
4892 \stex_smsmode_do:
4893 }
4894

```

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

#### symboldoc

```

4895 \NewDocumentEnvironment{symboldoc}{m}{
4896   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
4897   \seq_clear:N \l_tmpb_seq
4898   \seq_map_inline:Nn \l_tmpa_seq {
4899     \str_if_eq:nnF{ ##1 }{}{
4900       \stex_get_symbol:n { ##1 }
4901       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
4902         \l_stex_get_symbol_uri_str
4903       }

```

```

4904     }
4905   }
4906   \par
4907   \exp_args:Nnnx
4908   \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
4909   ){
4910     \end{stex_annotate_env}
4911   }
4912 </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>

```
4913 \package
4914 \@@=stex_sproof
4915
4916 %%%%%%%%%%%%%%%%% sproof.dtx %%%%%%%%%%%%%%%%%
4917
```

### 34.2 Proofs

We first define some keys for the proof environment.

```
4918 \keys_define:nn { stex / spf } {
4919   id          .str_set:N = \l__stex_sproof_spf_id_str,
4920   display     .tl_set:N  = \l__stex_sproof_spf_display_tl,
4921   for         .tl_set:N  = \l__stex_sproof_spf_for_tl ,
4922   from        .tl_set:N  = \l__stex_sproof_spf_from_tl ,
4923   proofend    .tl_set:N  = \l__stex_sproof_spf_proofend_tl,
4924   type        .tl_set:N  = \l__stex_sproof_spf_type_tl,
4925   title       .tl_set:N  = \l__stex_sproof_spf_title_tl,
4926   continues   .tl_set:N  = \l__stex_sproof_spf_continues_tl,
4927   functions   .tl_set:N  = \l__stex_sproof_spf_functions_tl,
4928   method      .tl_set:N  = \l__stex_sproof_spf_method_tl
4929 }
4930 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
4931   \str_clear:N \l__stex_sproof_spf_id_str
4932   \tl_clear:N \l__stex_sproof_spf_display_tl
4933   \tl_clear:N \l__stex_sproof_spf_for_tl
4934   \tl_clear:N \l__stex_sproof_spf_from_tl
4935   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\proof@box}
4936   \tl_clear:N \l__stex_sproof_spf_type_tl
4937   \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

```

4938 \tl_clear:N \l__stex_sproof_spf_continues_tl
4939 \tl_clear:N \l__stex_sproof_spf_functions_tl
4940 \tl_clear:N \l__stex_sproof_spf_method_tl
4941 \keys_set:nn { stex / spf }{ #1 }
4942 }

```

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

```

4943 \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.

```

4944 \newcount\count_ten
4945 \newenvironment{pst@with@label}[1]{
4946   \edef\pst@label{#1}
4947   \advance\count_ten by 1\relax
4948   \count_ten=1
4949 }{
4950   \advance\count_ten by -1\relax
4951 }

```

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

```

4952 \def\the@pst@label{
4953   \pst@make@label\pst@label{\number\count_ten}\l__stex_sproof_pstlabel_postfix_tl
4954 }

```

(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.

```

4955 \keys_define:nn { stex / pstlabel }{
4956   prefix      .tl_set:N   = \l__stex_sproof_pstlabel_prefix_tl,
4957   delimiter   .tl_set:N   = \l__stex_sproof_pstlabel_delimiter_tl,
4958   postfix     .tl_set:N   = \l__stex_sproof_pstlabel_postfix_tl
4959 }
4960 \cs_new_protected:Nn \__stex_sproof_pstlabel_args:n {

```

---

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

```

4961 \tl_set:Nn \l__stex_sproof_pstlabel_prefix_tl {P}
4962 \tl_set:Nn \l__stex_sproof_pstlabel_delimiter_tl {.}
4963 \tl_clear:N \l__stex_sproof_pstlabel_postfix_tl
4964 }
4965 \__stex_sproof_pstlabel_args:n {}
4966 \newcommand\setpstlabelstyle[1]{
4967   \__stex_sproof_pstlabel_args:n {#1}
4968 }
4969 \newcommand\setpstlabelstyledefault{%
4970   \__stex_sproof_pstlabel_args:n{prefix=P,delimiter=.,postfix={}}
4971 }

```

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

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

```

4972 \ExplSyntaxOff
4973 \def\pst@make@label@long#1#2{\@for\@I:=#1\do{\expandafter\expandafter\expandafter\@I\csname
4974 \def\pst@make@label@angles#1#2{\ensuremath{\@for\@I:=#1\do{\rangle}}#2}
4975 \def\pst@make@label@short#1#2{#2}
4976 \def\pst@make@label@empty#1#2{}
4977 \ExplSyntaxOn
4978 \def\pstlabelstyle#1{%
4979   \def\pst@make@label{\use:c{pst@make@label@#1}}%
4980 }%
4981 \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.

```

4982 \def\next@pst@label{%
4983   \global\advance\count\count10 by 1%
4984 }%

```

(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

```

4985 \def\sproof@box{
4986   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
4987 }
4988 \def\spf@proofend{\sproof@box}
4989 \def\sproofend{
4990   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
4991     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
4992   }
4993 }
4994 \def\sProofEndSymbol#1{\def\sproof@box{#1}}

```

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

**spf@\*@kw**

```

4995 \def\spf@proofsketch@kw{Proof Sketch}
4996 \def\spf@proof@kw{Proof}
4997 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

4998 \AddToHook{begindocument}{
4999   \ltx@ifpackageloaded{babel}{
5000     \makeatletter
5001     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5002     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5003       \input{sproof-ngerman.ldf}
5004     }
5005     \clist_if_in:NnT \l_tmpa_clist {finnish}{
5006       \input{sproof-finnish.ldf}
5007     }
5008     \clist_if_in:NnT \l_tmpa_clist {french}{
5009       \input{sproof-french.ldf}
5010     }
5011     \clist_if_in:NnT \l_tmpa_clist {russian}{
5012       \input{sproof-russian.ldf}
5013     }
5014     \makeatother
5015   }{}
5016 }

```

`spfsketch`

```

5017 \newcommand\spfsketch[2][]{
5018   \__stex_sproof_spf_args:n{#1}
5019   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
5020     \titleemph{
5021       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
5022         \spf@proofsketch@kw
5023       }{
5024         \l__stex_sproof_spf_type_tl
5025       }
5026     }:
5027   }
5028   {~#2}
5029   %\sref@label@id{this \ifx\spf@type\empty\spf@proofsketch@kw\else\spf@type\fi}
5030   \sproofend
5031 }

```

(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>

```

5032 \newenvironment{spfeq}[2][]{
5033   \__stex_sproof_spf_args:n{#1}
5034   %\sref@target
5035   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
5036     \titleemph{
5037       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
5038         \spf@proof@kw
5039       }{

```

<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

```

5040         \l__stex_sproof_spf_type_tl
5041     }
5042     }:
5043 }
5044 {~#2}
5045 \begin{displaymath}\begin{array}{rc1l}
5046 }{
5047 \end{array}\end{displaymath}
5048 }

```

(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.

```

5049 \newenvironment{spf@proof}[2] []{
5050   \l__stex_sproof_spf_args:n{#1}
5051   %\sref@target
5052   \count_ten=10
5053   \par\noindent
5054   \tl_if_eq:NNTF \l__stex_sproof_spf_display_tl\spf@flow{
5055     \titleemph{
5056       \tl_if_empty:NNTF \l__stex_sproof_spf_type_tl {
5057         \spf@proof@kw
5058       }{
5059         \l__stex_sproof_spf_type_tl
5060       }
5061     }:
5062   }
5063   {~#2}
5064   %\sref@label{id{this \ifx\spf@type\@empty\spf@proof@kw\else\spf@type\fi}
5065   \def\pst@label{}
5066   \newcount\pst@count% initialize the labeling mechanism
5067   \begin{description}\begin{pst@with@label}{\l__stex_sproof_pstlabel_prefix_tl}
5068 }{
5069   \end{pst@with@label}\end{description}
5070 }
5071 \newenvironment{sproof}[2] []{\begin{spf@proof}[#1]{#2}}{\sproofend\end{spf@proof}}
5072 \newenvironment{sProof}[2] []{\begin{spf@proof}[#1]{#2}}{\end{spf@proof}}

```

**\spfidea**

```

5073 \newcommand\spfidea[2] []{
5074   \l__stex_sproof_spf_args:n{#1}
5075   \titleemph{
5076     \tl_if_empty:NNTF \l__stex_sproof_spf_type_tl {Proof~Idea}{
5077       \l__stex_sproof_spf_type_tl
5078     }:
5079   }~#2
5080   \sproofend
5081 }

```

(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
5082 \newenvironment{spfstep}[1][]{
5083   \_stex_sproof_spf_args:n{#1}
5084   \@in@omtexttrue
5085   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
5086     \item[\the@pst@label]
5087   }
5088   \tl_if_empty:NF \l__stex_sproof_spf_title_tl {
5089     {(\titleemph{\l__stex_sproof_spf_title_tl})\enspace}
5090   }
5091   %\sref@label@id{\pst@label}
5092   \ignorespacesandpars
5093 }{
5094   \next@pst@label\ignorespacesandpars
5095 }

```

**sproofcomment**

```

5096 \newenvironment{sproofcomment}[1][]{
5097   \_stex_sproof_spf_args:n{#1}
5098   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
5099     \item[\the@pst@label]
5100   }
5101 }{
5102   \next@pst@label
5103 }

```

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.

```

5104 \newenvironment{subproof}[2][]{
5105   \_stex_sproof_spf_args:n{#1}
5106   \def\@test{#2}
5107   \ifx\@test\empty\else
5108     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
5109       \item[\the@pst@label]
5110     }{#2}
5111   \fi
5112   \begin{pst@with@label}{\pst@label,\number\count_ten}
5113 }{
5114   \end{pst@with@label}\next@pst@label
5115 }

```

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

```

5116 \newenvironment{spfcases}[2][]{
5117   \def\@test{#1}
5118   \ifx\@test\empty
5119     \begin{subproof}[method=by-cases]{#2}

```

---

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



```

5120 \else
5121   \begin{subproof}[#1,method=by-cases]{#2}
5122 \fi
5123 }{
5124   \end{subproof}
5125 }

```

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

```

5126 \newenvironment{spfcase}[2] [] {
5127   \__stex_sproof_spf_args:n{#1}
5128   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
5129     \item[\the@pst@label]
5130   }
5131   \def\@test{#2}
5132   \ifx\@test\@empty
5133   \else
5134     {\titleemph{#2}:~}
5135   \fi
5136   \begin{pst@with@label}{\pst@label,\number\count_ten}
5137 }{
5138   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
5139     \sproofend
5140   }
5141   \end{pst@with@label}
5142   \next@pst@label
5143 }

```

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

```

5144 \newcommand\spfcasesketch[3] [] {
5145   \__stex_sproof_spf_args:n{#1}
5146   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
5147     \item[\the@pst@label]
5148   }
5149   \def\@test{#2}
5150   \ifx\@test\@empty
5151   \else
5152     {\titleemph{#2}:~}
5153   \fi#3
5154   \next@pst@label
5155 }%

```

### 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.

```

5156 \keys_define:nn { stex / just }{
5157   id          .str_set:x:N = \l__stex_sproof_just_id_str,
5158   method      .tl_set:N    = \l__stex_sproof_just_method_tl,
5159   premises    .tl_set:N    = \l__stex_sproof_just_premises_tl,
5160   args        .tl_set:N    = \l__stex_sproof_just_args_tl
5161 }

```

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
5162 \newenvironment{justification}[1] [] {}{}

\premise
5163 \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.
5164 \newcommand\justarg[2] [] {}{#2}
5165 \</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

```
5166 <*package>
5167
5168 %%%%%%%%%% others.dtx %%%%%%%%%%
5169
5170 <@@=stex_others>
    Warnings and error messages
5171 % None

\MSC Math subject classifier

5172 \NewDocumentCommand \MSC {m} {
5173 % TODO
5174 }

(End definition for \MSC. This function is documented on page ??.)
    Patching tikzinput, if loaded
5175 \@ifpackageloaded{tikzinput}{
5176 \RequirePackage{stex-tikzinput}
5177 }{}
5178 </package>
```

## Chapter 36

# STEX -Metatheory Implementation

```
5179 <*package>
5180 <@@=stex_modules>
5181
5182 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
5183
5184 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
5185 \begingroup
5186 \stex_module_setup:nn{
5187   ns=\c_stex_metatheory_ns_str,
5188   meta=NONE
5189 }{Metatheory}
5190 \stex_reactivate_macro:N \symdecl
5191 \stex_reactivate_macro:N \notation
5192 \stex_reactivate_macro:N \symdef
5193 \ExplSyntaxOff
5194 \csname stex_suppress_html:n\endcsname{
5195   % is-a (a:A, a \in A, a is an A, etc.)
5196   \symdecl[args=ai]{isa}
5197   \notation[typed]{isa}{#1 \comp{:} #2}{##1 \comp, ##2}
5198   \notation[in]{isa}{#1 \comp\in #2}{##1 \comp, ##2}
5199   \notation[pred]{isa}{#2\comp(#1 \comp)}{##1 \comp, ##2}
5200
5201   % bind (\forall, \Pi, \lambda etc.)
5202   \symdecl[args=Bi]{bind}
5203   \notation[forall]{bind}{\comp\forall #1.\;#2}{##1 \comp, ##2}
5204   \notation[\Pi]{bind}{\comp\prod_{#1}#2}{##1 \comp, ##2}
5205   \notation[deffun]{bind}{\comp( #1 \comp{ }\;\to\; ) #2}{##1 \comp, ##2}
5206
5207   % dummy variable
5208   \symdecl{dummyvar}
5209   \notation[underscore]{dummyvar}{\comp\_}
5210   \notation[dot]{dummyvar}{\comp\cdot}
5211   \notation[dash]{dummyvar}{\comp{\rm --}}
5212
5213   %fromto (function space, Hom-set, implication etc.)
```

```

5214 \symdecl[args=ai]{fromto}
5215 \notation[xarrow]{fromto}{#1 \comp\to #2}{##1 \comp\times ##2}
5216 \notation[arrow]{fromto}{#1 \comp\to #2}{##1 \comp\to ##2}
5217
5218 % mapto (lambda etc.)
5219 %\symdecl[args=Bi]{mapto}
5220 %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
5221 %\notation[lambda]{mapto}{\comp\lambda #1 \comp.\; #2}{#1 \comp, #2}
5222 %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp.\; #2}{#1 \comp, #2}
5223
5224 % function/operator application
5225 \symdecl[args=ia]{apply}
5226 \notation[prec=0;0x\infpres,parens]{apply}{#1 \comp( #2 \comp)}{##1 \comp, ##2}
5227 \notation[prec=0;0x\infpres,lambda]{apply}{#1 \; #2 }{##1 \; ; ##2}
5228
5229 % ‘type’ of all collections (sets, classes, types, kinds)
5230 \symdecl{collection}
5231 \notation[U]{collection}{\comp{\mathcal{U}}}
5232 \notation[set]{collection}{\comp{\textsf{Set}}}
5233
5234 % collection of propositions/booleans/truth values
5235 \symdecl[name=proposition]{prop}
5236 \notation[prop]{prop}{\comp{\rm prop}}
5237 \notation[BOOL]{prop}{\comp{\rm BOOL}}
5238
5239 % sequences
5240 \symdecl[args=1]{seqtype}
5241 \notation[kleene]{seqtype}{#1^{\comp\ast}}
5242
5243 \symdef[args=2,li,prec=nobrackets]{sequence-index}{#1_{#2}}
5244 \notation[ui,prec=nobrackets]{sequence-index}{#1^{#2}}
5245
5246 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{\, \ellipses}}{##1\comp, ##2}
5247 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{\, \ellipses, }#2}{##1\comp, ##2}
5248 \symdef[args=aia,prec=nobrackets]{aseqfromtovia}{#1\comp{\, \ellipses, }#2\comp{\, \ellipses, }#3}
5249
5250 % letin (‘let’, local definitions, variable substitution)
5251 \symdecl[args=bii]{letin}
5252 \notation[let]{letin}{\comp{\rm let}}\; #1\comp{=}\; #2\; \comp{\rm in}}\; #3}
5253 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
5254 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
5255
5256 % structures
5257 \symdecl*[args=1]{module-type}
5258 \notation{module-type}{\mathtt{MOD} #1}
5259 \symdecl[name=mathematical-structure, args=a]{mathstruct} % TODO
5260 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\angle}{##1 \comp, ##2}
5261
5262 }
5263 \ExplSyntaxOn
5264 \stex_add_to_current_module:n{
5265   \let\appa\apply
5266   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
5267   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}

```

```

5268 \def\livar{\csname sequence-index\endcsname[li]}
5269 \def\uivar{\csname sequence-index\endcsname[ui]}
5270 \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
5271 \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
5272 \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
5273 }
5274 \_stex_modules_end_module:
5275 \endgroup
5276 </package>

```

## Chapter 37

# Tikzinput Implementation

```
5277 <*package>
5278
5279 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
5280
5281 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
5282 \RequirePackage{l3keys2e}
5283
5284 \keys_define:nn { tikzinput } {
5285   image .bool_set:N = \c_tikzinput_image_bool,
5286   image .default:n = false ,
5287   unknown .code:n = {}
5288 }
5289
5290 \ProcessKeysOptions { tikzinput }
5291
5292 \bool_if:NTF \c_tikzinput_image_bool {
5293   \RequirePackage{graphicx}
5294
5295   \providecommand\usetikzlibrary[]{}
5296   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
5297 }{
5298   \RequirePackage{tikz}
5299   \RequirePackage{standalone}
5300
5301   \newcommand \tikzinput [2] [] {
5302     \setkeys{Gin}{#1}
5303     \ifx \Gin@ewidth \Gin@exclamation
5304       \ifx \Gin@eheight \Gin@exclamation
5305         \input { #2 }
5306       \else
5307         \resizebox{!}{ \Gin@eheight }{
5308           \input { #2 }
5309         }
5310       \fi
5311     \else
5312       \ifx \Gin@eheight \Gin@exclamation
5313         \resizebox{ \Gin@ewidth }{!}{
5314           \input { #2 }
```

```

5315     }
5316     \else
5317         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
5318             \input { #2 }
5319         }
5320     \fi
5321 \fi
5322 }
5323 }
5324
5325 \newcommand \ctikzinput [2] [] {
5326     \begin{center}
5327         \tikzinput [ #1 ] { #2 }
5328     \end{center}
5329 }
5330
5331 \@ifpackageloaded{stex}{
5332     \RequirePackage{stex-tikzinput}
5333 }{}
5334
5335 </package>
5336 <*stex>
5337 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
5338 \RequirePackage{stex}
5339 \RequirePackage{tikzinput}
5340
5341 \newcommand\mhtikzinput [2] [] {%
5342     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
5343     \stex_in_repository:nn\Gin@mhrepos{
5344         \tikzinput [ #1 ] {\mhp{##1}{#2}}
5345     }
5346 }
5347 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [ #1 ] { #2 }\end{center}}
5348 </stex>

```

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



## Chapter 38

# document-structure.sty Implementation

### 38.1 The document-structure Class

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

```
5349 \*cls)
5350 \@@=document_structure)
5351 \ProvidesExplClass{document-structure}{2022/02/10}{3.0}{Modular Document Structure Class}
5352 \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`

```
5353 \keys_define:nn{ document-structure / pkg }{
5354   class      .str_set_x:N = \c_document_structure_class_str,
5355   minimal    .bool_set:N = \c_document_structure_minimal_bool,
5356   report     .code:n      = {
5357     \ClassWarning{document-structure}{the option 'report' is deprecated, use 'class=report',
5358     \str_set:Nn \c_document_structure_class_str {report}
5359   },
5360   book       .code:n      = {
5361     \ClassWarning{document-structure}{the option 'book' is deprecated, use 'class=book', ins
5362     \str_set:Nn \c_document_structure_class_str {book}
5363   },
5364   bookpart   .code:n      = {
5365     \ClassWarning{document-structure}{the option 'bookpart' is deprecated, use 'class=book,t
5366     \str_set:Nn \c_document_structure_class_str {book}
5367     \str_set:Nn \c_document_structure_topsect_str {chapter}
5368   },
```

```

5369 docopt      .str_set_x:N = \c_document_structure_docopt_str,
5370 unknown     .code:n      = {
5371   \PassOptionsToPackage{ \CurrentOption }{ document-structure }
5372 }
5373 }
5374 \ProcessKeysOptions{ document-structure / pkg }
5375 \str_if_empty:NT \c_document_structure_class_str {
5376   \str_set:Nn \c_document_structure_class_str {article}
5377 }
5378 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
5379   {\c_document_structure_class_str}
5380

```

### 38.3 Beefing up the document environment

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

```

5381 \RequirePackage{document-structure}
5382 \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>

```

5383 \keys_define:nn { document-structure / document }{
5384   id .str_set_x:N = \c_document_structure_document_id_str
5385 }
5386 \let\__document_structure_orig_document=\document
5387 \renewcommand{\document}[1][]{
5388   \keys_set:nn{ document-structure / document }{ #1 }
5389   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
5390   \__document_structure_orig_document
5391 }

```

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

```

5392 }
5393 \</cls>

```

### 38.4 Implementation: document-structure Package

```

5394 \*package>
5395 \ProvidesExplPackage{document-structure}{2022/02/10}{3.0}{Modular Document Structure}
5396 \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

```

5397
5398 \keys_define:nn{ document-structure / pkg }{
5399   class      .str_set_x:N = \c_document_structure_class_str,
5400   topsect    .str_set_x:N = \c_document_structure_topsect_str,
5401   % showignores .bool_set:N = \c_document_structure_showignores_bool,
5402 }
5403 \ProcessKeysOptions{ document-structure / pkg }
5404 \str_if_empty:NT \c_document_structure_class_str {
5405   \str_set:Nn \c_document_structure_class_str {article}
5406 }
5407 \str_if_empty:NT \c_document_structure_topsect_str {
5408   \str_set:Nn \c_document_structure_topsect_str {section}
5409 }

```

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

```

5410 \RequirePackage{xspace}
5411 \RequirePackage{comment}
5412 \AddToHook{begindocument}{
5413   \ltx@ifpackageloaded{babel}{
5414     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5415     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5416       \makeatletter\input{omdoc-ngerman.ldf}\makeatother
5417     }
5418   }{}
5419 }

```

`\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.

```

5420 \int_new:N \l_document_structure_section_level_int
5421 \str_case:VnF \c_document_structure_topsect_str {
5422   {part}}{
5423     \int_set:Nn \l_document_structure_section_level_int {0}
5424   }
5425   {chapter}}{
5426     \int_set:Nn \l_document_structure_section_level_int {1}
5427   }
5428 }{
5429   \str_case:VnF \c_document_structure_class_str {
5430     {book}}{
5431       \int_set:Nn \l_document_structure_section_level_int {0}
5432     }
5433     {report}}{
5434       \int_set:Nn \l_document_structure_section_level_int {0}
5435     }
5436   }{
5437     \int_set:Nn \l_document_structure_section_level_int {2}
5438   }
5439 }

```

## 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

```
5440 \def\current@section@level{document}%
5441 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
5442 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
5443 \cs_new_protected:Npn \skipomgroup {
5444   \ifcase\l_document_structure_section_level_int
5445   \or\stepcounter{part}
5446   \or\stepcounter{chapter}
5447   \or\stepcounter{section}
5448   \or\stepcounter{subsection}
5449   \or\stepcounter{subsubsection}
5450   \or\stepcounter{paragraph}
5451   \or\stepcounter{subparagraph}
5452   \fi
5453 }
```

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

`blindomgroup`

```
5454 \newcommand\at@begin@blindomgroup[1]{%
5455 \newenvironment{blindomgroup}
5456 {
5457   \int_incr:N\l_document_structure_section_level_int
5458   \at@begin@blindomgroup\l_document_structure_section_level_int
5459 }{}}
```

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

```
5460 \newcommand\omgroup@nonum[2]{
5461   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
5462   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
5463 }
```

*(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.

```
5464 \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.

```

5465 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
5466   \@nameuse{#1}{#2}
5467 }{
5468   \cs_if_exist:NTF\rdfmata@sectioning{
5469     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
5470   }{
5471     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
5472   }
5473 }
5474 %\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\omgroup@id
5475 }

```

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

omgroup

```

5476 \keys_define:nn { document-structure / omgroup }{
5477   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
5478   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
5479   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
5480   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
5481   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
5482   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
5483   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
5484   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
5485   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
5486   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
5487 }
5488 \cs_new_protected:Nn \__document_structure_omgroup_args:n {
5489   \str_clear:N \l__document_structure_omgroup_id_str
5490   \str_clear:N \l__document_structure_omgroup_date_str
5491   \clist_clear:N \l__document_structure_omgroup_creators_clist
5492   \clist_clear:N \l__document_structure_omgroup_contributors_clist
5493   \tl_clear:N \l__document_structure_omgroup_srccite_tl
5494   \tl_clear:N \l__document_structure_omgroup_type_tl
5495   \tl_clear:N \l__document_structure_omgroup_short_tl
5496   \tl_clear:N \l__document_structure_omgroup_display_tl
5497   \tl_clear:N \l__document_structure_omgroup_intro_tl
5498   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
5499   \keys_set:nn { document-structure / omgroup } { #1 }
5500 }

```

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.

```

5501 \newif\if@mainmatter\@mainmattertrue
5502 \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.

```

5503 \keys_define:nn { document-structure / sectioning }{
5504   name .str_set_x:N = \l__document_structure_sect_name_str ,
5505   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
5506   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
5507   clear .default:n = {true} ,
5508   num .bool_set:N = \l__document_structure_sect_num_bool ,

```

```

5509   num      .default:n    = {true}
5510 }
5511 \cs_new_protected:Nn \__document_structure_sect_args:n {
5512   \str_clear:N \l__document_structure_sect_name_str
5513   \str_clear:N \l__document_structure_sect_ref_str
5514   \bool_set_false:N \l__document_structure_sect_clear_bool
5515   \bool_set_false:N \l__document_structure_sect_num_bool
5516   \keys_set:nn { document-structure / sectioning } { #1 }
5517 }
5518 \newcommand\omdoc@sectioning[3][]{
5519   \__document_structure_sect_args:n {#1}
5520   \let\omdoc@sect@name\l__document_structure_sect_name_str
5521   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
5522   \if@mainmatter% numbering not overridden by frontmatter, etc.
5523     \bool_if:NTF \l__document_structure_sect_num_bool {
5524       \omgroup@num{#2}{#3}
5525     }{
5526       \omgroup@nonum{#2}{#3}
5527     }
5528     \def\current@section@level{\omdoc@sect@name}
5529   \else
5530     \omgroup@nonum{#2}{#3}
5531   \fi
5532 }% 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.

```

5533 \newcommand\omgroup@redefine@addtocontents[1]{%
5534 %\edef\__document_structureimport{#1}%
5535 %\@for\@I:=\__document_structureimport\do{%
5536 %\edef\@path{\csname module@\@I @path\endcsname}%
5537 %\@ifundefined{tf@toc}\relax%
5538 %   {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
5539 %\ifx\hyper@anchor\undefined% hyperref.sty loaded?
5540 %\def\addcontentsline##1##2##3{%
5541 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
5542 %\else% hyperref.sty not loaded
5543 %\def\addcontentsline##1##2##3{%
5544 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
5545 %\fi
5546 }% hypreref.sty loaded?

```

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

```

5547 \newenvironment{omgroup}[2][]{% keys, title
5548 {
5549   \__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.

```

5550 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
5551   \omgroup@redefine@addtocontents{
5552     \@ifundefined{module@id}\used@modules%

```

```

5553     %{\@ifundefined{module@}\module@id @path}{\used@modules}\module@id}
5554   }
5555 }

```

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

```

5556 \int_incr:N\l_document_structure_section_level_int
5557 \ifcase\l_document_structure_section_level_int
5558   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
5559   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
5560   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
5561   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
5562   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
5563   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
5564   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
5565 \fi
5566 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
5567 \str_if_empty:NF \l__document_structure_omgroup_id_str {
5568   \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
5569 }
5570 }% for customization
5571 {}

```

and finally, we localize the sections

```

5572 \newcommand\omdoc@part@kw{Part}
5573 \newcommand\omdoc@chapter@kw{Chapter}
5574 \newcommand\omdoc@section@kw{Section}
5575 \newcommand\omdoc@subsection@kw{Subsection}
5576 \newcommand\omdoc@subsubsection@kw{Subsubsection}
5577 \newcommand\omdoc@paragraph@kw{paragraph}
5578 \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`

```

5579 \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).

```

5580 \cs_if_exist:NTF\frontmatter{
5581   \let\__document_structure_orig_frontmatter\frontmatter
5582   \let\frontmatter\relax
5583 }{
5584   \tl_set:Nn\__document_structure_orig_frontmatter{
5585     \clearpage
5586     \@mainmatterfalse
5587     \pagenumbering{roman}

```

```

5588 }
5589 }
5590 \cs_if_exist:NTF\backmatter{
5591   \let\__document_structure_orig_backmatter\backmatter
5592   \let\backmatter\relax
5593 }{
5594   \tl_set:Nn\__document_structure_orig_backmatter{
5595     \clearpage
5596     \@mainmatterfalse
5597     \pagenumbering{roman}
5598   }
5599 }

```

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

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

```

5600 \newenvironment{frontmatter}{
5601   \__document_structure_orig_frontmatter
5602 }{
5603   \cs_if_exist:NTF\mainmatter{
5604     \mainmatter
5605   }{
5606     \clearpage
5607     \@mainmattertrue
5608     \pagenumbering{arabic}
5609   }
5610 }

```

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

```

5611 \newenvironment{backmatter}{
5612   \__document_structure_orig_backmatter
5613 }{
5614   \cs_if_exist:NTF\mainmatter{
5615     \mainmatter
5616   }{
5617     \clearpage
5618     \@mainmattertrue
5619     \pagenumbering{arabic}
5620   }
5621 }

```

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

```

5622 \@mainmattertrue\pagenumbering{arabic}

```

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

```

5623 \def \c__document_structure_document_str{document}
5624 \newcommand\afterprematurestop{}
5625 \def\prematurestop@endomgroup{
5626   \unless\ifx\@currenvir\c__document_structure_document_str
5627     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter\expandafter\expandafter
5628     \expandafter\prematurestop@endomgroup

```



```

5629 \fi
5630 }
5631 \providecommand\prematurestop{
5632 \message{Stopping~sTeX~processing~prematurely}
5633 \prematurestop@endumgroup
5634 \afterprematurestop
5635 \end{document}
5636 }

```

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

## 38.8 Global Variables

**\setSGvar** set a global variable

```

5637 \RequirePackage{etoolbox}
5638 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

**\useSGvar** use a global variable

```

5639 \newrobustcmd\useSGvar[1]{%
5640 \@ifundefined{sTeX@Gvar@#1}
5641 {\PackageError{document-structure}
5642 {The sTeX Global variable #1 is undefined}
5643 {set it with \protect\setSGvar}}
5644 \@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.

```

5645 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
5646 \@ifundefined{sTeX@Gvar@#1}
5647 {\PackageError{document-structure}
5648 {The sTeX Global variable #1 is undefined}
5649 {set it with \protect\setSGvar}}
5650 {\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.

```
5651 \*cls)
5652 \@@=notesslides)
5653 \ProvidesExplClass{notesslides}{2022/02/10}{3.0}{notesslides Class}
5654 \RequirePackage{l3keys2e,expl-keystr-compatible}
5655
5656 \keys_define:nn{notesslides / cls}{
5657   class .code:n = {
5658     \PassOptionsToClass{\CurrentOption}{document-structure}
5659     \str_if_eq:nnT{#1}{book}{
5660       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5661     }
5662     \str_if_eq:nnT{#1}{report}{
5663       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5664     }
5665   },
5666   notes .bool_set:N = \c__notesslides_notes_bool ,
5667   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
5668   unknown .code:n = {
5669     \PassOptionsToClass{\CurrentOption}{document-structure}
5670     \PassOptionsToClass{\CurrentOption}{beamer}
5671     \PassOptionsToPackage{\CurrentOption}{notesslides}
5672   }
5673 }
5674 \ProcessKeysOptions{ notesslides / cls }
5675 \bool_if:NTF \c__notesslides_notes_bool {
5676   \PassOptionsToPackage{notes=true}{notesslides}
5677 }{
5678   \PassOptionsToPackage{notes=false}{notesslides}
5679 }
5680 \</cls)
```

now we do the same for the notesslides package.

```

5681 <*package>
5682 \ProvidesExplPackage{notesslides}{2022/02/10}{3.0}{notesslides Package}
5683 \RequirePackage{l3keys2e,expl-keystr-compat}
5684
5685 \keys_define:nn{notesslides / pkg}{
5686   topsect      .str_set_x:N = \c_notesslides_topsect_str,
5687   defaulttopsect .str_set_x:N = \c_notesslides_defaulttopsec_str,
5688   notes        .bool_set:N = \c_notesslides_notes_bool ,
5689   slides        .code:n      = { \bool_set_false:N \c_notesslides_notes_bool },
5690   sectocframes .bool_set:N = \c_notesslides_sectocframes_bool ,
5691   frameimages .bool_set:N = \c_notesslides_frameimages_bool ,
5692   fiboxed      .bool_set:N = \c_notesslides_fiboxed_bool ,
5693   noproblems   .bool_set:N = \c_notesslides_noproblems_bool,
5694   unknown      .code:n      = {
5695     \PassOptionsToClass{\CurrentOption}{stex}
5696     \PassOptionsToClass{\CurrentOption}{tikzinput}
5697   }
5698 }
5699 \ProcessKeysOptions{ notesslides / pkg }
5700 \newif\ifnotes
5701 \bool_if:NTF \c_notesslides_notes_bool {
5702   \notesttrue
5703 }{
5704   \notesfalse
5705 }
5706

```

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

```

5707 \str_if_empty:NTF \c_notesslides_topsect_str {
5708   \str_set_eq:NN \__notesslidestopsect \c_notesslides_defaulttopsec_str
5709 }{
5710   \str_set_eq:NN \__notesslidestopsect \c_notesslides_topsect_str
5711 }
5712 </package>

```

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

```

5713 <*cls>
5714 \bool_if:NTF \c_notesslides_notes_bool {
5715   \LoadClass{document-structure}
5716 }{
5717   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
5718   \newcounter{Item}
5719   \newcounter{paragraph}
5720   \newcounter{subparagraph}
5721   \newcounter{Hfootnote}
5722   \RequirePackage{document-structure}
5723 }

```

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

```

5724 \RequirePackage{notesslides}
5725 </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.

```

5726 \*package>
5727 \bool_if:NT \c__notesslides_notes_bool {
5728   \RequirePackage{a4wide}
5729   \RequirePackage{marginnote}
5730   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
5731   \RequirePackage{mdframed}
5732   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
5733   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
5734 }
5735 \RequirePackage{stex-tikzinput}
5736 \RequirePackage{etoolbox}
5737 \RequirePackage{amssymb}
5738 \RequirePackage{amsmath}
5739 \RequirePackage{comment}
5740 \RequirePackage{textcomp}
5741 \RequirePackage{url}
5742 \RequirePackage{graphicx}
5743 \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>

```

5744 \bool_if:NT \c__notesslides_notes_bool {
5745   \renewcommand\usetheme[2][\usepackage[#1]{beamernotestheme#2}]
5746 }

```

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

```

5747 \newcounter{slide}
5748 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
5749 \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.

```

5750 \bool_if:NTF \c__notesslides_notes_bool {
5751   \renewenvironment{note}{\ignorespaces}{}
5752 }{
5753   \excludecomment{note}
5754 }

```

---

<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.

```
5755 \bool_if:NT \c__notesslides_notes_bool {
5756   \newlength{\slideframewidth}
5757   \setlength{\slideframewidth}{1.5pt}
```

**frame** We first define the keys.

```
5758 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
5759   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
5760     \bool_set_true:N #1
5761   }{
5762     \bool_set_false:N #1
5763   }
5764 }
5765 \keys_define:nn{notesslides / frame}{
5766   label .str_set_x:N = \l__notesslides_frame_label_str,
5767   allowframebreaks .code:n = {
5768     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
5769   },
5770   allowdisplaybreaks .code:n = {
5771     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
5772   },
5773   fragile .code:n = {
5774     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
5775   },
5776   shrink .code:n = {
5777     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
5778   },
5779   squeeze .code:n = {
5780     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
5781   },
5782   t .code:n = {
5783     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
5784   },
5785 }
5786 \cs_new_protected:Nn \__notesslides_frame_args:n {
5787   \str_clear:N \l__notesslides_frame_label_str
5788   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
5789   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
5790   \bool_set_true:N \l__notesslides_frame_fragile_bool
5791   \bool_set_true:N \l__notesslides_frame_shrink_bool
5792   \bool_set_true:N \l__notesslides_frame_squeeze_bool
5793   \bool_set_true:N \l__notesslides_frame_t_bool
5794   \keys_set:nn { notesslides / frame }{ #1 }
5795 }
```

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

```
5796 \renewenvironment{frame}[1][]{
5797   \__notesslides_frame_args:n{#1}
5798   \sffamily
5799   \stepcounter{slide}
5800   \def\@currentlabel{\theslide}
5801   \str_if_empty:NF \l__notesslides_frame_label_str {
5802     \label{\l__notesslides_frame_label_str}
```

5803 }

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

5804 \def\itemize@level{outer}  
5805 \def\itemize@outer{outer}  
5806 \def\itemize@inner{inner}  
5807 \renewcommand\newpage{\addtocounter{framenum}{1}}  
5808 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}  
5809 \renewenvironment{itemize}{  
5810 \ifx\itemize@level\itemize@outer  
5811 \def\itemize@label{\\$ \rhd\$}  
5812 \fi  
5813 \ifx\itemize@level\itemize@inner  
5814 \def\itemize@label{\\$ \scriptstyle \rhd\$}  
5815 \fi  
5816 \begin{list}  
5817 {\itemize@label}  
5818 {\setlength{\labelsep}{.3em}  
5819 \setlength{\labelwidth}{.5em}  
5820 \setlength{\leftmargin}{1.5em}  
5821 }  
5822 \edef\itemize@level{\itemize@inner}  
5823 }{  
5824 \end{list}  
5825 }

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

5826 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth=100pt]  
5827 }{  
5828 \medskip\miko@slidelabel\end{mdframed}  
5829 }

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

\frametitle

5830 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip  
5831 }

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

EdN:21

\pause 21

5832 \bool\_if:NT \c\_\_notesslides\_notes\_bool {  
5833 \newcommand\pause{  
5834 }

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

nparagraph

5835 \bool\_if:NTF \c\_\_notesslides\_notes\_bool {  
5836 \newenvironment{nparagraph}[1][\begin{sparagraph}[#1]}{\end{sparagraph}}  
5837 }{  
5838 \excludecomment{nparagraph}  
5839 }

---

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

nomgroup

```

5840 \bool_if:NTF \c__notesslides_notes_bool {
5841   \newenvironment{nomgroup}[2] [] {\begin{omgroup}[#1]{#2}}{\end{omgroup}}
5842 }{
5843   \excludecomment{nomgroup}
5844 }

```

ndefinition

```

5845 \bool_if:NTF \c__notesslides_notes_bool {
5846   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}{\end{sdefinition}}
5847 }{
5848   \excludecomment{ndefinition}
5849 }

```

nassertion

```

5850 \bool_if:NTF \c__notesslides_notes_bool {
5851   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}{\end{sassertion}}
5852 }{
5853   \excludecomment{nassertion}
5854 }

```

nsproof

```

5855 \bool_if:NTF \c__notesslides_notes_bool {
5856   \newenvironment{nproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
5857 }{
5858   \excludecomment{nproof}
5859 }

```

nexample

```

5860 \bool_if:NTF \c__notesslides_notes_bool {
5861   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}{\end{sexample}}
5862 }{
5863   \excludecomment{nexample}
5864 }

```

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

```

5865 \def\inputref@preskip{\smallskip}
5866 \def\inputref@postskip{\medskip}

```

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

\inputref\*

```

5867 \let\orig@inputref\inputref
5868 \def\inputref{\@ifstar\ninputref\orig@inputref}
5869 \newcommand\ninputref[2] [] {
5870   \bool_if:NT \c__notesslides_notes_bool {
5871     \orig@inputref[#1]{#2}
5872   }
5873 }

```

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

## 39.3 Header and Footer Lines

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

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

```
5874 \newlength{\slidelogoheight}
5875
5876 \bool_if:NTF \c__notesslides_notes_bool {
5877   \setlength{\slidelogoheight}{.4cm}
5878 }{
5879   \setlength{\slidelogoheight}{1cm}
5880 }
5881 \newsavebox{\slidelogo}
5882 \sbox{\slidelogo}{\TeX}
5883 \newrobustcmd{\setslidelogo}[1]{
5884   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
5885 }
```

(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.

```
5886 \def\source{Michael Kohlhase}% customize locally
5887 \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.

```
5888 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
5889 \newsavebox{\cclogo}
5890 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
5891 \newif\ifcchref\cchreffalse
5892 \AtBeginDocument{
5893   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
5894 }
5895 \def\licensing{
5896   \ifcchref
5897     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
5898   \else
5899     {\usebox{\cclogo}}
5900   \fi
5901 }
5902 \newrobustcmd{\setlicensing}[2][]{
5903   \def@url{#1}
5904   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
5905   \ifx@url@empty
5906     \def\licensing{{\usebox{\cclogo}}}
5907   \else
5908     \def\licensing{
```



```

5909     \ifcchref
5910     \href{#1}{\usebox{\cclogo}}
5911     \else
5912     {\usebox{\cclogo}}
5913     \fi
5914   }
5915 \fi
5916 }

```

(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>

```

5917 \newrobustcmd\miko@slidelabel{
5918   \vbox to \slidelogoheight{
5919     \vss\hbox to \slidewidth
5920     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
5921   }
5922 }

```

(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.

```

5923 \def\Gin@mhrepos{}
5924 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
5925 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
5926 \newrobustcmd\frameimage[2][]{
5927   \stepcounter{slide}
5928   \bool_if:NT \c__notesslides_frameimages_bool {
5929     \def\Gin@ewidth{}\setkeys{Gin}{#1}
5930     \bool_if:NF \c__notesslides_notes_bool { \vfill }
5931     \begin{center}
5932       \bool_if:NTF \c__notesslides_fiboxed_bool {
5933         \fbox{
5934           \ifx\Gin@ewidth\@empty
5935             \ifx\Gin@mhrepos\@empty
5936               \mhgraphics[width=\slidewidth,#1]{#2}
5937             \else
5938               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5939             \fi
5940           \else% Gin@ewidth empty
5941             \ifx\Gin@mhrepos\@empty
5942               \mhgraphics[#1]{#2}
5943             \else
5944               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5945             \fi
5946           \fi% Gin@ewidth empty
5947         }
5948       }{
5949         \ifx\Gin@ewidth\@empty

```

---

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

```

5950         \ifx\Gin@mhrepos\empty
5951           \mhgraphics[width=\slidewidth,#1]{#2}
5952         \else
5953           \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5954         \fi
5955         \ifx\Gin@mhrepos\empty
5956           \mhgraphics[#1]{#2}
5957         \else
5958           \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5959         \fi
5960       \fi% Gin@ewidth empty
5961     }
5962   \end{center}
5963   \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
5964   \bool_if:NF \c__notesslides_notes_bool { \vfill }
5965 }
5966 } % 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.

```

5967 \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

```

5968 \AddToHook{begindocument}{
5969   \definecolor{green}{rgb}{0,.5,0}
5970   \definecolor{purple}{cmyk}{.3,1,0,.17}
5971 }

```

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`.

```

5972 % \def\STpresent#1{\textcolor{blue}{#1}}
5973 \def\defemph#1{\textcolor{magenta}{#1}}
5974 \def\symrefemph#1{\textcolor{cyan}{#1}}
5975 \def\compemph#1{\textcolor{blue}{#1}}
5976 \def\titleemph#1{\textcolor{blue}{#1}}
5977 \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.

```

5978 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
5979 \def\smalltextwarning{
5980   \pgfuseimage{miko@small@dbend}
5981   \xspace
5982 }
5983 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}

```

```

5984 \newrobustcmd\textwarning{
5985   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
5986   \xspace
5987 }
5988 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}
5989 \newrobustcmd\bigtextwarning{
5990   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
5991   \xspace
5992 }

```

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

```

5993 \newrobustcmd\putgraphicsat[3]{
5994   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
5995 }
5996 \newrobustcmd\putat[2]{
5997   \begin{picture}(0,0)\put(#1){#2}\end{picture}
5998 }

```

## 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`.

```

5999 \bool_if:NT \c__notesslides_sectocframes_bool {
6000   \str_if_eq:VnTF \__notesslidesstopsect{part}{
6001     \newcounter{chapter}\counterwithin*{section}{chapter}
6002   }{
6003     \str_if_eq:VnT\__notesslidesstopsect{chapter}{
6004       \newcounter{chapter}\counterwithin*{section}{chapter}
6005     }
6006   }
6007 }

```

`\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
6008 \def\part@prefix{}
6009 \@ifpackageloaded{document-structure}{\{
6010   \str_case:VnF \__notesslidesstopsect {
6011     {part}{
6012       \int_set:Nn \l_document_structure_section_level_int {0}
6013       \def\thesection{\arabic{chapter}.\arabic{section}}
6014       \def\part@prefix{\arabic{chapter}.}
6015     }
6016     {chapter}{
6017       \int_set:Nn \l_document_structure_section_level_int {1}
6018       \def\thesection{\arabic{chapter}.\arabic{section}}
6019       \def\part@prefix{\arabic{chapter}.}
6020     }
6021   }{
6022     \int_set:Nn \l_document_structure_section_level_int {2}
6023     \def\part@prefix{}

```

```

6024 }
6025 }
6026
6027 \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

```

6028 \renewenvironment{omgroup}[2][]{
6029   \__document_structure_omgroup_args:n { #1 }
6030   \int_incr:N \l_document_structure_section_level_int
6031   \bool_if:NT \c__notesslides_sectocframes_bool {
6032     \stepcounter{slide}
6033     \begin{frame}[noframenumbering]
6034       \vfill\Large\centering
6035       \red{
6036         \ifcase\l_document_structure_section_level_int\or
6037           \stepcounter{part}
6038           \def\__notesslideslabel{\omdoc@part@kw~\Roman{part}}
6039           \def\currentsectionlevel{\omdoc@part@kw}
6040         \or
6041           \stepcounter{chapter}
6042           \def\__notesslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
6043           \def\currentsectionlevel{\omdoc@chapter@kw}
6044         \or
6045           \stepcounter{section}
6046           \def\__notesslideslabel{\part@prefix\arabic{section}}
6047           \def\currentsectionlevel{\omdoc@section@kw}
6048         \or
6049           \stepcounter{subsection}
6050           \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
6051           \def\currentsectionlevel{\omdoc@subsection@kw}
6052         \or
6053           \stepcounter{subsubsection}
6054           \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
6055           \def\currentsectionlevel{\omdoc@subsubsection@kw}
6056         \or
6057           \stepcounter{paragraph}
6058           \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{paragraph}}
6059           \def\currentsectionlevel{\omdoc@paragraph@kw}
6060         \else
6061           \def\__notesslideslabel{}
6062           \def\currentsectionlevel{\omdoc@paragraph@kw}
6063         \fi% end ifcase
6064         \__notesslideslabel%\sref@label@id\__notesslideslabel
6065         \quad #2%
6066       }%
6067       \vfill%
6068     \end{frame}%
6069   }
6070   \str_if_empty:NF \l__document_structure_omgroup_id_str {
6071     \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str

```

```

6072     }
6073   }{}
6074 }

```

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

```

6075 \def\inserttheorembodyfont{\normalfont}
6076 %\bool_if:NF \c__notesslides_notes_bool {
6077 %   \defbeamertemplate{theorem begin}{miko}
6078 %   {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
6079 %     \ifx\inserttheoremaddition@empty\else\ (\inserttheoremaddition)\fi%
6080 %     \inserttheorempunctuation\inserttheorembodyfont\hspace}
6081 %   \defbeamertemplate{theorem end}{miko}{}

```

and we set it as the default one.

```

6082 % \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.

```

6083 % \expandafter\def\csname Parent2\endcsname{}
6084 %}
6085
6086 \AddToHook{begindocument}{ % this does not work for some reason
6087   \setbeamertemplate{theorems}[ams style]
6088 }
6089 \bool_if:NT \c__notesslides_notes_bool {
6090   \renewenvironment{columns}[1][ ]{%
6091     \par\noindent%
6092     \begin{minipage}%
6093       \slidewidth\centering\leavevmode%
6094   }{%
6095     \end{minipage}\par\noindent%
6096   }%
6097   \newsavebox\columnbox%
6098   \renewenvironment<>{column}[2][ ]{%
6099     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
6100   }{%
6101     \end{minipage}\end{lrbox}\usebox\columnbox%
6102   }%
6103 }
6104 \bool_if:NTF \c__notesslides_noproblems_bool {
6105   \newenvironment{problems}{}{}
6106 }{
6107   \excludacomment{problems}
6108 }

```

## 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.

```

6109 \gdef\printexcursions{}
6110 \newcommand\excursionref[2]{% label, text

```

```

6111 \bool_if:NT \c__notesslides_notes_bool {
6112   \begin{sparagraph}[title=Excursion]
6113     #2 \sref[fallback=the appendix]{#1}.
6114   \end{sparagraph}
6115 }
6116 }
6117 \newcommand\activate@excursion[2][]{
6118   \gappto\printexcursions{\inputref[#1]{#2}}
6119 }
6120 \newcommand\excursion[4][]{% repos, label, path, text
6121   \bool_if:NT \c__notesslides_notes_bool {
6122     \activate@excursion[#1]{#3}\excursionref{#2}{#4}
6123   }
6124 }

```

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

#### \excursiongroup

```

6125 \keys_define:nn{notesslides / excursiongroup }{
6126   id          .str_set_x:N = \l__notesslides_excursion_id_str,
6127   intro       .tl_set:N   = \l__notesslides_excursion_intro_tl,
6128   mhrepos     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
6129 }
6130 \cs_new_protected:Nn \__notesslides_excursion_args:n {
6131   \tl_clear:N \l__notesslides_excursion_intro_tl
6132   \str_clear:N \l__notesslides_excursion_id_str
6133   \str_clear:N \l__notesslides_excursion_mhrepos_str
6134   \keys_set:nn {notesslides / excursiongroup }{ #1 }
6135 }
6136 \newcommand\excursiongroup[1][]{
6137   \__notesslides_excursion_args:n{ #1 }
6138   \ifdefempty\printexcursions{}% only if there are excursions
6139   {\begin{note}
6140     \begin{omgroup}[#1]{Excursions}%
6141       \ifdefempty\l__notesslides_excursion_intro_tl{
6142         \inputref[\l__notesslides_excursion_mhrepos_str]{
6143           \l__notesslides_excursion_intro_tl
6144         }
6145       }
6146       \printexcursions%
6147     \end{omgroup}
6148   \end{note}}
6149 }
6150 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{\fi
6151 \</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.

```
6152 <*package>
6153 <@@=problems>
6154 \ProvidesExplPackage{problem}{2019/03/20}{1.3}{Semantic Markup for Problems}
6155 \RequirePackage{l3keys2e,expl-keystr-compatible}
6156
6157 \keys_define:nn { problem / pkg }{
6158   notes      .default:n    = { true },
6159   notes      .bool_set:N   = \c__problems_notes_bool,
6160   gnotes     .default:n    = { true },
6161   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
6162   hints      .default:n    = { true },
6163   hints      .bool_set:N   = \c__problems_hints_bool,
6164   solutions  .default:n    = { true },
6165   solutions  .bool_set:N   = \c__problems_solutions_bool,
6166   pts        .default:n    = { true },
6167   pts        .bool_set:N   = \c__problems_pts_bool,
6168   min        .default:n    = { true },
6169   min        .bool_set:N   = \c__problems_min_bool,
6170   boxed      .default:n    = { true },
6171   boxed      .bool_set:N   = \c__problems_boxed_bool,
6172   unknown    .code:n       = {}
6173 }
6174 \newif\ifsolutions
6175
6176 \ProcessKeysOptions{ problem / pkg }
6177 \bool_if:NTF \c__problems_solutions_bool {
6178   \solutionstrue
6179 }{
6180   \solutionsfalse
6181 }
```

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

```
6182 \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.

```
6183 \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.

```
6184 \def\prob@problem@kw{Problem}
6185 \def\prob@solution@kw{Solution}
6186 \def\prob@hint@kw{Hint}
6187 \def\prob@note@kw{Note}
6188 \def\prob@gnote@kw{Grading}
6189 \def\prob@pt@kw{pt}
6190 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
6191 \AddToHook{begindocument}{
6192   \ltx@ifpackageloaded{babel}{
6193     \makeatletter
6194     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6195     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6196       \input{problem-ngerman.ldf}
6197     }
6198     \clist_if_in:NnT \l_tmpa_clist {finnish}{
6199       \input{problem-finnish.ldf}
6200     }
6201     \clist_if_in:NnT \l_tmpa_clist {french}{
6202       \input{problem-french.ldf}
6203     }
6204     \clist_if_in:NnT \l_tmpa_clist {russian}{
6205       \input{problem-russian.ldf}
6206     }
6207     \makeatother
6208   }{ }
6209 }
```

## 40.2 Problems and Solutions

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

```
6210 \keys_define:nn{ problem / problem }{
6211   id      .str_set:x:N = \l__problems_prob_id_str,
6212   pts     .tl_set:N    = \l__problems_prob_pts_tl,
6213   min     .tl_set:N    = \l__problems_prob_min_tl,
6214   title   .tl_set:N    = \l__problems_prob_title_tl,
6215   type    .tl_set:N    = \l__problems_prob_type_tl,
6216   refnum  .int_set:N    = \l__problems_prob_refnum_int
6217 }
6218 \cs_new_protected:Nn \__problems_prob_args:n {
```



```

6219 \str_clear:N \l__problems_prob_id_str
6220 \tl_clear:N \l__problems_prob_pts_tl
6221 \tl_clear:N \l__problems_prob_min_tl
6222 \tl_clear:N \l__problems_prob_title_tl
6223 \tl_clear:N \l__problems_prob_type_tl
6224 \int_zero_new:N \l__problems_prob_refnum_int
6225 \keys_set:nn { problem / problem }{ #1 }
6226 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
6227   \let\l__problems_prob_refnum_int\undefined
6228 }
6229 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

6230 \newcounter{problem}
6231 \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.

```

6232 \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

```

6233 \newcommand\prob@number{
6234   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
6235     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
6236   }{
6237     \int_if_exist:NTF \l__problems_prob_refnum_int {
6238       \prob@label{\int_use:N \l__problems_prob_refnum_int }
6239     }{
6240       \prob@label\theproblem
6241     }
6242   }
6243 }

```

(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.

```

6244 \newcommand\prob@title[3]{%
6245   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
6246     #2 \l__problems_inclprob_title_tl #3
6247   }{
6248     \tl_if_exist:NTF \l__problems_prob_title_tl {
6249       #2 \l__problems_prob_title_tl #3
6250     }{
6251       #1
6252     }
6253   }
6254 }

```

(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.

```

6255 \def\prob@heading{
6256   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
6257   %\sref@label{id{\prob@problem@kw~\prob@number}}{~}
6258 }

```

(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`

```

6259 \newenvironment{sproblem}[1][{}]{
6260   \__problems_prob_args:n{#1}%\sref@target%
6261   \@in@omtexttrue% we are in a statement (for inline definitions)
6262   \stepcounter{problem}\record@problem
6263   \def\current@section@level{\prob@problem@kw}
6264   \tl_if_exist:NTF \l__problems_inclprob_type_tl {
6265     \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
6266   }{
6267     \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
6268   }
6269   \str_if_exist:NTF \l__problems_inclprob_id_str {
6270     \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
6271   }{
6272     \str_set_eq:NN \sproblemid \l__problems_prob_id_str
6273   }
6274
6275
6276   \clist_set:No \l_tmpa_clist \sproblemtype
6277   \tl_clear:N \l_tmpa_tl
6278   \clist_map_inline:Nn \l_tmpa_clist {
6279     \tl_if_exist:cT {\__problems_sproblem_##1_start:}{
6280       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_start:}}
6281     }
6282   }
6283   \tl_if_empty:NTF \l_tmpa_tl {
6284     \__problems_sproblem_start:
6285   }{
6286     \l_tmpa_tl
6287   }
6288   \stex_ref_new_doc_target:n \sproblemid
6289 }{
6290   \clist_set:No \l_tmpa_clist \sproblemtype
6291   \tl_clear:N \l_tmpa_tl
6292   \clist_map_inline:Nn \l_tmpa_clist {
6293     \tl_if_exist:cT {\__problems_sproblem_##1_end:}{
6294       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_end:}}
6295     }

```

```

6296 }
6297 \tl_if_empty:NTF \l_tmpa_tl {
6298   \__problems_sproblem_end:
6299 }{
6300   \l_tmpa_tl
6301 }
6302
6303
6304 \smallskip
6305 }
6306
6307
6308 \cs_new_protected:Nn \__problems_sproblem_start: {
6309   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
6310 }
6311 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
6312
6313 \newcommand\stexpatchproblem[3][] {
6314   \str_set:Nx \l_tmpa_str{ #1 }
6315   \str_if_empty:NTF \l_tmpa_str {
6316     \tl_set:Nn \__problems_sproblem_start: { #2 }
6317     \tl_set:Nn \__problems_sproblem_end: { #3 }
6318   }{
6319     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
6320     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
6321   }
6322 }
6323
6324
6325 \bool_if:NT \c__problems_boxed_bool {
6326   \surroundwithmdframed{problem}
6327 }

```

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

```

6328 \def\record@problem{
6329   \protected@write\@auxout{}
6330   {
6331     \string\@problem{\prob@number}
6332     {
6333       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
6334         \l__problems_inclprob_pts_tl
6335       }{
6336         \l__problems_prob_pts_tl
6337       }
6338     }%
6339     {
6340       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
6341         \l__problems_inclprob_min_tl
6342       }{
6343         \l__problems_prob_min_tl
6344       }
6345     }
6346   }
6347 }

```

(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).

```
6348 \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.

```
6349 \keys_define:nn { problem / solution }{
6350   id                .str_set_x:N = \l__problems_solution_id_str ,
6351   for               .tl_set:N    = \l__problems_solution_for_tl ,
6352   height            .dim_set:N   = \l__problems_solution_height_dim ,
6353   creators           .clist_set:N = \l__problems_solution_creators_clist ,
6354   contributors       .clist_set:N = \l__problems_solution_contributors_clist ,
6355   srccite            .tl_set:N    = \l__problems_solution_srccite_tl
6356 }
6357 \cs_new_protected:Nn \__problems_solution_args:n {
6358   \str_clear:N \l__problems_solution_id_str
6359   \tl_clear:N \l__problems_solution_for_tl
6360   \tl_clear:N \l__problems_solution_srccite_tl
6361   \clist_clear:N \l__problems_solution_creators_clist
6362   \clist_clear:N \l__problems_solution_contributors_clist
6363   \dim_zero:N \l__problems_solution_height_dim
6364   \keys_set:nn { problem / solution }{ #1 }
6365 }
```

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

```
6366 \newcommand\@startsolution[1][{}]{
6367   \__problems_solution_args:n { #1 }
6368   \@in@omtexttrue% we are in a statement.
6369   \bool_if:NF \c__problems_boxed_bool { \hrule }
6370   \smallskip\noindent
6371   {\textbf\prob@solution@kw : \enspace}
6372   \begin{small}
6373   \def\current@section@level{\prob@solution@kw}
6374   \ignorespacesandpars
6375 }
```

**\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.

```
6376 \newcommand\startsolutions{
6377   \specialcomment{solution}{\@startsolution}{
6378     \bool_if:NF \c__problems_boxed_bool {
6379       \hrule\medskip
6380     }
6381     \end{small}%
6382   }
6383   \bool_if:NT \c__problems_boxed_bool {
6384     \surroundwithmdframed{solution}
6385   }
6386 }
```

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

\stopsolutions

```
6387 \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.

```
6388 \ifsolutions
6389 \startsolutions
6390 \else
6391 \stopsolutions
6392 \fi
```

exnote

```
6393 \bool_if:NTF \c__problems_notes_bool {
6394 \newenvironment{exnote}[1][]{
6395 \par\smallskip\hrule\smallskip
6396 \noindent\textbf{\prob@note@kw : }\small
6397 }{
6398 \smallskip\hrule
6399 }
6400 }{
6401 \excludecomment{exnote}
6402 }
```

hint

```
6403 \bool_if:NTF \c__problems_notes_bool {
6404 \newenvironment{hint}[1][]{
6405 \par\smallskip\hrule\smallskip
6406 \noindent\textbf{\prob@hint@kw :~ }\small
6407 }{
6408 \smallskip\hrule
6409 }
6410 \newenvironment{exhint}[1][]{
6411 \par\smallskip\hrule\smallskip
6412 \noindent\textbf{\prob@hint@kw :~ }\small
6413 }{
6414 \smallskip\hrule
6415 }
6416 }{
6417 \excludecomment{hint}
6418 \excludecomment{exhint}
6419 }
```

gnote

```
6420 \bool_if:NTF \c__problems_notes_bool {
6421 \newenvironment{gnote}[1][]{
6422 \par\smallskip\hrule\smallskip
6423 \noindent\textbf{\prob@gnote@kw : }\small
6424 }{
6425 \smallskip\hrule
6426 }
6427 }{
6428 \excludecomment{gnote}
6429 }
```

## 40.3 Multiple Choice Blocks

EdN:23

mcb 23

```
6430 \newenvironment{mcb}{
6431   \begin{enumerate}
6432 }{
6433   \end{enumerate}
6434 }
```

we define the keys for the mcc macro

```
6435 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
6436   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
6437     \bool_set_true:N #1
6438   }{
6439     \bool_set_false:N #1
6440   }
6441 }
6442 \keys_define:nn { problem / mcc }{
6443   id          .str_set:x:N = \l__problems_mcc_id_str ,
6444   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
6445   T           .default:n    = { true } ,
6446   T           .bool_set:N   = \l__problems_mcc_t_bool ,
6447   F           .default:n    = { true } ,
6448   F           .bool_set:N   = \l__problems_mcc_f_bool ,
6449   Ttext       .code:n       = {
6450     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
6451   } ,
6452   Ftext       .code:n       = {
6453     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
6454   }
6455 }
6456 \cs_new_protected:Nn \l__problems_mcc_args:n {
6457   \str_clear:N \l__problems_mcc_id_str
6458   \tl_clear:N \l__problems_mcc_feedback_tl
6459   \bool_set_true:N \l__problems_mcc_t_bool
6460   \bool_set_true:N \l__problems_mcc_f_bool
6461   \bool_set_true:N \l__problems_mcc_Ttext_bool
6462   \bool_set_false:N \l__problems_mcc_Ftext_bool
6463   \keys_set:nn { problem / mcc }{ #1 }
6464 }
```

\mcc

```
6465 \newcommand\mcc[2][] {
6466   \l__problems_mcc_args:n{ #1 }
6467   \item #2
6468   \ifsolutions
6469     \\\
6470     \bool_if:NT \l__problems_mcc_t_bool {
6471       % TODO!
6472       % \ifcsstring{mcc@T}{T}{\mcc@Ttext}%
6473     }
6474     \bool_if:NT \l__problems_mcc_f_bool {
```

---

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

```

6475      % TODO!
6476      % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
6477    }
6478    \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
6479      !
6480    }{
6481      \l__problems_mcc_feedback_tl
6482    }
6483    \fi
6484  } %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.

```

6485
6486 \keys_define:nn{ problem / inclproblem }{
6487   id      .str_set:N = \l__problems_inclprob_id_str,
6488   pts     .tl_set:N  = \l__problems_inclprob_pts_tl,
6489   min     .tl_set:N  = \l__problems_inclprob_min_tl,
6490   title   .tl_set:N  = \l__problems_inclprob_title_tl,
6491   refnum  .int_set:N  = \l__problems_inclprob_refnum_int,
6492   type    .tl_set:N  = \l__problems_inclprob_type_tl,
6493   mhrepos .str_set:N  = \l__problems_inclprob_mhrepos_str
6494 }
6495 \cs_new_protected:Nn \l__problems_inclprob_args:n {
6496   \str_clear:N \l__problems_prob_id_str
6497   \tl_clear:N \l__problems_inclprob_pts_tl
6498   \tl_clear:N \l__problems_inclprob_min_tl
6499   \tl_clear:N \l__problems_inclprob_title_tl
6500   \tl_clear:N \l__problems_inclprob_type_tl
6501   \int_zero_new:N \l__problems_inclprob_refnum_int
6502   \str_clear:N \l__problems_inclprob_mhrepos_str
6503   \keys_set:nn { problem / inclproblem }{ #1 }
6504   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
6505     \let\l__problems_inclprob_pts_tl\undefined
6506   }
6507   \tl_if_empty:NT \l__problems_inclprob_min_tl {
6508     \let\l__problems_inclprob_min_tl\undefined
6509   }
6510   \tl_if_empty:NT \l__problems_inclprob_title_tl {
6511     \let\l__problems_inclprob_title_tl\undefined
6512   }
6513   \tl_if_empty:NT \l__problems_inclprob_type_tl {
6514     \let\l__problems_inclprob_type_tl\undefined
6515   }
6516   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
6517     \let\l__problems_inclprob_refnum_int\undefined
6518   }
6519 }

```

```

6520
6521 \cs_new_protected:Nn \__problems_inclprob_clear: {
6522   \let\l__problems_inclprob_id_str\undefined
6523   \let\l__problems_inclprob_pts_tl\undefined
6524   \let\l__problems_inclprob_min_tl\undefined
6525   \let\l__problems_inclprob_title_tl\undefined
6526   \let\l__problems_inclprob_type_tl\undefined
6527   \let\l__problems_inclprob_refnum_int\undefined
6528   \let\l__problems_inclprob_mhrepos_str\undefined
6529 }
6530 \__problems_inclprob_clear:
6531
6532 \newcommand\includeproblem[2][ ]{
6533   \__problems_inclprob_args:n{ #1 }
6534   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
6535     \input{#2}
6536   }{
6537     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
6538       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
6539     }
6540   }
6541   \__problems_inclprob_clear:
6542 }

```

(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.

```

6543 \AddToHook{enddocument}{
6544   \bool_if:NT \c__problems_pts_bool {
6545     \message{Total:~\arabic{pts}~points}
6546   }
6547   \bool_if:NT \c__problems_min_bool {
6548     \message{Total:~\arabic{min}~minutes}
6549   }
6550 }

```

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

```

6551 \def\pts#1{
6552   \bool_if:NT \c__problems_pts_bool {
6553     \marginpar{#1~\prob@pt@kw}
6554   }
6555 }
6556 \def\min#1{
6557   \bool_if:NT \c__problems_min_bool {
6558     \marginpar{#1~\prob@min@kw}
6559   }
6560 }

```



`\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.

```

6561 \newcounter{pts}
6562 \def\show@pts{
6563   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
6564     \bool_if:NT \c__problems_pts_bool {
6565       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
6566       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
6567     }
6568   }{
6569     \tl_if_exist:NT \l__problems_prob_pts_tl {
6570       \bool_if:NT \c__problems_pts_bool {
6571         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
6572         \addtocounter{pts}{\l__problems_prob_pts_tl}
6573       }
6574     }
6575   }
6576 }

```

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

`\show@min`

```

6577 \newcounter{min}
6578 \def\show@min{
6579   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
6580     \bool_if:NT \c__problems_min_bool {
6581       \marginpar{\l__problems_inclprob_min_tl\ min}
6582       \addtocounter{min}{\l__problems_inclprob_min_tl}
6583     }
6584   }{
6585     \tl_if_exist:NT \l__problems_prob_min_tl {
6586       \bool_if:NT \c__problems_min_bool {
6587         \marginpar{\l__problems_prob_min_tl\ min}
6588         \addtocounter{min}{\l__problems_prob_min_tl}
6589       }
6590     }
6591   }
6592 }
6593 \</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.

```
6594 \@@=hwexam>
6595 \*cls>
6596 \ProvidesExplClass{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
6597 \RequirePackage{l3keys2e,expl-keystr-compatible}
6598 \DeclareOption*{
6599   \PassOptionsToClass{\CurrentOption}{document-structure}
6600   \PassOptionsToPackage{\CurrentOption}{stex}
6601   \PassOptionsToPackage{\CurrentOption}{hwexam}
6602   \PassOptionsToPackage{\CurrentOption}{tikzinput}
6603 }
6604 \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.

```
6605 \LoadClass{document-structure}
6606 \RequirePackage{stex}
6607 \RequirePackage{hwexam}
6608 \RequirePackage{tikzinput}
6609 \RequirePackage{graphicx}
6610 \RequirePackage{a4wide}
6611 \RequirePackage{amssymb}
6612 \RequirePackage{amstext}
6613 \RequirePackage{amsmath}
```

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

```

6614 \newcommand\assig@default@type{\hwexam@assignment@kw}
6615 \def\document@hwexamtype{\assig@default@type}
6616 <@@=document_structure>
6617 \keys_define:nn { document-structure / document }{
6618 id .str_set_x:N = \c_document_structure_document_id_str,
6619 hwexamtype .tl_set:N = \document@hwexamtype
6620 }
6621 <@@=hwexam>
6622 </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.

```
6623 \*package>
6624 \ProvidesExplPackage{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
6625 \RequirePackage{l3keys2e,expl-keystr-compat}
6626
6627 \newif\iftest\testfalse
6628 \DeclareOption{test}{\testtrue}
6629 \newif\ifmultiple\multiplefalse
6630 \DeclareOption{multiple}{\multipletrue}
6631 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
6632 \ProcessOptions
```

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

```
6633 \RequirePackage{keyval}[1997/11/10]
6634 \RequirePackage{problem}
```

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

```
6635 \newcommand\hwexam@assignment@kw{Assignment}
6636 \newcommand\hwexam@given@kw{Given}
6637 \newcommand\hwexam@due@kw{Due}
6638 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
6639 blank~for~extra~space}
6640 \def\hwexam@minutes@kw{minutes}
6641 \newcommand\correction@probs@kw{prob.}
6642 \newcommand\correction@pts@kw{total}
6643 \newcommand\correction@reached@kw{reached}
6644 \newcommand\correction@sum@kw{Sum}
6645 \newcommand\correction@grade@kw{grade}
6646 \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

```

6647 \AddToHook{begindocument}{
6648 \ltx@ifpackageloaded{babel}{
6649 \makeatletter
6650 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6651 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6652 \input{hwexam-ngerman.ldf}
6653 }
6654 \clist_if_in:NnT \l_tmpa_clist {finnish}{
6655 \input{hwexam-finnish.ldf}
6656 }
6657 \clist_if_in:NnT \l_tmpa_clist {french}{
6658 \input{hwexam-french.ldf}
6659 }
6660 \clist_if_in:NnT \l_tmpa_clist {russian}{
6661 \input{hwexam-russian.ldf}
6662 }
6663 \makeatother
6664 }{}
6665 }
6666

```

## 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.

```

6667 \newcounter{assignment}
6668 \numberproblemsin{assignment}
6669 \renewcommand\prob@label[1]{\assignment@number.#1}

```

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

```

6670 \keys_define:nn { hwexam / assignment } {
6671 id .str_set:N = \l__hwexam_assign_id_str,
6672 number .int_set:N = \l__hwexam_assign_number_int,
6673 title .tl_set:N = \l__hwexam_assign_title_tl,
6674 type .tl_set:N = \l__hwexam_assign_type_tl,
6675 given .tl_set:N = \l__hwexam_assign_given_tl,
6676 due .tl_set:N = \l__hwexam_assign_due_tl,
6677 loadmodules .code:n = {
6678 \bool_set_true:N \l__hwexam_assign_loadmodules_bool
6679 }
6680 }
6681 \cs_new_protected:Nn \__hwexam_assignment_args:n {
6682 \str_clear:N \l__hwexam_assign_id_str
6683 \int_set:Nn \l__hwexam_assign_number_int {-1}
6684 \tl_clear:N \l__hwexam_assign_title_tl
6685 \tl_clear:N \l__hwexam_assign_type_tl
6686 \tl_clear:N \l__hwexam_assign_given_tl
6687 \tl_clear:N \l__hwexam_assign_due_tl
6688 \bool_set_false:N \l__hwexam_assign_loadmodules_bool

```

```

6689 \keys_set:nn { hwexam / assignment }{ #1 }
6690 }

```

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.

```

6691 \newcommand\given@due[2]{
6692 \bool_lazy_all:nF {
6693 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
6694 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
6695 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
6696 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
6697 }{ #1 }
6698
6699 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
6700 \tl_if_empty:NF \l__hwexam_assign_given_tl {
6701 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
6702 }
6703 }{
6704 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
6705 }
6706
6707 \bool_lazy_or:nnF {
6708 \bool_lazy_and_p:nn {
6709 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
6710 }{
6711 \tl_if_empty_p:V \l__hwexam_assign_due_tl
6712 }
6713 }{
6714 \bool_lazy_and_p:nn {
6715 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
6716 }{
6717 \tl_if_empty_p:V \l__hwexam_assign_due_tl
6718 }
6719 }{ ,~ }
6720
6721 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
6722 \tl_if_empty:NF \l__hwexam_assign_due_tl {
6723 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
6724 }
6725 }{
6726 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
6727 }
6728
6729 \bool_lazy_all:nF {
6730 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
6731 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
6732 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
6733 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
6734 }{ #2 }
6735 }

```

`\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.

```

6736 \newcommand\assignment@title[3]{
6737 \tl_if_empty:NTF \l__hwexam_inclasssign_title_tl {
6738 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
6739 #1
6740 }{
6741 #2\l__hwexam_assign_title_tl#3
6742 }
6743 }{
6744 #2\l__hwexam_inclasssign_title_tl#3
6745 }
6746 }

```

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

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

```

6747 \newcommand\assignment@number{
6748 \int_compare:nNnTF \l__hwexam_inclasssign_number_int = {-1} {
6749 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
6750 \arabic{assignment}
6751 } {
6752 \int_use:N \l__hwexam_assign_number_int
6753 }
6754 }{
6755 \int_use:N \l__hwexam_inclasssign_number_int
6756 }
6757 }

```

(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.

```

6758 \newenvironment{assignment}[1][]{
6759 \__hwexam_assignment_args:n { #1 }
6760 %\sref@target
6761 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
6762 \global\stepcounter{assignment}
6763 }{
6764 \global\setcounter{assignment}{\int_use:N\l__hwexam_assign_number_int}
6765 }
6766 \setcounter{problem}{0}
6767 \def\current@section@level{\document@hwexamtype}
6768 %\sref@label@id{\document@hwexamtype \thesection}
6769 \begin{@assignment}
6770 }{
6771 \end{@assignment}
6772 }

```

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

```

6773 \def\ass@title{
6774 \protect\document@hwexamtype~\arabic{assignment}
6775 \assignment@title{ }\{ \} \; } -- \given@due{ }{ }
6776 }
6777 \ifmultiple
6778 \newenvironment{@assignment}{
6779 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
6780 \begin{omgroup}[loadmodules]{\ass@title}
6781 }{
6782 \begin{omgroup}{\ass@title}
6783 }
6784 }{
6785 \end{omgroup}
6786 }

```

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

```

6787 \else
6788 \newenvironment{@assignment}{
6789 \begin{center}\bf
6790 \Large@title\strut\
6791 \document@hwexamtype~\arabic{assignment}\assignment@title{ }\{ \} \; }{
6792 \large\given@due{-- \; }{ \; }{ }
6793 \end{center}
6794 }{ }
6795 \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.

```

6796 \keys_define:nn { hwexam / inclassignment } {
6797 %id .str_set_x:N = \l__hwexam_assign_id_str,
6798 number .int_set:N = \l__hwexam_inclassign_number_int,
6799 title .tl_set:N = \l__hwexam_inclassign_title_tl,
6800 type .tl_set:N = \l__hwexam_inclassign_type_tl,
6801 given .tl_set:N = \l__hwexam_inclassign_given_tl,
6802 due .tl_set:N = \l__hwexam_inclassign_due_tl,
6803 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
6804 }
6805 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
6806 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
6807 \tl_clear:N \l__hwexam_inclassign_title_tl
6808 \tl_clear:N \l__hwexam_inclassign_type_tl
6809 \tl_clear:N \l__hwexam_inclassign_given_tl
6810 \tl_clear:N \l__hwexam_inclassign_due_tl
6811 \str_clear:N \l__hwexam_inclassign_mhrepos_str
6812 \keys_set:nn { hwexam / inclassignment }{ #1 }
6813 }
6814 \__hwexam_inclassignment_args:n {}
6815
6816 \newcommand\inputassignment[2][ ]{

```



```

6817 \_hwexam_inclassnment_args:n { #1 }
6818 \str_if_empty:NTF \l__hwexam_inclassn_mhrepos_str {
6819 \input{#2}
6820 }{
6821 \stex_in_repository:nn{\l__hwexam_inclassn_mhrepos_str}{
6822 \input{\mhp{path}\l__hwexam_inclassn_mhrepos_str}{#2}}
6823 }
6824 }
6825 \_hwexam_inclassnment_args:n {}
6826 }
6827 \newcommand\includeassignment[2][]{
6828 \newpage
6829 \inputassignment[#1]{#2}
6830 }

```

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

## 42.4 Typesetting Exams

\quizheading

```

6831 \ExplSyntaxOff
6832 \newcommand\quizheading[1]{%
6833 \def\@tas{#1}%
6834 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
6835 \ifx\@tas\@empty\else%
6836 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
6837 \fi%
6838 }
6839 \ExplSyntaxOn

```

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

\testheading

```

6840
6841 \def\hwexamheader{\input{hwexam-default.header}}
6842
6843 \def\hwexamminutes{
6844 \tl_if_empty:NTF \testheading@duration {
6845 {\testheading@min}~\hwexam@minutes@kw
6846 }{
6847 \testheading@duration
6848 }
6849 }
6850
6851 \keys_define:nn { hwexam / testheading } {
6852 min .tl_set:N = \testheading@min,
6853 duration .tl_set:N = \testheading@duration,
6854 reqpts .tl_set:N = \testheading@reqpts,
6855 tools .tl_set:N = \testheading@tools
6856 }
6857 \cs_new_protected:Nn \_hwexam_testheading_args:n {
6858 \tl_clear:N \testheading@min
6859 \tl_clear:N \testheading@duration

```

```

6860 \tl_clear:N \testheading@reqpts
6861 \tl_clear:N \testheading@tools
6862 \keys_set:nn { hwexam / testheading }{ #1 }
6863 }
6864 \newenvironment{testheading}[1][]{
6865   \_hwexam_testheading_args:n{ #1 }
6866   \newcount\check@time\check@time=\testheading@min
6867   \advance\check@time by -\theassignment@totalmin
6868   \newif\if@bonuspoints
6869   \tl_if_empty:NTF \testheading@reqpts {
6870     \@bonuspointsfalse
6871   }{
6872     \newcount\bonus@pts
6873     \bonus@pts=\theassignment@totalpts
6874     \advance\bonus@pts by -\testheading@reqpts
6875     \edef\bonus@pts{\the\bonus@pts}
6876     \@bonuspointstrue
6877   }
6878   \edef\check@time{\the\check@time}
6879
6880   \makeatletter\hwexamheader\makeatother
6881 }{
6882   \newpage
6883 }

```

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

\testspace

```

6884 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

6885 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

6886 \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.

```

6887 <@=problems>
6888 \renewcommand\@problem[3]{
6889   \stepcounter{assignment@probs}
6890   \def\__problemspts{#2}
6891   \ifx\__problemspts\@empty\else
6892     \addtocounter{assignment@totalpts}{#2}
6893   \fi
6894   \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
6895   \xdef\correction@probs{\correction@probs & #1}%
6896   \xdef\correction@pts{\correction@pts & #2}
6897   \xdef\correction@reached{\correction@reached &}

```

```

6898 }
6899 <@@=hwexam>

```

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

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

```

6900 \newcounter{assignment@probs}
6901 \newcounter{assignment@totalpts}
6902 \newcounter{assignment@totalmin}
6903 \def\correction@probs{\correction@probs@kw}
6904 \def\correction@pts{\correction@pts@kw}
6905 \def\correction@reached{\correction@reached@kw}
6906 \stepcounter{assignment@probs}
6907 \newcommand\correction@table{
6908 \resizebox{\textwidth}{!}{%
6909 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
6910 &\multicolumn{\theassignment@probs}{c|}|%|
6911 {\footnotesize\correction@forgrading@kw} &\\ \hline
6912 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
6913 \correction@pts & \theassignment@totalpts & \\ \hline
6914 \correction@reached & & \[.7cm]\hline
6915 \end{tabular}}
6916 </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}

```