

# The sTeX3 Package \*

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

2022-02-17

## 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 2022-02-17)

# Contents

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

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

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

|           |  |            |
|-----------|--|------------|
| <b>25</b> | <b>STeX-Basics Implementation</b>              | <b>72</b>  |
| 25.1      | The STeXDocument Class . . . . .               | 72         |
| 25.2      | Preliminaries . . . . .                        | 72         |
| 25.3      | Messages and logging . . . . .                 | 73         |
| 25.4      | Persistence . . . . .                          | 74         |
| 25.5      | HTML Annotations . . . . .                     | 74         |
| 25.6      | Languages . . . . .                            | 77         |
| 25.7      | Activating/Deactivating Macros . . . . .       | 78         |
| <b>26</b> | <b>STeX-MathHub Implementation</b>             | <b>80</b>  |
| 26.1      | Generic Path Handling . . . . .                | 80         |
| 26.2      | PWD and kpsewhich . . . . .                    | 82         |
| 26.3      | File Hooks and Tracking . . . . .              | 83         |
| 26.4      | MathHub Repositories . . . . .                 | 84         |
| <b>27</b> | <b>STeX-References Implementation</b>          | <b>92</b>  |
| 27.1      | Document URIs and URLs . . . . .               | 92         |
| 27.2      | Setting Reference Targets . . . . .            | 94         |
| 27.3      | Using References . . . . .                     | 95         |
| <b>28</b> | <b>STeX-Modules Implementation</b>             | <b>99</b>  |
| 28.1      | The module environment . . . . .               | 102        |
| 28.2      | Invoking modules . . . . .                     | 108        |
| <b>29</b> | <b>STeX-Module Inheritance Implementation</b>  | <b>110</b> |
| 29.1      | SMS Mode . . . . .                             | 110        |
| 29.2      | Inheritance . . . . .                          | 113        |
| <b>30</b> | <b>STeX-Symbols Implementation</b>             | <b>118</b> |
| 30.1      | Symbol Declarations . . . . .                  | 118        |
| 30.2      | Notations . . . . .                            | 125        |
| <b>31</b> | <b>STeX-Terms Implementation</b>               | <b>136</b> |
| 31.1      | Symbol Invocations . . . . .                   | 136        |
| 31.2      | Terms . . . . .                                | 139        |
| 31.3      | Notation Components . . . . .                  | 146        |
| <b>32</b> | <b>STeX-Structural Features Implementation</b> | <b>149</b> |
| 32.1      | Imports with modification . . . . .            | 149        |
| 32.2      | The feature environment . . . . .              | 156        |
| 32.3      | Features . . . . .                             | 158        |
| <b>33</b> | <b>STeX-Statements Implementation</b>          | <b>163</b> |
| 33.1      | Definitions . . . . .                          | 163        |
| 33.2      | Assertions . . . . .                           | 168        |
| 33.3      | Examples . . . . .                             | 170        |
| 33.4      | Logical Paragraphs . . . . .                   | 173        |

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

**Part I**  
**Manual**

# Chapter 1

## What is sTeX?

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

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

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

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

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

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



# Chapter 2

## Quickstart

### 2.1 Setup

#### 2.1.1 The sTeX IDE

TODO: VSCode Plugin

#### 2.1.2 Manual Setup

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

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

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

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

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

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

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

---

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

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

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

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

Having set everything up, we can write a first  $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$  document. As an example, we will use the `smglom/calculus` and `smglom/arithmetics` archives, which should be present in the designated MathHub-folder.

The document we will consider is the following:

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

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

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

Compiling this document with `pdflatex` should yield the output

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

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

### \usemodule

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

$\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$  now reads this file and makes all semantic macros therein available to use, along with all its dependencies. This enables the usage of `\infinitiesum` later on.

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

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

---

`\symref`  
`\symname`

---

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

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

---

`\importmodule`

---

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

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

## Chapter 3

# Using Semantic Macros

TODO

## Chapter 4

# TeX Archives

### 4.1 The Local MathHub-Directory

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

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

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

### 4.2 The Structure of TeX Archives

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

Each such archive needs two subdirectories:

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

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

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

### 4.3 MANIFEST.MF-Files

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

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

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

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

`source-base` or

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

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

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

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

## Chapter 5

# Creating New Modules and Symbols

TODO

### Example 1

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

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

## 5.1 Advanced Structuring Mechanisms

Given modules:

### Example 2

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

Module 2:  
Module 3:  
Module 4:

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

### Example 3

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

Module 5:

Test:  $a \circ a$

TODO: explain donotclone

### Example 4

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

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

Module 6:

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



## Chapter 6

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

## Chapter 7

# Additional Packages

7.1 Modular Document Structuring

7.2 Slides and Course Notes

7.3 Homework, Problems and Exams

# Chapter 8

## Stuff

### 8.1 Modules

---

`\sTeX`  
`\stex`

---

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

#### 8.1.1 Semantic Macros and Notations

Semantic macros invoke a formally declared symbol.

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

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

##### Example 5

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

$ab$

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

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

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

### Example 6

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

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

.

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

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

### Example 7

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

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

.

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

### Example 8

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

Multiplying again by  $b$  yields...

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

### Example 9

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

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

<sup>4</sup>EdNOTE: TODO

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

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

### Example 10

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

The operator  $\textcolor{teal}{+}$  adds two elements, as in  $a\textcolor{teal}{+}b$ .

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

### Example 11

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

$\textcolor{teal}{\textit{Multiplication}}$  (denoted by  $\cdot$ ) is defined by...

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

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

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

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

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

## Other Argument Types

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

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

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

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

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

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

### Example 12

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

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

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

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

### Example 13

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

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

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

5 6

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

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

## Precedences

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

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

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

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

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

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

For example:

**Module 9:**

### Example 14

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

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

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

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

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

**sms** ( $\langle boolean \rangle$ ) use *persisted* mode (see ???).

**image** ( $\langle boolean \rangle$ ) passed on to tikzinput.

### 9.1 Macros and Environments

---

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

---

---

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

---

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

---

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

---

---

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

---

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

---

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

---

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

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

---

|  |
|--|
| <code>\c_stex_languages_prop</code>        |
| <code>\c_stex_language_abbrevs_prop</code> |

---

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

---

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

---

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

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

---

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

---

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

# Chapter 10

## sTEX-MathHub

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

### 10.1 Macros and Environments

---

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

#### 10.1.1 Files, Paths, URIs

---

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

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

---

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

---

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

---

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

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

---

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

`\g_stex_currentfile_seq`

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

**Test 1**

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

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

**10.1.2 MathHub Archives**

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

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

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

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

`\l_stex_current_repository_prop`

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

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

## Test 2

```

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

```

```

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

```

## Chapter 11

# sTeX-References

Code related to links and cross-references

### 11.1 Macros and Environments

# Chapter 12

## sTeX-Modules

Code related to Modules

### 12.1 Macros and Environments

---

---

`\l_stex_current_module_str`

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

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

Additionally, it stores:

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

---

---

`\l_stex_all_modules_seq`

Stores full URIs for all modules currently in scope.



---

```
\g_stex_module_files_prop
\g_stex_modules_in_file_seq
```

---

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

---

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

---



---

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

---

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

---

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

---

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

---

```
\stex_add_constant_to_current_module:n
```

---

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

---

```
\stex_add_import_to_current_module:n
```

---

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

---

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

---

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

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

---

```
\stex_modules_current_namespace:
```

---

Computes the current namespace

### Test 3

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

```

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

```

### 12.1.1 The module-environment

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

---

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

---

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

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

### Test 4

```

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

```

```

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

```

## Test 5

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

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

---

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

---

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

---

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

---

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

## Test 6

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

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

---

**`\stex_activate_module:n`**

---

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

## Chapter 13

# STEX-Module Inheritance

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

### 13.1 Macros and Environments

#### 13.1.1 SMS Mode

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

---

`\g_stex_smsmode_allowedmacros_tl`

---

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

---

`\g_stex_smsmode_allowedmacros_escape_tl`

---

Macros that are executed with the category codes restored.

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

---

`\g_stex_smsmode_allowedenvs_seq`

---

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

---

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

---

Tests whether SMS mode is currently active.

---

`\stex_smsmode_set_codes:`

---

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

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

---

**\stex\_in\_smsmode:nn**

---

**\stex\_in\_smsmode:nn** {<name>} {<code>}

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

### Test 7

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

## 13.1.2 Imports and Inheritance

---

**\importmodule**

---

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

Imports a module by reading it from a file and “activating” it.  $\TeX$  determines the module and its containing file by passing its arguments on to **\stex\_import\_module\_path:nn**.

### Test 8

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

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

---

**\usemodule**

---

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

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

## Test 9

```

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

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

```

Module 18:

Module 19: Meaning: »macro:->\stex\_invoke\_symbol:n {file://stextest?UseTest1?foo}<

Module 20: Meaning: »undefined<

Meaning: »macro:->\stex\_invoke\_symbol:n {file://stextest?UseTest2?bar}<

All modules: <http://mathhub.info/sTeX?Metatheory, file://stextest?UseTest3, file://stextest?UseTest2>

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

## Test 10

```

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

```

Circular dependencies:

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

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

---

---

`\stex_import_module_uri:nn`

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

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

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

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

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

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

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

2. Otherwise:

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

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

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

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

---

---

`\stex_import_require_module:nnnn`

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

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

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



# Chapter 14

## TeX-Symbols

Code related to symbol declarations and notations

### 14.1 Macros and Environments

---

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

---

---

`\stex_symdecl_do:n`

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

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

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

### Test 11

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

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

---

---

`\l_stex_all_symbols_seq`

Stores full URIs for all modules currently in scope.

---

---

`\stex_get_symbol:n`

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

---

---

`\notation`

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

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

---

---

`\stex_notation_do:nn`

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

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

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

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

### Test 12

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

Module 23:

---

---

`\symdef`

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

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

### Test 13

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

Module 24:  $a+b+c$

# Chapter 15

## STEX-Terms

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

### 15.1 Macros and Environments

|  |   |
|--|---|
| <hr/> <hr/> <code>\STEXsymbol</code>   | Uses <code>\stex_get_symbol:n</code> to find the symbol denoted by the first argument and passes the result on to <code>\stex_invoke_symbol:n</code>  |
| <hr/> <hr/> <code>\symref</code>   | <code>\symref{&lt;symbol&gt;}{&lt;text&gt;}</code><br>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.<br>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><br><code>\_stex_term_math_oma:nnnn</code><br><code>\_stex_term_math_omb:nnnn</code> | <code>&lt;URI&gt;&lt;fragment&gt;&lt;precedence&gt;&lt;body&gt;</code><br>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><br>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><br>Annotates <code>&lt;body&gt;</code> as the <code>&lt;int&gt;</code> th (associative) <i>sequence</i> argument (as comma-separated list of terms) of the current OMA or OMBIND, with (downwards) argument precedence <code>&lt;prec&gt;</code> and associative notation <code>&lt;notation&gt;</code> .  |

---

`\infprec`  
`\neginfprec`

---

Maximal and minimal notation precedences.

---

`\dobrackets`

---

`\dobrackets {⟨body⟩}`

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

---

`\withbrackets`

---

`\withbrackets ⟨left⟩ ⟨right⟩ {⟨body⟩}`

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

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

### Test 14

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

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

### Test 15

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

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

Module 26:  $\langle a|_{[b;c;d;e;f]}^g \rangle$  and  $\langle a|_{[b;c]}^g \rangle$  and  $\langle a|_{[b]^c} \rangle$

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

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

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

---

`\stex_term_custom:nn`

---

`\stex_term_custom:nn{⟨URI⟩}{⟨args⟩}`

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

## Test 16

```
\begin{smodule}{TextTest}
\importmodule{Foo}

\bar[some ]a[ and some ]b[ and also some ]c[ here].

$\bar*[\text{some }]a[\text{ and some }]b[\text{ and also some }]c[\text{ here}]\$.

$\bar!![\mathtt{bar}]\$

\bar*{a}*{b}[or just some ]c

\bar![bar]

\bar[or first ]*[2]{b}[ , then ]*[3]{c}[ , and finally ]a

\end{smodule}
```

```
Module 27:  some aand some band also some chere.
           some a and some b and also some c here.
           bar
           or just some c
           bar
           or first b, then c, and finally a
```

---

`\stex_highlight_term:nn`

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

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

---

`\comp`

`\comp{<args>}`

`\compemph`

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

`\compemph@uri`

`\defemph`

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

`\defemph@uri`

`\symrefemph`

`\symrefemph@uri`

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

---

`\STEXinvisible`

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

---

`\ellipses`

TODO

## Chapter 16

# ST<sub>E</sub>X-Structural Features

Code related to structural features

### 16.1 Macros and Environments

#### 16.1.1 Structures

`mathstructure` TODO

## Chapter 17

# sTeX-Statements

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

### 17.1 Macros and Environments

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



## Chapter 18

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

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

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

## Contents

## 18.1 Introduction

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

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

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

Example 1: A very explicit proof, marked up semantically

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

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

## 18.2 The User Interface

### 18.2.1 Package Options

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

### 18.2.2 Proofs and Proof steps

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

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

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

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

### 18.2.3 Justifications

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

#### 18.2.4 Proof Structure

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

### 18.2.5 Proof End Markers

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

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

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

### 18.2.6 Configuration of the Presentation

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

| Environment | configuration macro | value       |
|-------------|---------------------|-------------|
| sproof      | \spf@proof@kw       | Proof       |
| sketchproof | \spf@sketchproof@kw | ProofSketch |

Figure 1: Configuration Hooks for Semantic Proof Markup

`\pstlabelstyle`

`\pstlabelstyle{style}` sets the style; see Figure 2 for an overview of styles. Package writers can add additional styles by adding a macro `\pstmake@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 `\LaTeX \@for...:=...\do{...}` macro; see Figure 2 for examples.

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

Figure 2: Configuration Proof Step Label Styles

### 18.3 Limitations

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

---

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

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

## Chapter 19

# sTeX-Metatheory

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

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

### 19.1 Symbols



**Part III**  
**Extensions**

## Chapter 20

# Tikzinput

### 20.1 Macros and Environments

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

## Chapter 21

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

The `document-structure` package is part of the  $\S\TeX$  collection, a version of  $\TeX$ / $\LaTeX$  that allows to markup  $\TeX$ / $\LaTeX$  documents semantically without leaving the document format, essentially turning  $\TeX$ / $\LaTeX$  into a document format for mathematical knowledge management (MKM).

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

### 21.1 Introduction

$\S\TeX$  is a version of  $\TeX$ / $\LaTeX$  that allows to markup  $\TeX$ / $\LaTeX$  documents semantically without leaving the document format, essentially turning  $\TeX$ / $\LaTeX$  into a document format for mathematical knowledge management (MKM). The package supports direct translation to the OMDOC format [Koh06]

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

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

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

## 21.2 The User Interface

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

### 21.2.1 Package and Class Options

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

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

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

### 21.2.2 Document Structure

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

```

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

```

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

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

`blindomgroup`

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

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

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

Example 3: A typical Document Structure of a Book

`\skipomgroup`

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

`\currentsectionlevel`  
`\CurrentSectionLevel`

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

---

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

### 21.2.3 Ignoring Inputs

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

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

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

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

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

### 21.2.4 Structure Sharing

`\STRlabel` The `\STRlabel` macro takes two arguments: a label and the content and stores the content for later use by `\STRcopy[\langle URL \rangle]{\langle label \rangle}`, which expands to the previously stored content. If the `\STRlabel` macro was in a different file, then we can give a URL `\langle URL \rangle` that lets L<sup>A</sup>T<sub>E</sub>XML generate the correct reference.

`\STRcopy`

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

### 21.2.5 Global Variables

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

`\setSGvar`

`\useSGvar`

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

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

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

### 21.2.6 Colors

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

## 21.3 Limitations

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

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

## Chapter 22

# NotesSlides – Slides and Course Notes

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

### 22.1 Introduction

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

### 22.2 The User Interface

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

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

#### 22.2.1 Package Options

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

- |   |  |
|---|--|
| <code>slides</code><br><code>notes</code> | <ul style="list-style-type: none"><li>• The options <code>slides</code> and <code>notes</code> switch between slides mode and notes mode (see Section 22.2.2).</li></ul> |
|---|--|



|  |   |
|--|---|
| <code>sectocframes</code>                        | <ul style="list-style-type: none"> <li>If the option <code>sectocframes</code> is given, then for the <code>omgroups</code>, special frames with the <code>omgroup</code> title (and number) are generated.</li> </ul>  |
| <code>showmeta</code>                            | <ul style="list-style-type: none"> <li><code>showmeta</code>. If this is set, then the metadata keys are shown (see [Koh20b] for details and customization options).</li> </ul>   |
| <code>frameimages</code><br><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.

```
\title{320101 General Computer Science (Fall 2010)}
\begin{testheading}[duration=one hour,min=60,reqpts=27]
  Good luck to all students!
\end{testheading}
```

Example 8: A generated test heading.

Part IV

# Implementation

## Chapter 25

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

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

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

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

### 25.2 Preliminaries

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

Package options:

```

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

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

```

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

## 25.3 Messages and logging

```

42 <@@=stex_log>

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

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

```

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

Redirecting messages:

```

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

```

## 25.4 Persistence

77 `<@@=stex_persist>`

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

```

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

```

(End definition for `\c__stex_persist_sms_iow`.)

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

```

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

```

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

## 25.5 HTML Annotations

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

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

```

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

```

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

```

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

```



```

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

```

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

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

```

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

```

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

`\_stex_annotate_checkempty:n`

```

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

```

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

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

```

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

```

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

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

```

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

```

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

`\stex_annotate:env`

`\stex_annotate_invisible:n`

`\stex_annotate_invisible:nnn`

We define four macros for introducing attributes in the HTML output. The definitions depend on the “backend” used (L<sup>A</sup>T<sub>E</sub>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>).

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.

```

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

```

```

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

```

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

## 25.6 Languages

```

233 <@=stex_language>

```

`\c_stex_languages_prop`  
`\c_stex_language_abbrevs_prop`

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

```

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

```

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

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

```

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

```

## 25.7 Activating/Deactivating Macros

`\stex_deactivate_macro:Nn`

```

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

```

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

`\stex_reactivate_macro:N`

```

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

```

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

`\stex_do_aftergroup:nn`

```

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

```

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

```

300
301 \protected\def\ignorespacesandpars{
302   \begingroup\catcode13=10\relax
303   \@ifnextchar\par{
304     \endgroup\expandafter\ignorespacesandpars\@gobble
305   }{
306     \endgroup
307   }
308 }
309
310
311 </package>

```

## Chapter 26

# STEX -MathHub Implementation

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

### 26.1 Generic Path Handling

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

```

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

```

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

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

```

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

```

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

## 26.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

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

```

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

We determine the PWD

`\c_stex_pwd_seq`  
`\c_stex_pwd_str`

```

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

```



```

417 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
418 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
419 \stex_debug:nn {mathhub} {PWD:~\str_use:N\c_stex_pwd_str}

```

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

## 26.3 File Hooks and Tracking

```

420 <@@=stex_files>

```

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

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

`\g_stex_files_stack` keeps track of file changes

```

421 \seq_gclear_new:N\g_stex_files_stack

```

(End definition for `\g_stex_files_stack`.)

`\c_stex_mainfile_seq`  
`\c_stex_mainfile_str`

```

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

```

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

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

```

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

```

```

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

```

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

## 26.4 MathHub Repositories

```

454 <@@=stex_mathhub>

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

```

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

```

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

```

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

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

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

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

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

```

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

```

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

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

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

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

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

```

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

```

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

`\stex_set_current_repository:n`

```

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

```

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

`\stex_require_repository:n`

```

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

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

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

```

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

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

```

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

```

```

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

```

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

```

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

```

```

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

```

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

### `\mhpath`

```

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

```

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

### `\libinput`

```

702 \cs_new_protected:Npn \libinput #1 {
703   \prop_if_exist:NF \l_stex_current_repository_prop {
704     \msg_error:nnn{stex}{error/notinarchive}\libinput
705   }
706   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
707     \msg_error:nnn{stex}{error/notinarchive}\libinput
708   }
709   \tl_clear:N \l__stex_mathhub_libinput_files_seq
710   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
711   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
712
713   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
714     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}

```

```

715 \IfFileExists{ \l_tmpa_str }{
716   \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
717 }{}
718 \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
719 \seq_put_right:No \l_tmpa_seq \l_tmpa_str
720 }
721
722 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
723 \IfFileExists{ \l_tmpa_str }{
724   \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
725 }{}
726
727 \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
728   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
729 }{
730   \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
731     \input{ ##1 }
732   }
733 }
734 }

```

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

#### `\libusepackage`

```

735 \NewDocumentCommand \libusepackage {0{ } m} {
736   \prop_if_exist:NF \l_stex_current_repository_prop {
737     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
738   }
739   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
740     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
741   }
742   \tl_clear:N \l__stex_mathhub_libinput_files_seq
743   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
744   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
745
746   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
747     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2.sty}
748     \IfFileExists{ \l_tmpa_str }{
749       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
750     }{}
751     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
752     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
753   }
754
755   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2.sty}
756   \IfFileExists{ \l_tmpa_str }{
757     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
758   }{}
759
760   \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
761     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
762   }{
763     \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
764       \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {

```



```

765         \usepackage[#1]{ ##1 }
766     }
767 }{
768     \msg_error:nxxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
769 }
770 }
771 }

```

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

```

772
773 \AddToHook{begindocument}{
774 \ltx@ifpackageloaded{graphicx}{
775     \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
776     \newcommand\mhgraphics[2][]{\%
777         \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
778         \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}}
779     \newcommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
780 }{}
781 \ltx@ifpackageloaded{listings}{
782     \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
783     \newcommand\lstinputmhlisting[2][]{\%
784         \def\lst@mhrepos{}\setkeys{lst}{#1}%
785         \lstinputlisting[#1]{\mhp\lst@mhrepos{#2}}}
786     \newcommand\clstinputmhlisting[2][]{\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
787 }{}
788 }
789
790
791 \</package>

```

## Chapter 27

# STEX -References Implementation

```
792 <*package>
793
794 %%%%%%%%%% references.dtx %%%%%%%%%%
795
796 %\RequirePackage{hyperref}
797 %\RequirePackage{cleveref}
798 <@@=stex_refs>
799
800 \iow_new:N \c__stex_refs_refs_iow
801 \AddToHook{begindocument}{
802   \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
803 }
804 \AddToHook{enddocument}{
805   \iow_close:N \c__stex_refs_refs_iow
806 }
807
808 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
809
810 \NewDocumentCommand \STEXreftitle { m } {
811   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
812 }
```

### 27.1 Document URIs and URLs

```
813
814 \str_new:N \l_stex_current_docns_str
815
816 \cs_new_protected:Nn \stex_get_document_uri: {
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 { narr } \l_tmpa_str {
826     \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
827   }
828 }
829
830 \str_if_empty:NTF \l_tmpa_str {
831   \str_set:Nx \l_stex_current_docns_str {
832     file:/\stex_path_to_string:N \l_tmpa_seq
833   }
834 }{
835   \bool_set_true:N \l_tmpa_bool
836   \bool_while_do:Nn \l_tmpa_bool {
837     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
838     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
839       {source} { \bool_set_false:N \l_tmpa_bool }
840     }{}{
841       \seq_if_empty:NT \l_tmpa_seq {
842         \bool_set_false:N \l_tmpa_bool
843       }
844     }
845   }
846
847   \seq_if_empty:NTF \l_tmpa_seq {
848     \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
849   }{
850     \str_set:Nx \l_stex_current_docns_str {
851       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
852     }
853   }
854 }
855 }
856
857 \str_new:N \l_stex_current_docurl_str
858 \cs_new_protected:Nn \stex_get_document_url: {
859   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
860   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
861   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
862   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
863   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
864
865   \str_clear:N \l_tmpa_str
866   \prop_if_exist:NT \l_stex_current_repository_prop {
867     \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
868       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
869         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
870       }
871     }
872
873   \str_if_empty:NTF \l_tmpa_str {
874     \str_set:Nx \l_stex_current_docurl_str {
875       file:/\stex_path_to_string:N \l_tmpa_seq

```

```

876     }
877   }{
878     \bool_set_true:N \l_tmpa_bool
879     \bool_while_do:Nn \l_tmpa_bool {
880       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
881       \exp_args:No \str_case:nnTF { \l_tmpb_str } {
882         {source} { \bool_set_false:N \l_tmpa_bool }
883       }{}{
884         \seq_if_empty:NT \l_tmpa_seq {
885           \bool_set_false:N \l_tmpa_bool
886         }
887       }
888     }
889
890     \seq_if_empty:NTF \l_tmpa_seq {
891       \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
892     }{
893       \str_set:Nx \l_stex_current_docurl_str {
894         \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
895       }
896     }
897   }
898 }

```

## 27.2 Setting Reference Targets

```

899 \str_const:Nn \c__stex_refs_url_str{URL}
900 \str_const:Nn \c__stex_refs_ref_str{REF}
901 \str_new:N \l__stex_refs_curr_label_str
902 % @currentlabel -> number
903 % @currentlabelname -> title
904 % @currentHref -> name.number <- id of some kind
905 % \theH# -> \arabic{section}
906 % \the# -> number
907 % \hyper@makecurrent{#}
908 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
909   \str_clear:N \l__stex_refs_curr_label_str
910   \str_set:Nx \l_tmpa_str { #1 }
911   \str_if_empty:NF \l_tmpa_str {
912     \stex_get_document_uri:
913     \str_set:Nx \l__stex_refs_curr_label_str {
914       \l_stex_current_docns_str?#1
915     }
916     \seq_if_exist:cF{g__stex_refs_labels_#1_seq}{
917       \seq_new:c {g__stex_refs_labels_#1_seq}
918     }
919     \seq_if_in:coF{g__stex_refs_labels_#1_seq}\l__stex_refs_curr_label_str {
920       \seq_gput_right:co{g__stex_refs_labels_#1_seq}\l__stex_refs_curr_label_str
921     }
922     \stex_if_smsmode:TF {
923       \stex_get_document_url:
924       \str_gset_eq:cN {sref_url_}\l__stex_refs_curr_label_str _str}\l_stex_current_docurl_str
925       \str_gset_eq:cN {sref_}\l__stex_refs_curr_label_str _type}\c__stex_refs_url_str
926     }{

```

```

927     \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~~~\expandafter\unexpanded\expandafter
928     \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
929     \immediate\write\@auxout{\stexauxadddocref{\l_stex_current_docns_str}{#1}}
930     \str_gset:cx {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_ref_str
931   }
932 }
933 }
934
935 \cs_new_protected:Npn \stexauxadddocref #1 #2 {
936   \str_set:Nn \l_tmpa_str {#1?#2}
937   \str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
938   \seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
939     \seq_new:c {g__stex_refs_labels_#2_seq}
940   }
941   \seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
942     \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
943   }
944 }
945
946 \AtEndDocument{
947   \def\stexauxadddocref#1 #2 {}{}}
948 }
949
950 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
951   \stex_if_smsmode:TF {
952     \str_if_exist:cF{sref_sym_#1_type}{
953       \stex_get_document_url:
954       \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str
955       \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
956     }
957   }{
958     \str_if_empty:NF \l__stex_refs_curr_label_str {
959       \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
960       \immediate\write\@auxout{
961         \exp_not:N\expandafter\def\exp_not:N\csname sref_sym_#1_label_str\exp_not:N\endcsname
962         \l__stex_refs_curr_label_str
963       }
964     }
965   }
966 }
967 }

```

## 27.3 Using References

```

968 \str_new:N \l__stex_refs_indocument_str
969 \keys_define:nn { stex / sref } {
970   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
971   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
972   pre           .tl_set:N = \l__stex_refs_pre_tl ,
973   post          .tl_set:N = \l__stex_refs_post_tl ,
974   %indoc        .str_set_x:N = \l__stex_refs_repo_str ,
975 }
976
977

```

```

978
979 \cs_new_protected:Nn \__stex_refs_args:n {
980   \tl_clear:N \l__stex_refs_linktext_tl
981   \tl_clear:N \l__stex_refs_fallback_tl
982   \tl_clear:N \l__stex_refs_pre_tl
983   \tl_clear:N \l__stex_refs_post_tl
984   \str_clear:N \l__stex_refs_repo_str
985   \keys_set:nn { stex / sref } { #1 }
986 }
987
988 \NewDocumentCommand \sref { 0{} m}{
989   \__stex_refs_args:n { #1 }
990   \str_if_empty:NTF \l__stex_refs_indocument_str {
991     \str_set:Nx \l_tmpa_str { #2 }
992     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
993     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
994       \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
995         \seq_get_left:cnF {g__stex_refs_labels_\l_tmpa_str _seq} \l_tmpa_str {
996           \str_clear:N \l_tmpa_str
997         }
998       }{
999         \str_clear:N \l_tmpa_str
1000       }
1001     }{
1002       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1003       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1004       \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
1005       \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
1006         \str_set_eq:NN \l_tmpc_str \l_tmpa_str
1007         \str_clear:N \l_tmpa_str
1008         \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str _seq} {
1009           \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
1010             \str_range:nnn { ##1 }{ -\l_tmpa_int}{ -1 }
1011           }{
1012             \seq_map_break:n {
1013               \str_set:Nn \l_tmpa_str { ##1 }
1014             }
1015           }
1016         }
1017       }{
1018         \str_clear:N \l_tmpa_str
1019       }
1020     }
1021   \str_if_empty:NTF \l_tmpa_str {
1022     \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_lin
1023   }{
1024     \str_if_eq:cNTF {sref_\l_tmpa_str _type} \c__stex_refs_ref_str {
1025       \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1026         \cs_if_exist:cTF{autoref}{
1027           \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1028         }{
1029           \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1030         }
1031       }{

```

```

1032         \ltx@ifpackageloaded{hyperref}{
1033             \hyperref[sref_\l_tmpa_str]{\l__stex_refs_linktext_tl
1034         }{
1035             \l__stex_refs_linktext_tl
1036         }
1037     }
1038 }{
1039     \ltx@ifpackageloaded{hyperref}{
1040         \href{\use:c{sref_url_\l_tmpa_str _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl \
1041     }{
1042         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs
1043     }
1044 }
1045 }
1046 }{
1047     % TODO
1048 }
1049 }
1050
1051 \NewDocumentCommand \srefsym { 0{} m}{
1052     \stex_get_symbol:n { #2 }
1053     \__stex_refs_sym_aux:nn{#1}{\l_stex_get_symbol_uri_str}
1054 }
1055
1056 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1057     \str_if_exist:cTF {sref_sym_#2 _label_str }{
1058         \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1059     }{
1060         \__stex_refs_args:n { #1 }
1061         \str_if_empty:NTF \l__stex_refs_indocument_str {
1062             \tl_if_exist:cTF{sref_sym_#2 _type}{
1063                 % doc uri in \l_tmpb_str
1064                 \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1065                 \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1066                     % reference
1067                     \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1068                         \cs_if_exist:cTF{autoref}{
1069                             \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1070                         }{
1071                             \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1072                         }
1073                     }{
1074                         \ltx@ifpackageloaded{hyperref}{
1075                             \hyperref[sref_sym_#2]{\l__stex_refs_linktext_tl
1076                         }{
1077                             \l__stex_refs_linktext_tl
1078                         }
1079                     }
1080                 }{
1081                     % URL
1082                     \ltx@ifpackageloaded{hyperref}{
1083                         \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl \
1084                     }{
1085                         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_re

```

```

1086         }
1087     }
1088     }{
1089         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_l
1090     }
1091     }{
1092         % TODO
1093     }
1094 }
1095 }
1096
1097 \cs_new_protected:Npn \srefsymuri #1 #2 {
1098     \__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1099 }
1100
1101 \end{package}

```



## Chapter 28

# STEX -Modules Implementation

```
1102 <*package>
1103
1104 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1105
1106 <@@=stex_modules>
1107
1108     Warnings and error messages
1109 \msg_new:nnn{stex}{error/unknownmodule}{
1110     No~module~#1~found
1111 }
1112 \msg_new:nnn{stex}{error/syntax}{
1113     Syntax~error:~#1
1114 }
1115 \msg_new:nnn{stex}{error/siglanguage}{
1116     Module~#1~declares~signature~#2,~but~does~not~
1117     declare~its~language
1118 }
1119 \msg_new:nnn{stex}{warning/deprecated}{
1120     #1~is~deprecated;~please~use~#2~instead!
1121 }
1122 \msg_new:nnn{stex}{error/conflictingmodules}{
1123     Conflicting~imports~for~module~#1
1124 }
1125
1126 \l_stex_current_module_str The current module:
1127 \str_new:N \l_stex_current_module_str
1128
1129 (End definition for \l_stex_current_module_str. This variable is documented on page 26.)
1130
1131 \l_stex_all_modules_seq Stores all available modules
1132 \seq_new:N \l_stex_all_modules_seq
1133
1134 (End definition for \l_stex_all_modules_seq. This variable is documented on page 26.)
```

```

\stex_if_in_module_p:
\stex_if_in_module:TF
1126 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1127   \str_if_empty:NTF \l_stex_current_module_str
1128   \prg_return_false: \prg_return_true:
1129 }

```

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

```

\stex_if_module_exists_p:n
\stex_if_module_exists:nTF
1130 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1131   \prop_if_exist:cTF { c_stex_module_#1_prop }
1132   \prg_return_true: \prg_return_false:
1133 }

```

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

`\stex_add_to_current_module:n` Only allowed within modules:

```

\STEXexport
1134 \cs_new_protected:Nn \stex_add_to_current_module:n {
1135   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1136 }
1137 \cs_new_protected:Npn \STEXexport {
1138   \beginngroup
1139   \newlinechar=-1\relax
1140   \endlinechar=-1\relax
1141   %\catcode'\ = 9\relax
1142   \expandafter\endgroup\STEXexport:n
1143 }
1144 \cs_new_protected:Nn \STEXexport:n {
1145   \ignorespaces #1
1146   \stex_add_to_current_module:n { \ignorespaces #1 }
1147   \stex_smsmode_do:
1148 }
1149 \stex_deactivate_macro:Nn \STEXexport {module~environments}

```

(End definition for `\stex_add_to_current_module:n` and `\STEXexport`. These functions are documented on page 27.)

```

\stex_add_constant_to_current_module:n
1150 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1151   \str_set:Nx \l_tmpa_str { #1 }
1152   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1153 }
1154
1155 %\cs_new_protected:Nn \stex_add_field_to_current_module:n {
1156 % \str_set:Nx \l_tmpa_str { #1 }
1157 % \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _fields} { \l_tmpa_str }
1158 %}

```

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

```

\stex_collect_imports:n
1159 \cs_new_protected:Nn \stex_collect_imports:n {
1160   \seq_clear:N \l_stex_collect_imports_seq
1161   \__stex_modules_collect_imports:n {#1}

```

```

1162 }
1163 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1164   \seq_map_inline:cn {c_stex_module_#1_imports} {
1165     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1166       \__stex_modules_collect_imports:n { ##1 }
1167     }
1168   }
1169   \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1170     \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1171   }
1172 }

```

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

`\stex_add_import_to_current_module:n`

```

1173 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1174   \str_set:Nx \l_tmpa_str { #1 }
1175   \exp_args:Nno
1176   \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1177     \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1178   }
1179 }

```

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

`\stex_modules_compute_namespace:nN`

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

```

1180 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
1181   \str_set:Nx \l_tmpa_str { #1 }
1182   \seq_set_eq:NN \l_tmpa_seq #2
1183   % split off file extension
1184   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1185   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1186   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1187   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1188
1189   \bool_set_true:N \l_tmpa_bool
1190   \bool_while_do:Nn \l_tmpa_bool {
1191     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1192     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1193       {source} { \bool_set_false:N \l_tmpa_bool }
1194     }{}{
1195       \seq_if_empty:NT \l_tmpa_seq {
1196         \bool_set_false:N \l_tmpa_bool
1197       }
1198     }
1199   }
1200
1201   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
1202   \str_if_empty:NTF \l_stex_modules_subpath_str {
1203     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
1204   }{
1205     \str_set:Nx \l_stex_modules_ns_str {
1206       \l_tmpa_str/\l_stex_modules_subpath_str

```

```

1207     }
1208   }
1209 }

```

(End definition for `\stex_modules_compute_namespace:nN`. This function is documented on page 27.)  
Stores its return values in:

```

\l_stex_modules_ns_str
\l_stex_modules_subpath_str
1210 \str_new:N \l_stex_modules_ns_str
1211 \str_new:N \l_stex_modules_subpath_str

```

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

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

```

1212 \cs_new_protected:Nn \stex_modules_current_namespace: {
1213   \str_clear:N \l_stex_modules_subpath_str
1214   \prop_if_exist:NTF \l_stex_current_repository_prop {
1215     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1216     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1217   }{
1218     % split off file extension
1219     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1220     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1221     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1222     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1223     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1224     \str_set:Nx \l_stex_modules_ns_str {
1225       file:/\stex_path_to_string:N \l_tmpa_seq
1226     }
1227   }
1228 }

```

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

## 28.1 The module environment

module arguments:

```

1229 \keys_define:nn { stex / module } {
1230   title      .tl_set:N      = \smodulename ,
1231   type       .str_set_x:N   = \smodulename ,
1232   id         .str_set_x:N   = \smodulename ,
1233   deprecate  .str_set_x:N   = \l_stex_module_deprecate_str ,
1234   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1235   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1236   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1237   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1238   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1239   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1240   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1241 }
1242
1243 \cs_new_protected:Nn \__stex_modules_args:n {

```

```

1244 \str_clear:N \smoduletitle
1245 \str_clear:N \smoduletype
1246 \str_clear:N \smoduleid
1247 \str_clear:N \l_stex_module_ns_str
1248 \str_clear:N \l_stex_module_deprecate_str
1249 \str_clear:N \l_stex_module_lang_str
1250 \str_clear:N \l_stex_module_sig_str
1251 \str_clear:N \l_stex_module_creators_str
1252 \str_clear:N \l_stex_module_contributors_str
1253 \str_clear:N \l_stex_module_meta_str
1254 \str_clear:N \l_stex_module_srccite_str
1255 \keys_set:nn { stex / module } { #1 }
1256 }
1257
1258 % module parameters here? In the body?
1259

```

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

```

1260 \cs_new_protected:Nn \stex_module_setup:nn {
1261   \str_set:Nx \l_stex_module_name_str { #2 }
1262   \__stex_modules_args:n { #1 }
1263
1264   First, we set up the name and namespace of the module.
1265   Are we in a nested module?
1266
1267   \stex_if_in_module:TF {
1268     % Nested module
1269     \prop_get:cnN {c_stex_module\l_stex_current_module_str _prop}
1270       { ns } \l_stex_module_ns_str
1271     \str_set:Nx \l_stex_module_name_str {
1272       \prop_item:cn {c_stex_module\l_stex_current_module_str _prop}
1273         { name } / \l_stex_module_name_str
1274     }
1275   }{
1276     % not nested:
1277     \str_if_empty:NT \l_stex_module_ns_str {
1278       \stex_modules_current_namespace:
1279       \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1280       \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1281         / {\l_stex_module_ns_str}
1282       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1283       \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1284         \str_set:Nx \l_stex_module_ns_str {
1285           \stex_path_to_string:N \l_tmpa_seq
1286         }
1287       }
1288     }
1289   }
1290 }

```

Next, we determine the language of the module:

```

1286 \str_if_empty:NT \l_stex_module_lang_str {
1287   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1288   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1289   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1290   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>

```

```

1291 \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1292 \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1293   inferred~from~file~name}
1294 \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1295 }
1296 }
1297
1298 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
1299 \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1300 \l_tmpa_str {
1301 \ltx@ifpackageloaded{babel}{
1302 \exp_args:Nx \selectlanguage { \l_tmpa_str }
1303 }{}
1304 } {
1305 \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
1306 }
1307 }}

```

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

```

1308 \str_if_empty:NTF \l_stex_module_sig_str {
1309 \exp_args:Nnx \prop_gset_from_keyval:cn {
1310 c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1311 } {
1312 name = \l_stex_module_name_str ,
1313 ns = \l_stex_module_ns_str ,
1314 file = \exp_not:o { \g_stex_currentfile_seq } ,
1315 lang = \l_stex_module_lang_str ,
1316 sig = \l_stex_module_sig_str ,
1317 deprecate = \l_stex_module_deprecate_str ,
1318 meta = \l_stex_module_meta_str
1319 }
1320 \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1321 \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _fields}
1322 \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1323 \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1324 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1325 \str_if_empty:NT \l_stex_module_meta_str {
1326 \str_set:Nx \l_stex_module_meta_str {
1327 \c_stex_metatheory_ns_str ? Metatheory
1328 }
1329 }
1330 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1331 \bool_set_true:N \l_stex_in_meta_bool
1332 \exp_args:Nx \stex_add_to_current_module:n {
1333 \bool_set_true:N \l_stex_in_meta_bool
1334 \stex_activate_module:n {\l_stex_module_meta_str}
1335 \bool_set_false:N \l_stex_in_meta_bool
1336 }
1337 \stex_activate_module:n {\l_stex_module_meta_str}
1338 \bool_set_false:N \l_stex_in_meta_bool
1339 }

```

```

1340 }{
1341   \str_if_empty:NT \l_stex_module_lang_str {
1342     \msg_error:nnxx{stex}{error/siglanguage}{
1343       \l_stex_module_ns_str?\l_stex_module_name_str
1344     }\l_stex_module_sig_str}
1345   }
1346
1347   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1348   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1349   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1350   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1351   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1352   \str_set:Nx \l_tmpa_str {
1353     \stex_path_to_string:N \l_tmpa_seq /
1354     \l_tmpa_str . \l_stex_module_sig_str .tex
1355   }
1356   \IfFileExists \l_tmpa_str {
1357     \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1358       \str_clear:N \l_stex_current_module_str
1359       \seq_clear:N \l_stex_all_modules_seq
1360       \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1361     }
1362   }{
1363     \msg_error:nnxx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1364   }
1365   \stex_if_smsmode:F {
1366     \stex_activate_module:n {
1367       \l_stex_module_ns_str ? \l_stex_module_name_str
1368     }
1369   }
1370   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1371 }
1372 \str_if_empty:NF \l_stex_module_deprecate_str {
1373   \msg_warning:nnxx{stex}{warning/deprecated}{
1374     Module~\l_stex_current_module_str
1375   }{
1376     \l_stex_module_deprecate_str
1377   }
1378 }
1379 }

```

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

**module** The module environment.

```

\__stex_modules_begin_module: implements \begin{smodule}

1380 \int_new:N \l_stex_module_group_depth_int
1381 \cs_new_protected:Nn \__stex_modules_begin_module: {
1382   \stex_reactivate_macro:N \STEXexport
1383   \stex_reactivate_macro:N \importmodule
1384   \stex_reactivate_macro:N \symdecl
1385   \stex_reactivate_macro:N \notation
1386   \stex_reactivate_macro:N \symdef
1387

```

```

1388 \stex_debug:nn{modules}{
1389   New~module:\\
1390   Namespace::~\l_stex_module_ns_str\\
1391   Name::~\l_stex_module_name_str\\
1392   Language::~\l_stex_module_lang_str\\
1393   Signature::~\l_stex_module_sig_str\\
1394   Metatheory::~\l_stex_module_meta_str\\
1395   File::~\stex_path_to_string:N \g_stex_currentfile_seq
1396 }
1397
1398 \seq_put_right:Nx \l_stex_all_modules_seq {
1399   \l_stex_module_ns_str ? \l_stex_module_name_str
1400 }
1401
1402 % \seq_gput_right:Nx \g_stex_modules_in_file_seq
1403 %   { \l_stex_module_ns_str ? \l_stex_module_name_str }
1404
1405
1406 \stex_if_smsmode:F{
1407   \begin{stex_annotate_env} {theory} {
1408     \l_stex_module_ns_str ? \l_stex_module_name_str
1409   }
1410
1411   \stex_annotate_invisible:nnn{header}{} {
1412     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1413     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1414     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1415       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1416     }
1417     \str_if_empty:NF \smodulotype {
1418       \stex_annotate:nnn{type}{\smodulotype}{}
1419     }
1420   }
1421 }
1422 \int_set:Nn \l_stex_module_group_depth_int {\currentgrouplevel}
1423 % TODO: Inherit metatheory for nested modules?
1424 }
1425 \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}

```

1426 \cs_new_protected:Nn \_stex_modules_end_module: {
1427 % \str_set:Nx \l_tmpa_str {
1428 %   c_stex_module_
1429 %   \prop_item:Nn \l_stex_current_module_prop { ns } ?
1430 %   \prop_item:Nn \l_stex_current_module_prop { name }
1431 %   _prop
1432 % }
1433 %%^A \prop_new:c { \l_tmpa_str }
1434 % \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
1435 \stex_debug:nn{modules}{Closing~module~\prop_item:cn {c_stex_module_\l_stex_current_module_
1436 }

```

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



smodule The core environment, with no header

```

1437 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1438 \NewDocumentEnvironment { smodule } { 0{} m } {
1439   \stex_module_setup:nn{#1}{#2}
1440   \par
1441   \stex_if_smsmode:F{
1442     \tl_clear:N \l_tmpa_tl
1443     \clist_map_inline:Nn \smoduletype {
1444       \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1445         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1446       }
1447     }
1448     \tl_if_empty:NTF \l_tmpa_tl {
1449       \__stex_modules_smodule_start:
1450     }{
1451       \l_tmpa_tl
1452     }
1453   }
1454   \__stex_modules_begin_module:
1455   \stex_ref_new_doc_target:n \smoduleid
1456   \stex_smsmode_do:
1457 } {
1458   \__stex_modules_end_module:
1459   \stex_if_smsmode:TF {
1460     % \exp_args:Nx \stex_add_to_sms:n {
1461     %   \prop_gset_from_keyval:cn {
1462     %     c_stex_module_
1463     %     \prop_item:Nn \l_stex_current_module_prop { ns } ?
1464     %     \prop_item:Nn \l_stex_current_module_prop { name }
1465     %     _prop
1466     %   } {
1467     %     name      = \prop_item:cn { \l_tmpa_str } { name } ,
1468     %     ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
1469     %     file      = \prop_item:cn { \l_tmpa_str } { file } ,
1470     %     lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
1471     %     sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
1472     %     meta      = \prop_item:cn { \l_tmpa_str } { meta }
1473     %   }
1474     % }
1475   }{
1476     \end{stex_annotate_env}
1477     \clist_set:Nn \l_tmpa_clist \smoduletype
1478     \tl_clear:N \l_tmpa_tl
1479     \clist_map_inline:Nn \l_tmpa_clist {
1480       \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1481         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1482       }
1483     }
1484     \tl_if_empty:NTF \l_tmpa_tl {
1485       \__stex_modules_smodule_end:
1486     }{
1487       \l_tmpa_tl
1488     }
1489   }

```

```

1490 }
1491
1492 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1493 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1494
1495 \newcommand\stexpatchmodule[3] [] {
1496   \str_set:Nx \l_tmpa_str{ #1 }
1497   \str_if_empty:NTF \l_tmpa_str {
1498     \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1499     \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1500   }{
1501     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1502     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1503   }
1504 }
1505

```

## 28.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n
1506 \NewDocumentCommand \STEXModule { m } {
1507   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1508   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1509   \tl_set:Nn \l_tmpa_tl {
1510     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1511   }
1512   \seq_map_inline:Nn \l_stex_all_modules_seq {
1513     \str_set:Nn \l_tmpb_str { ##1 }
1514     \str_if_eq:eeT { \l_tmpa_str } {
1515       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1516     } {
1517       \seq_map_break:n {
1518         \tl_set:Nn \l_tmpa_tl {
1519           \stex_invoke_module:n { ##1 }
1520         }
1521       }
1522     }
1523   }
1524   \l_tmpa_tl
1525 }
1526
1527 \cs_new_protected:Nn \stex_invoke_module:n {
1528   \stex_debug:nn{modules}{Invoking~module~#1}
1529   \peek_charcode_remove:NTF ! {
1530     \__stex_modules_invoke_uri:nN { #1 }
1531   } {
1532     \peek_charcode_remove:NTF ? {
1533       \__stex_modules_invoke_symbol:nn { #1 }
1534     } {
1535       \msg_error:nnx{stex}{error/syntax}{
1536         ?~or~!~expected~after~
1537         \c_backslash_str STEXModule{#1}
1538       }
1539     }
1540   }
1541 }

```

```

1539     }
1540   }
1541 }
1542
1543 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1544   \str_set:Nn #2 { #1 }
1545 }
1546
1547 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1548   \stex_invoke_symbol:n{#1?#2}
1549 }

```

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

`\stex_activate_module:n`

```

1550 \bool_new:N \l_stex_in_meta_bool
1551 \bool_set_false:N \l_stex_in_meta_bool
1552 \cs_new_protected:Nn \stex_activate_module:n {
1553   \stex_debug:nn{modules}{Activating~module~#1}
1554   \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1555     \msg_error:nnn{stex}{error/conflictingmodules}{ #1 }
1556   }
1557   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1558     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1559     \use:c{ c_stex_module_#1_code }
1560   }
1561 }

```

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

```

1562 </package>

```

## Chapter 29

# STEX -Module Inheritance Implementation

```
1563 <*package>
1564
1565 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1566
```

### 29.1 SMS Mode

```
1567 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1568 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1569 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1570 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1571
1572 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1573   \makeatletter
1574   \makeatother
1575   \ExplSyntaxOn
1576   \ExplSyntaxOff
1577   \rustexBREAK
1578 }
1579
1580 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1581   \symdef
1582   \importmodule
1583   \notation
1584   \symdecl
1585   \STEXexport
1586   \inlineass
1587   \inlinedef
1588   \inlineex
1589   \endinput
1590   \setnotation
```

```

1591 \copynotation
1592 }
1593
1594 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1595   \tl_to_str:n {
1596     smodule,
1597     copymodule,
1598     interpretmodule
1599     sdefinition,
1600     sexample,
1601     sassertion,
1602     sparagraph
1603   }
1604 }

```

(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`

```

1605 \bool_new:N \g__stex_smsmode_bool
1606 \bool_set_false:N \g__stex_smsmode_bool
1607 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1608   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1609 }

```

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

`\stex_in_smsmode:nn`

```

1610 \cs_new_protected:Nn \stex_in_smsmode:nn {
1611   \vbox_set:Nn \l_tmpa_box {
1612     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1613     \bool_gset_true:N \g__stex_smsmode_bool
1614     #2
1615     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1616   }
1617   \box_clear:N \l_tmpa_box
1618 }
1619
1620 \quark_new:N \q__stex_smsmode_break
1621
1622 %\ior_new:N \c__stex_smsmode_ior
1623 %\tl_new:N \l__stex_smsmode_filecontent_tl
1624 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1625   % \tl_clear:N \l__stex_smsmode_filecontent_tl
1626   % \ior_open:Nn \c__stex_smsmode_ior {#1}
1627   % \ior_map_inline:Nn \c__stex_smsmode_ior {
1628     %   \tl_put_right:Nn \l__stex_smsmode_filecontent_tl { ##1 }
1629   % }
1630   % \ior_close:N \c__stex_smsmode_ior
1631   \stex_filestack_push:n{#1}
1632   \stex_in_smsmode:nn{#1} {
1633     #2
1634     \everyeof{\q__stex_smsmode_break\noexpand}
1635     \expandafter\expandafter\expandafter
1636     \stex_smsmode_do:

```

```

1637 \csname @ @ input\endcsname "#1"\relax
1638 %\expandafter \stex_smsmode_do: \l__stex_smsmode_filecontent_tl
1639 }
1640 \stex_filestack_pop:
1641 }

```

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

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

```

1642 \cs_new_protected:Npn \stex_smsmode_do: {
1643   \cs_if_eq:NnF \stex_smsmode_do: \l__stex_smsmode_do:w {
1644     \stex_smsmode_do:w
1645   }
1646 }
1647 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1648   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } } {
1649     \expandafter\if\expandafter\relax\noexpand#1
1650     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1651   } \else\expandafter\__stex_smsmode_do:w\fi
1652 } {
1653   \__stex_smsmode_do:w % #1
1654 }
1655 }
1656 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1657   \cs_if_eq:NnF #1 \q__stex_smsmode_break {
1658     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1659       #1\__stex_smsmode_do:w
1660     } {
1661       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1662         #1
1663       } {
1664         \cs_if_eq:NnTF \begin #1 {
1665           \__stex_smsmode_check_begin:n
1666         } {
1667           \cs_if_eq:NnTF \end #1 {
1668             \__stex_smsmode_check_end:n
1669           } {
1670             \__stex_smsmode_do:w
1671           }
1672         }
1673       }
1674     }
1675   }
1676 }
1677 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1678   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} } {
1679     \begin{#1}
1680   } {
1681     \__stex_smsmode_do:w
1682   }
1683 }
1684 }
1685 \cs_new_protected:Nn \__stex_smsmode_check_end:n {

```

```

1686 \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1687   \end{#1}\__stex_smsmode_do:w
1688 }{
1689   \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1690 }
1691 }

```

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

## 29.2 Inheritance

```

1692 <@@=stex_importmodule>

```

`\stex_import_module_uri:nn`

```

1693 \cs_new_protected:Nn \stex_import_module_uri:nn {
1694   \str_set:Nx \l_stex_import_archive_str { #1 }
1695   \str_set:Nn \l_stex_import_path_str { #2 }
1696
1697   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1698   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1699   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1700
1701   \stex_modules_current_namespace:
1702   \bool_lazy_all:nTF {
1703     {\str_if_empty_p:N \l_stex_import_archive_str}
1704     {\str_if_empty_p:N \l_stex_import_path_str}
1705     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1706   }{
1707     \str_set_eq:NN \l_stex_import_path_str \l_stex_modules_subpath_str
1708     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1709   }{
1710     \str_if_empty:NT \l_stex_import_archive_str {
1711       \prop_if_exist:NT \l_stex_current_repository_prop {
1712         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1713       }
1714     }
1715     \str_if_empty:NTF \l_stex_import_archive_str {
1716       \str_if_empty:NF \l_stex_import_path_str {
1717         \str_set:Nx \l_stex_import_ns_str {
1718           \l_stex_module_ns_str / \l_stex_import_path_str
1719         }
1720       }
1721     }{
1722       \stex_require_repository:n \l_stex_import_archive_str
1723       \prop_get:cnN { c_stex_mathhub_\l_stex_import_archive_str_manifest_prop } { ns }
1724       \l_stex_import_ns_str
1725       \str_if_empty:NF \l_stex_import_path_str {
1726         \str_set:Nx \l_stex_import_ns_str {
1727           \l_stex_import_ns_str / \l_stex_import_path_str
1728         }
1729       }
1730     }
1731   }
1732 }

```

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

```

\l_stex_import_name_str Store the return values of \stex_import_module_uri:nn.
\l_stex_import_archive_str 1733 \str_new:N \l_stex_import_name_str
\l_stex_import_path_str 1734 \str_new:N \l_stex_import_archive_str
\l_stex_import_ns_str 1735 \str_new:N \l_stex_import_path_str
1736 \str_new:N \l_stex_import_ns_str

```

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

```

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

```



```

1780     }
1781
1782   } {
1783     \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1784     \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1785
1786     \ltx@ifpackageloaded{babel} {
1787       \exp_args:NnX \prop_get:NnNF \c_stex_language_abbrevs_prop
1788         { \language } \l_tmpb_str {
1789         \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1790       }
1791     } {
1792       \str_clear:N \l_tmpb_str
1793     }
1794
1795     \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1796
1797     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1798     \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1799       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1800     }{
1801       \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1802       \IfFileExists{ \l_tmpa_str/#4.tex }{
1803         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1804       }{
1805         % try english as default
1806         \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1807         \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1808           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1809         }{
1810           \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1811           \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1812             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1813           }{
1814             \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1815             \IfFileExists{ \l_tmpa_str.tex }{
1816               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1817             }{
1818               % try english as default
1819               \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1820               \IfFileExists{ \l_tmpa_str.en.tex }{
1821                 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1822               }{
1823                 \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1824               }
1825             }
1826           }
1827         }
1828       }
1829     }
1830   }
1831
1832   \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
1833     \seq_clear:N \l_stex_all_modules_seq

```

```

1834     \str_clear:N \l_stex_current_module_str
1835     \str_set:Nx \l_tmpb_str { #2 }
1836     \str_if_empty:NF \l_tmpb_str {
1837         \stex_set_current_repository:n { #2 }
1838     }
1839     \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1840 }
1841
1842 \stex_if_module_exists:nF { #1 ? #4 } {
1843     \msg_error:nnx{stex}{error/unknownmodule}{
1844         #1?#4~(in~file~\g__stex_importmodule_file_str)
1845     }
1846 }
1847 }
1848 \stex_activate_module:n { #1 ? #4 }
1849 }

```

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

### `\importmodule`

```

1850 \NewDocumentCommand \importmodule { 0{} m } {
1851     \stex_import_module_uri:nn { #1 } { #2 }
1852     \stex_debug:nn{modules}{Importing~module:~
1853         \l_stex_import_ns_str ? \l_stex_import_name_str
1854     }
1855     \stex_if_smsmode:F {
1856         \stex_import_require_module:nnnn
1857         { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1858         { \l_stex_import_path_str } { \l_stex_import_name_str }
1859         \stex_annotate_invisible:nnn
1860         {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1861     }
1862     \exp_args:Nx \stex_add_to_current_module:n {
1863         \stex_import_require_module:nnnn
1864         { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1865         { \l_stex_import_path_str } { \l_stex_import_name_str }
1866     }
1867     \exp_args:Nx \stex_add_import_to_current_module:n {
1868         \l_stex_import_ns_str ? \l_stex_import_name_str
1869     }
1870     \stex_smsmode_do:
1871     \ignorespacesandpars
1872 }
1873 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

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

### `\usemodule`

```

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

```

```

1881     {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1882   }
1883   \stex_smsmode_do:
1884   \ignorespacesandpars
1885 }

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

1886 \endpackage

```

# Chapter 30

## STEX -Symbols Implementation

```
1887 <*package>
1888
1889 %%%%%%%%%%%%% symbols.dtx %%%%%%%%%%%%%
1890
```

Warnings and error messages

```
1891
```

### 30.1 Symbol Declarations

```
1892 <@@=stex_symdecl>
```

`\l_stex_all_symbols_seq` Stores all available symbols

```
1893 \seq_new:N \l_stex_all_symbols_seq
```

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

`\STEXsymbol`

```
1894 \NewDocumentCommand \STEXsymbol { m } {
1895   \stex_get_symbol:n { #1 }
1896   \exp_args:No
1897   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1898 }
```

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

symdecl arguments:

```
1899 \keys_define:nn { stex / symdecl } {
1900   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1901   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1902   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1903   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
1904   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,
1905   align     .str_set:N    = \l_stex_symdecl_align_str , % TODO(?)
1906   gfc       .str_set:N    = \l_stex_symdecl_gfc_str , % TODO(?)
1907   specializes .str_set:N  = \l_stex_symdecl_specializes_str , % TODO(?)
1908   def       .tl_set:N    = \l_stex_symdecl_definiens_tl
```

```

1909 }
1910
1911 \bool_new:N \l_stex_symdecl_make_macro_bool
1912
1913 \cs_new_protected:Nn \__stex_symdecl_args:n {
1914   \str_clear:N \l_stex_symdecl_name_str
1915   \str_clear:N \l_stex_symdecl_args_str
1916   \str_clear:N \l_stex_symdecl_deprecate_str
1917   \bool_set_false:N \l_stex_symdecl_local_bool
1918   \tl_clear:N \l_stex_symdecl_type_tl
1919   \tl_clear:N \l_stex_symdecl_definiens_tl
1920
1921   \keys_set:nn { stex / symdecl } { #1 }
1922 }

```

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

```

1923
1924 \NewDocumentCommand \symdecl { s O{} m } {
1925   \__stex_symdecl_args:n { #2 }
1926   \IfBooleanTF #1 {
1927     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1928   } {
1929     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1930   }
1931   \stex_symdecl_do:n { #3 }
1932   \stex_smsmode_do:
1933 }
1934 \stex_deactivate_macro:Nn \symdecl {module~environments}

```

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

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

```

1935 \cs_new_protected:Nn \stex_symdecl_do:n {
1936   \stex_if_in_module:F {
1937     % TODO throw error? some default namespace?
1938   }
1939
1940   \str_if_empty:NT \l_stex_symdecl_name_str {
1941     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1942   }
1943
1944   \prop_if_exist:cT { l_stex_symdecl_
1945     \l_stex_current_module_str ?
1946     \l_stex_symdecl_name_str
1947   }_prop
1948   }{
1949     % TODO throw error (beware of circular dependencies)
1950   }
1951
1952   \prop_clear:N \l_tmpa_prop
1953   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
1954   \seq_clear:N \l_tmpa_seq
1955   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str

```

```

1956 \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1957
1958 \str_if_empty:NT \l_stex_symdecl_deprecate_str {
1959   \str_if_empty:NF \l_stex_module_deprecate_str {
1960     \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
1961   }
1962 }
1963 \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
1964
1965 \exp_args:No \stex_add_constant_to_current_module:n {
1966   \l_stex_symdecl_name_str
1967 }
1968
1969 % arity/args
1970 \int_zero:N \l_tmpb_int
1971
1972 \bool_set_true:N \l_tmpa_bool
1973 \str_map_inline:Nn \l_stex_symdecl_args_str {
1974   \token_case_meaning:NnF ##1 {
1975     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1976     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1977     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1978     {\tl_to_str:n a} {
1979       \bool_set_false:N \l_tmpa_bool
1980       \int_incr:N \l_tmpb_int
1981     }
1982     {\tl_to_str:n B} {
1983       \bool_set_false:N \l_tmpa_bool
1984       \int_incr:N \l_tmpb_int
1985     }
1986   }{
1987     \msg_set:nnn{stex}{error/wrongargs}{
1988       args~value~in~symbol~declaration~for~
1989       \l_stex_current_module_str ?
1990       \l_stex_symdecl_name_str ~
1991       needs~to~be~
1992       i,~a,~b~or~B,~but~##1~given
1993     }
1994     \msg_error:nn{stex}{error/wrongargs}
1995   }
1996 }
1997 \bool_if:NTF \l_tmpa_bool {
1998   % possibly numeric
1999   \str_if_empty:NTF \l_stex_symdecl_args_str {
2000     \prop_put:Nnn \l_tmpa_prop { args } {}
2001     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2002   }{
2003     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
2004     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2005     \str_clear:N \l_tmpa_str
2006     \int_step_inline:nn \l_tmpa_int {
2007       \str_put_right:Nn \l_tmpa_str i
2008     }
2009     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }

```

```

2010     }
2011 } {
2012   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2013   \prop_put:Nnx \l_tmpa_prop { arity }
2014     { \str_count:N \l_stex_symdecl_args_str }
2015 }
2016 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2017
2018
2019 % semantic macro
2020
2021 \bool_if:NT \l_stex_symdecl_make_macro_bool {
2022   \exp_args:Nx \stex_do_aftergroup:n {
2023     \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2024       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2025     }}
2026   }
2027
2028   \bool_if:NF \l_stex_symdecl_local_bool {
2029     \exp_args:Nx \stex_add_to_current_module:n {
2030       \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2031         \l_stex_current_module_str ? \l_stex_symdecl_name_str
2032       } }
2033     }
2034   }
2035 }
2036
2037 % add to all symbols
2038
2039 \bool_if:NF \l_stex_symdecl_local_bool {
2040   \exp_args:Nx \stex_add_to_current_module:n {
2041     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2042       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2043     }
2044   }
2045   % \exp_args:Nx \stex_add_field_to_current_module:n {
2046   %   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2047   % }
2048 }
2049
2050 \stex_debug:nn{symbols}{New~symbol:~
2051   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2052   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2053   Args:~\prop_item:Nn \l_tmpa_prop { args }
2054 }
2055
2056 % circular dependencies require this:
2057
2058 \prop_if_exist:cF {
2059   l_stex_symdecl_
2060   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2061   _prop
2062 } {
2063   \prop_set_eq:cN {

```

```

2064     l_stex_symdecl_
2065     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2066     _prop
2067 } \l_tmpa_prop
2068 }
2069
2070 \seq_clear:c {
2071     l_stex_symdecl_
2072     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2073     _notations
2074 }
2075
2076 \bool_if:NF \l_stex_symdecl_local_bool {
2077     \exp_args:Nx
2078     \stex_add_to_current_module:n {
2079         \seq_clear:c {
2080             l_stex_symdecl_
2081             \l_stex_current_module_str ? \l_stex_symdecl_name_str
2082             _notations
2083         }
2084         \prop_set_from_keyval:cn {
2085             l_stex_symdecl_
2086             \l_stex_current_module_str ? \l_stex_symdecl_name_str
2087             _prop
2088         } {
2089             name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2090             module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2091             type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2092             args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2093             arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2094             assoc     = \prop_item:Nn \l_tmpa_prop { assoc }     ,
2095         }
2096     }
2097 }
2098
2099 \stex_if_smsmode:TF {
2100     \bool_if:NF \l_stex_symdecl_local_bool {
2101 %         \exp_args:Nx \stex_add_to_sms:n {
2102 %             \prop_set_from_keyval:cn {
2103 %                 l_stex_symdecl_
2104 %                 \l_stex_current_module_str ? \l_stex_symdecl_name_str
2105 %                 _prop
2106 %             } {
2107 %                 name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2108 %                 module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2109 %                 local     = \prop_item:Nn \l_tmpa_prop { local }     ,
2110 %                 type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2111 %                 args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2112 %                 arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2113 %                 assoc     = \prop_item:Nn \l_tmpa_prop { assoc }     ,
2114 %             }
2115 %         \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2116 %             \l_stex_current_module_str ? \l_stex_symdecl_name_str
2117 %         }

```



```

2118 %      }
2119     }
2120   }{
2121     \exp_args:Nx \stex_do_aftergroup:n {
2122       \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2123         \l_stex_current_module_str ? \l_stex_symdecl_name_str
2124       }
2125     }
2126     \stex_if_do_html:T {
2127       \stex_annotate_invisible:nnn {symdecl} {
2128         \l_stex_current_module_str ? \l_stex_symdecl_name_str
2129       } {
2130         \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}{}{${\l_st
2131         \stex_annotate_invisible:nnn{args}{}{
2132           \prop_item:Nn \l_tmpa_prop { args }
2133         }
2134         \stex_annotate_invisible:nnn{macroname}{#1}{}
2135         \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2136           \stex_annotate_invisible:nnn{definiens}{}
2137             {${\l_stex_symdecl_definiens_tl$}
2138         }
2139       }
2140     }
2141   }
2142 }

```

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

`\stex_get_symbol:n`

```

2143 \str_new:N \l_stex_get_symbol_uri_str
2144
2145 \cs_new_protected:Nn \stex_get_symbol:n {
2146   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2147     \__stex_symdecl_get_symbol_from_cs:n { #1 }
2148   }{
2149     % argument is a string
2150     % is it a command name?
2151     \cs_if_exist:cTF { #1 }{
2152       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2153       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2154       \str_if_empty:NTF \l_tmpa_str {
2155         \exp_args:Nx \cs_if_eq:NNTF {
2156           \tl_head:N \l_tmpa_tl
2157         } \stex_invoke_symbol:n {
2158           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
2159         }{
2160           \__stex_symdecl_get_symbol_from_string:n { #1 }
2161         }
2162       } {
2163         \__stex_symdecl_get_symbol_from_string:n { #1 }
2164       }
2165     }{
2166       % argument is not a command name
2167       \__stex_symdecl_get_symbol_from_string:n { #1 }

```

```

2168     % \l_stex_all_symbols_seq
2169   }
2170 }
2171 \str_if_eq:eeF {
2172   \prop_item:cn {
2173     l_stex_symdecl\l_stex_get_symbol_uri_str _prop
2174   }{ deprecate }
2175 }{}{
2176   \msg_warning:nxxx{stex}{warning/deprecated}{
2177     Symbol~\l_stex_get_symbol_uri_str
2178   }{
2179     \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{ deprecate }
2180   }
2181 }
2182 }
2183
2184 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2185   \str_set:Nn \l_tmpa_str { #1 }
2186   \bool_set_false:N \l_tmpa_bool
2187   \stex_if_in_module:T {
2188     \exp_args:Nno \seq_if_in:cnT {c_stex_module\l_stex_current_module_str _constants} { \l_
2189       \bool_set_true:N \l_tmpa_bool
2190       \str_set:Nx \l_stex_get_symbol_uri_str {
2191         \l_stex_current_module_str ? #1
2192       }
2193     }
2194   }
2195   \bool_if:NF \l_tmpa_bool {
2196     \tl_set:Nn \l_tmpa_tl {
2197       \msg_set:nnn{stex}{error/unknownsymbol}{
2198         No~symbol~#1~found!
2199       }
2200       \msg_error:nn{stex}{error/unknownsymbol}
2201     }
2202     \str_set:Nn \l_tmpa_str { #1 }
2203     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2204     \seq_map_inline:Nn \l_stex_all_symbols_seq {
2205       \str_set:Nn \l_tmpb_str { ##1 }
2206       \str_if_eq:eeT { \l_tmpa_str } {
2207         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
2208       } {
2209         \seq_map_break:n {
2210           \tl_set:Nn \l_tmpa_tl {
2211             \str_set:Nn \l_stex_get_symbol_uri_str {
2212               ##1
2213             }
2214           }
2215         }
2216       }
2217     }
2218     \l_tmpa_tl
2219   }
2220 }
2221

```

```

2222 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
2223   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2224     { \tl_tail:N \l_tmpa_tl }
2225   \tl_if_single:NTF \l_tmpa_tl {
2226     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2227       \exp_after:wN \str_set:Nn \exp_after:wN
2228         \l_stex_get_symbol_uri_str \l_tmpa_tl
2229     }{
2230       % TODO
2231       % tail is not a single group
2232     }
2233   }{
2234     % TODO
2235     % tail is not a single group
2236   }
2237 }

```

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

## 30.2 Notations

```

2238 <@@=stex_notation>
2239 notation arguments:
2240 \keys_define:nn { stex / notation } {
2241   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2242   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2243   prec .str_set_x:N = \l__stex_notation_prec_str ,
2244   op .tl_set:N = \l__stex_notation_op_tl ,
2245   primary .bool_set:N = \l__stex_notation_primary_bool ,
2246   primary .default:n = {true} ,
2247   unknown .code:n = \str_set:Nx
2248     \l__stex_notation_variant_str \l_keys_key_str
2249 }
2250 \cs_new_protected:Nn \_stex_notation_args:n {
2251   \str_clear:N \l__stex_notation_lang_str
2252   \str_clear:N \l__stex_notation_variant_str
2253   \str_clear:N \l__stex_notation_prec_str
2254   \tl_clear:N \l__stex_notation_op_tl
2255   \bool_set_false:N \l__stex_notation_primary_bool
2256
2257   \keys_set:nn { stex / notation } { #1 }
2258 }

```

**\notation**

```

2259 \NewDocumentCommand \notation { 0{ } m } {
2260   \_stex_notation_args:n { #1 }
2261   \tl_clear:N \l_stex_symdecl_definiens_tl
2262   \stex_get_symbol:n { #2 }
2263   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
2264 }
2265 \stex_deactivate_macro:Nn \notation {module~environments}

```

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

`\stex_notation_do:nn`

```

2266 \seq_new:N \l__stex_notation_precedences_seq
2267 \tl_new:N \l__stex_notation_opprec_tl
2268 \int_new:N \l__stex_notation_currarg_int
2269
2270 \cs_new_protected:Nn \stex_notation_do:nn {
2271   \let\l_stex_current_symbol_str\relax
2272   \str_set:Nx \l__stex_notation_symbol_str { #1 }
2273   \seq_clear:N \l__stex_notation_precedences_seq
2274   \tl_clear:N \l__stex_notation_opprec_tl
2275   \prop_get:cnN {
2276     l_stex_symdecl_ #1 _prop
2277   } { args } \l__stex_notation_args_str
2278
2279   % precedences
2280   \prop_get:cnN {
2281     l_stex_symdecl_ #1 _prop
2282   } { arity } \l__stex_notation_arity_str
2283   \str_if_empty:NTF \l__stex_notation_prec_str {
2284     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2285       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2286     }{
2287       \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2288     }
2289   } {
2290     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2291       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2292       \int_step_inline:nn { \l__stex_notation_arity_str } {
2293         \exp_args:NNo
2294         \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2295       }
2296     }{
2297       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2298       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2299         \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2300         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2301           \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2302             \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2303           \seq_map_inline:Nn \l_tmpa_seq {
2304             \seq_put_right:Nn \l_tmpb_seq { ##1 }
2305           }
2306         }
2307       }{
2308         \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2309           \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2310         }{
2311           \tl_set:No \l__stex_notation_opprec_tl { 0 }
2312         }
2313       }
2314     }
2315   }
2316
2317   \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2318   \int_step_inline:nn { \l__stex_notation_arity_str } {

```

```

2319 \seq_pop_left:NnF \l_tmpa_seq \l_tmpb_str {
2320 \exp_args:NNo
2321 \seq_put_right:No \l__stex_notation_precedences_seq {
2322 \l__stex_notation_opprec_tl
2323 }
2324 }
2325 }
2326
2327 \tl_clear:N \l__stex_notation_dummyargs_tl
2328
2329 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2330 \exp_args:NNe
2331 \cs_set:Npn \l__stex_notation_macrocode_cs {
2332 \stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2333 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2334 { \l__stex_notation_opprec_tl }
2335 { \exp_not:n { #2 } }
2336 }
2337 \__stex_notation_final:
2338 }{
2339 \str_if_in:NnTF \l__stex_notation_args_str b {
2340 \exp_args:Nne \use:nn
2341 {
2342 \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2343 \cs_set:Npn \l__stex_notation_arity_str } { {
2344 \stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2345 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2346 { \l__stex_notation_opprec_tl }
2347 { \exp_not:n { #2 } }
2348 } }
2349 }{
2350 \str_if_in:NnTF \l__stex_notation_args_str B {
2351 \exp_args:Nne \use:nn
2352 {
2353 \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2354 \cs_set:Npn \l__stex_notation_arity_str } { {
2355 \stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2356 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2357 { \l__stex_notation_opprec_tl }
2358 { \exp_not:n { #2 } }
2359 } }
2360 }{
2361 \exp_args:Nne \use:nn
2362 {
2363 \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2364 \cs_set:Npn \l__stex_notation_arity_str } { {
2365 \stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2366 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2367 { \l__stex_notation_opprec_tl }
2368 { \exp_not:n { #2 } }
2369 } }
2370 }
2371 }
2372

```

```

2373 \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2374 \int_zero:N \l__stex_notation_currarg_int
2375 \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2376 \__stex_notation_arguments:
2377 }
2378 }

```

(End definition for \stex\_notation\_do:nn. This function is documented on page 37.)

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

```

2379 \cs_new_protected:Nn \__stex_notation_arguments: {
2380 \int_incr:N \l__stex_notation_currarg_int
2381 \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2382 \__stex_notation_final:
2383 }{
2384 \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2385 \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2386 \str_if_eq:VnTF \l_tmpa_str a {
2387 \__stex_notation_argument_assoc:n
2388 }{
2389 \str_if_eq:VnTF \l_tmpa_str B {
2390 \__stex_notation_argument_assoc:n
2391 }{
2392 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2393 \tl_put_right:Nx \l__stex_notation_dummyargs_tl {
2394 { \stex_term_math_arg:nnn
2395 { \int_use:N \l__stex_notation_currarg_int }
2396 { \l_tmpa_str }
2397 { ###\int_use:N \l__stex_notation_currarg_int }
2398 }
2399 }
2400 \__stex_notation_arguments:
2401 }
2402 }
2403 }
2404 }

```

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

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

```

2405 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
2406
2407 \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2408 {\l__stex_notation_arity_str}{
2409 #1
2410 }
2411 \int_zero:N \l_tmpa_int
2412 \tl_clear:N \l_tmpa_tl
2413 \str_map_inline:Nn \l__stex_notation_args_str {
2414 \int_incr:N \l_tmpa_int
2415 \tl_put_right:Nx \l_tmpa_tl {
2416 \str_if_eq:nnTF {##1}{a}{ {} }{ }{
2417 \str_if_eq:nnTF {##1}{B}{ {} }{ }{
2418 {##### \int_use:N \l_tmpa_int}

```

```

2419     }
2420   }
2421 }
2422 }
2423 \exp_after:wN\exp_after:wN\exp_after:wN \def
2424 \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2425 \exp_after:wN\exp_after:wN\exp_after:wN ##
2426 \exp_after:wN\exp_after:wN\exp_after:wN 1
2427 \exp_after:wN\exp_after:wN\exp_after:wN ##
2428 \exp_after:wN\exp_after:wN\exp_after:wN 2
2429 \exp_after:wN\exp_after:wN\exp_after:wN {
2430   \exp_after:wN \exp_after:wN \exp_after:wN
2431   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2432     \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2433   }
2434 }
2435
2436 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2437 \tl_put_right:Nx \l__stex_notation_dummyargs_tl { {
2438   \stex_term_math_assoc_arg:nnnn
2439   { \int_use:N \l__stex_notation_currarg_int }
2440   { \l_tmpa_str }
2441   { ####\int_use:N \l__stex_notation_currarg_int }
2442   { \l_tmpa_cs {####1} {####2} }
2443 } }
2444 %\cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
2445 %\tl_put_right:Nx \l_tmpa_tl {
2446 % { \stex_term_math_assoc_arg:nnnn
2447 %   { \int_use:N \l_tmpa_int }
2448 %   { \l_tmpb_str }
2449 %   \exp_args:No \exp_not:n
2450 %   {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
2451 %   { ####\int_use:N \l_tmpa_int }
2452 % }
2453 %}
2454 \__stex_notation_arguments:
2455 }

```

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

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

```

2456 \cs_new_protected:Nn \__stex_notation_final: {
2457   \exp_args:Nne \use:nn
2458   {
2459     \cs_generate_from_arg_count:cNnn {
2460       stex_notation_ \l__stex_notation_symbol_str \c_hash_str
2461       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2462       _cs
2463     }
2464     \cs_set:Npn \l__stex_notation_arity_str { { {
2465       \exp_after:wN \exp_after:wN \exp_after:wN
2466       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2467       { \exp_after:wN \l__stex_notation_macrocode_cs \l__stex_notation_dummyargs_tl }
2468     } }

```

```

2469
2470 \tl_if_empty:NF \l__stex_notation_op_tl {
2471   \cs_set:cpx {
2472     stex_op_notation_ \l__stex_notation_symbol_str \c_hash_str
2473     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2474     _cs
2475   } {
2476     \stex_term_oms:nnn {
2477       \l__stex_notation_symbol_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2478       \l__stex_notation_lang_str
2479     }{
2480       \l__stex_notation_symbol_str
2481     }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2482   }
2483 }
2484
2485 \exp_args:Ne
2486 \stex_add_to_current_module:n {
2487   \cs_generate_from_arg_count:cNnn {
2488     stex_notation_ \l__stex_notation_symbol_str \c_hash_str
2489     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2490     _cs
2491   } \cs_set:Npn {\l__stex_notation_arity_str} {
2492     \exp_after:wN \exp_after:wN \exp_after:wN
2493     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2494     { \exp_after:wN \l__stex_notation_macrocode_cs \l__stex_notation_dummyargs_tl }
2495   }
2496   \tl_if_empty:NF \l__stex_notation_op_tl {
2497     \cs_set:cpn {
2498       stex_op_notation_ \l__stex_notation_symbol_str \c_hash_str
2499       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2500       _cs
2501     } {
2502       \stex_term_oms:nnn {
2503         \l__stex_notation_symbol_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2504         \l__stex_notation_lang_str
2505       }{
2506         \l__stex_notation_symbol_str
2507       }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2508     }
2509   }
2510 }
2511 \exp_args:Nx
2512 % \stex_do_aftergroup:n {
2513   \seq_put_right:cx {
2514     l_stex_symdecl_ \l__stex_notation_symbol_str
2515     _notations
2516   } {
2517     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2518   }
2519 % }
2520
2521 \stex_debug:nn{symbols}{
2522   Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str

```



```

2523 ~for~\l__stex_notation_symbol_str^^J
2524 Operator~precedence:~\l__stex_notation_opprec_tl^^J
2525 Argument~precedences:~
2526   \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2527 Notation: \cs_meaning:c {
2528   stex_notation_ \l__stex_notation_symbol_str \c_hash_str
2529   \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2530   _cs
2531 }
2532 }
2533
2534 %\prop_set_eq:cn {
2535 %   l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2536 %   \c_hash_str \l__stex_notation_lang_str _prop
2537 %} \l_tmpb_prop
2538
2539 \exp_args:Ne
2540 \stex_add_to_current_module:n {
2541   \seq_put_right:cn {
2542     l_stex_symdecl_ \l__stex_notation_symbol_str
2543     _notations
2544   } {
2545     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2546   }
2547   %\prop_set_from_keyval:cn {
2548   %   l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2549   %   \c_hash_str \l__stex_notation_lang_str _prop
2550   %} {
2551   %   symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2552   %   language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2553   %   variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,
2554   %   opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2555   %   argprecs    = \prop_item:Nn \l_tmpb_prop { argprecs }    ,
2556   %}
2557 }
2558
2559 \stex_if_smsmode:TF {
2560 %   \exp_args:Nx \stex_add_to_sms:n {
2561 %     \prop_set_from_keyval:cn {
2562 %       l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2563 %       \c_hash_str \l__stex_notation_lang_str _prop
2564 %     } {
2565 %       symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2566 %       language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2567 %       variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,
2568 %       opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2569 %       argprecs    = \prop_item:Nn \l_tmpb_prop { argprecs }    ,
2570 %     }
2571 %   }
2572 }{
2573
2574 % HTML annotations
2575 \stex_if_do_html:T {
2576   \stex_annotate_invisible:nnn { notation }

```

```

2577 { \l__stex_notation_symbol_str } {
2578   \stex_annotate_invisible:nnn { notationfragment }
2579   { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
2580   \stex_annotate_invisible:nnn { precedence }
2581   { \l__stex_notation_prec_str }{}
2582
2583   \int_zero:N \l_tmpa_int
2584   \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2585   \tl_clear:N \l_tmpa_tl
2586   \int_step_inline:nn { \l__stex_notation_arity_str }{
2587     \int_incr:N \l_tmpa_int
2588     \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2589     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_r
2590     \str_if_eq:VnTF \l_tmpb_str a {
2591       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2592         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2593         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2594       } }
2595     }{
2596       \str_if_eq:VnTF \l_tmpb_str B {
2597         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2598           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2599           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2600         } }
2601       }{
2602         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2603           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2604         } }
2605       }
2606     }
2607   }
2608   \stex_annotate_invisible:nnn { notationcomp }{}{
2609     \str_set:Nx \l_stex_current_symbol_str { \l__stex_notation_symbol_str }
2610     $ \exp_args:Nno \use:nn { \use:c {
2611       stex_notation_ \l_stex_current_symbol_str
2612       \c_hash_str \l__stex_notation_variant_str
2613       \c_hash_str \l__stex_notation_lang_str _cs
2614     } } { \l_tmpa_tl } $
2615   }
2616 }
2617 }
2618 }
2619 \stex_smsmode_do:
2620 }

```

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

\setnotation

```

2621 \keys_define:nn { stex / setnotation } {
2622   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2623   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2624   unknown .code:n = \str_set:Nx
2625     \l__stex_notation_variant_str \l_keys_key_str
2626 }

```

```

2627
2628 \cs_new_protected:Nn \stex_setnotation_args:n {
2629   \str_clear:N \l__stex_notation_lang_str
2630   \str_clear:N \l__stex_notation_variant_str
2631   \keys_set:nn { stex / setnotation } { #1 }
2632 }
2633
2634 \NewDocumentCommand \setnotation {m m} {
2635   \stex_get_symbol:n { #1 }
2636   \stex_setnotation_args:n { #2 }
2637   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl \l_stex_get_symbol_uri_str _notations }
2638   { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2639     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl \l_stex_get_symbol_uri_str _notation
2640     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2641     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl \l_stex_get_symbol_uri_str _notation
2642     { \c_hash_str }
2643     \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl \l_stex_get_symbol_uri_str _notations
2644     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2645     \exp_args:Nx \stex_add_to_current_module:n {
2646       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl \l_stex_get_symbol_uri_str _notati
2647       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2648       \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl \l_stex_get_symbol_uri_str _notation
2649       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2650       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl \l_stex_get_symbol_uri_str _notati
2651       { \c_hash_str }
2652     }
2653     \stex_debug:nn {notations}{
2654       Setting~default~notation~
2655       {\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str}~for~
2656       \l_stex_get_symbol_uri_str \
2657       \expandafter\meaning\csname
2658       l_stex_symdecl \l_stex_get_symbol_uri_str _notations\endcsname
2659     }
2660   }{
2661     % todo throw error
2662   }
2663   \stex_smsmode_do:
2664 }
2665
2666 \cs_new_protected:Nn \stex_copy_notations:nn {
2667   \stex_debug:nn {notations}{
2668     Copying~notations~from~#2~to~#1\
2669     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
2670   }
2671   \tl_clear:N \l_tmpa_tl
2672   \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2673     \tl_put_right:Nn \l_tmpa_tl { {##} ##1} }
2674   }
2675   \seq_map_inline:cn {l_stex_symdecl_#2_notations}{
2676     \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2677     \edef \l_tmpa_tl {
2678       \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2679       \exp_after:wN\exp_after:wN\exp_after:wN {
2680         \exp_after:wN \l_tmpa_cs \l_tmpa_tl

```

```

2681     }
2682   }
2683   \exp_args:Nx
2684   \stex_do_aftergroup:n {
2685     \seq_put_right:cn{l_stex_symdecl_#1_notations}{##1}
2686     \cs_generate_from_arg_count:cNnn {
2687       stex_notation_ #1 \c_hash_str ##1 _cs
2688     } \cs_set:Npn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }{
2689       \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl}
2690     }
2691   }
2692 }
2693 }
2694
2695 \NewDocumentCommand \copynotation {m m} {
2696   \stex_get_symbol:n { #1 }
2697   \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2698   \stex_get_symbol:n { #2 }
2699   \exp_args:Noo
2700   \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2701   \exp_args:Nx \stex_add_import_to_current_module:n{
2702     \stex_copy_notations:nn {\l_tmpa_str} {\l_stex_get_symbol_uri_str}
2703   }
2704   \stex_smsmode_do:
2705 }
2706

```

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

## \symdef

```

2707 \keys_define:nn { stex / symdef } {
2708   name .str_set_x:N = \l_stex_symdecl_name_str ,
2709   local .bool_set:N = \l_stex_symdecl_local_bool ,
2710   args .str_set_x:N = \l_stex_symdecl_args_str ,
2711   type .tl_set:N = \l_stex_symdecl_type_tl ,
2712   def .tl_set:N = \l_stex_symdecl_definiens_tl ,
2713   op .tl_set:N = \l_stex_notation_op_tl ,
2714   lang .str_set_x:N = \l__stex_notation_lang_str ,
2715   variant .str_set_x:N = \l__stex_notation_variant_str ,
2716   prec .str_set_x:N = \l__stex_notation_prec_str ,
2717   unknown .code:n = \str_set:Nx
2718     \l__stex_notation_variant_str \l_keys_key_str
2719 }
2720
2721 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2722   \str_clear:N \l_stex_symdecl_name_str
2723   \str_clear:N \l_stex_symdecl_args_str
2724   \bool_set_false:N \l_stex_symdecl_local_bool
2725   \tl_clear:N \l_stex_symdecl_type_tl
2726   \tl_clear:N \l_stex_symdecl_definiens_tl
2727   \str_clear:N \l__stex_notation_lang_str
2728   \str_clear:N \l__stex_notation_variant_str
2729   \str_clear:N \l__stex_notation_prec_str
2730   \tl_clear:N \l__stex_notation_op_tl

```

```

2731 \keys_set:nn { stex / symdef } { #1 }
2732 }
2733 }
2734
2735 \NewDocumentCommand \symdef { 0{} m } {
2736   \__stex_notation_symdef_args:n { #1 }
2737   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2738   \stex_symdecl_do:n { #2 }
2739   \exp_args:Nx \stex_notation_do:nn {
2740     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2741   }
2742 }
2743 \stex_deactivate_macro:Nn \symdef {module~environments}

```

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

```

2744 \endpackage

```

## Chapter 31

# STEX -Terms Implementation

```
2745 <*package>
2746
2747 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2748
2749 <@@=stex_terms>
2750
2751 Warnings and error messages
2752 \msg_new:nnn{stex}{error/nonotation}{
2753   Symbol~#1~invoked,~but~has~no~notation~#2!
2754 }
2755 \msg_new:nnn{stex}{error/notationarg}{
2756   Error~in~parsing~notation~#1
2757 }
2758 \msg_new:nnn{stex}{error/noop}{
2759   Symbol~#1~has~no~operator~notation~for~notation~#2
2760 }
```

### 31.1 Symbol Invocations

Arguments:

```
2760 \keys_define:nn { stex / terms } {
2761   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2762   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2763   unknown .code:n = \str_set:Nx
2764     \l__stex_terms_variant_str \l_keys_key_str
2765 }
2766
2767 \cs_new_protected:Nn \__stex_terms_args:n {
2768   \str_clear:N \l__stex_terms_lang_str
2769   \str_clear:N \l__stex_terms_variant_str
2770   \str_clear:N \l__stex_terms_prec_str
2771   \tl_clear:N \l__stex_terms_op_tl
2772
2773   \keys_set:nn { stex / terms } { #1 }
```

2774 }

`\stex_invoke_symbol:n` Invokes a semantic macro

```

2775 \cs_new_protected:Nn \stex_invoke_symbol:n {
2776   \str_if_eq:eeF {
2777     \prop_item:cn {
2778       l_stex_symdecl_#1_prop
2779     }{ deprecate }
2780   }{}{
2781     \msg_warning:nxxx{stex}{warning/deprecated}{
2782       Symbol~#1
2783     }{
2784       \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
2785     }
2786   }
2787   \if_mode_math:
2788     \exp_after:wN \__stex_terms_invoke_math:n
2789   \else:
2790     \exp_after:wN \__stex_terms_invoke_text:n
2791   \fi: { #1 }
2792 }
```

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

`\__stex_terms_invoke_math:n`

```

2793 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2794   \peek_charcode_remove:NTF ! {
2795     \peek_charcode:NTF [ {
2796       \__stex_terms_invoke_op:nw { #1 }
2797     }{
2798       \peek_charcode_remove:NTF ! {
2799         \peek_charcode:NTF [ {
2800           \__stex_terms_invoke_op_custom:nw
2801         }{
2802           % TODO throw error
2803         }
2804       }{
2805         \__stex_terms_invoke_op:nw { #1 } []
2806       }
2807     }
2808   }{
2809     \peek_charcode_remove:NTF * {
2810       \__stex_terms_invoke_text:n { #1 }
2811     }{
2812       \peek_charcode:NTF [ {
2813         \__stex_terms_invoke_math:nw { #1 }
2814       }{
2815         \__stex_terms_invoke_math:nw { #1 } []
2816       }
2817     }
2818   }
2819 }
```

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

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

```

2820 \cs_new_protected:Npn \_stex_terms_invoke_op_custom:nw #1 [#2] {
2821   \_stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
2822     \stex_highlight_term:nn{#1}{#2}
2823   }
2824 }

```

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

\\_stex\_terms\_invoke\_op:nw

```

2825 \cs_new_protected:Npn \_stex_terms_invoke_op:nw #1 [#2] {
2826   \_stex_terms_args:n { #2 }
2827   \cs_if_exist:cTF {
2828     stex_op_notation_ #1 \c_hash_str
2829     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str_cs
2830   }{
2831     \csname stex_op_notation_ #1 \c_hash_str
2832       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str_cs
2833     \endcsname
2834   }{
2835     \msg_error:nnxx{stex}{error/noop}{#1}{\l__stex_terms_variant_str \c_hash_str \l__stex_t
2836   }
2837 }

```

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

\\_stex\_terms\_invoke\_math:nw

```

2838 \cs_new_protected:Npn \_stex_terms_invoke_math:nw #1 [#2] {
2839   \_stex_terms_args:n { #2 }
2840   \seq_if_empty:cTF {
2841     l_stex_symdecl_ #1 _notations
2842   } {
2843     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
2844   } {
2845     \seq_if_in:cxTF {
2846       l_stex_symdecl_ #1 _notations
2847     }
2848     { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
2849       \str_set:Nn \l_stex_current_symbol_str { #1 }
2850       \stex_debug:nn{terms}{Using~
2851         #1\c_hash_str\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str \
2852         \expandafter\meaning\csname stex_notation_ #1 \c_hash_str
2853         \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2854         _cs\endcsname
2855       }
2856       \use:c{
2857         stex_notation_ #1 \c_hash_str
2858         \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2859         _cs
2860       }
2861     }{
2862       \str_if_empty:NTF \l__stex_terms_variant_str {
2863         \str_if_empty:NTF \l__stex_terms_lang_str {
2864           \seq_get_left:cN {

```



```

2865         l_stex_symdecl_ #1 _notations
2866     } \l_tmpa_str
2867     \str_set:Nn \l_stex_current_symbol_str { #1 }
2868     \stex_debug:nn{terms}{Using~
2869         #1\c_hash_str\l_tmpa_str \\
2870         \expandafter\meaning\csname stex_notation_ #1 \c_hash_str
2871         \l_tmpa_str
2872         _cs\endcsname
2873     }
2874     \use:c{
2875         stex_notation_ #1 \c_hash_str \l_tmpa_str
2876         _cs
2877     }
2878     ){
2879         \msg_error:nnxx{stex}{error/nonotation}{#1}{
2880             ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2881         }
2882     }
2883     ){
2884         \msg_error:nnxx{stex}{error/nonotation}{#1}{
2885             ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2886         }
2887     }
2888 }
2889 }
2890 }

```

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

\\_stex\_terms\_invoke\_text:n

```

2891 \cs_new_protected:Nn \_stex_terms_invoke_text:n {
2892     \peek_charcode_remove:NTF ! {
2893         \stex_term_custom:nn { #1 } { }
2894     }{
2895         \prop_set_eq:Nc \l_tmpa_prop {
2896             l_stex_symdecl_ #1 _prop
2897         }
2898         \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2899         \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2900     }
2901 }

```

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

## 31.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
2902 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2903 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2904 \int_new:N \l__stex_terms_downprec
2905 \int_set_eq:NN \l__stex_terms_downprec \infprec

```

(End definition for `\infprec`, `\neginfprec`, and `\l__stex_terms_downprec`. These variables are documented on page 39.)

Bracketing:

`\l__stex_terms_left_bracket_str`  
`\l__stex_terms_right_bracket_str`

```
2906 \tl_set:Nn \l__stex_terms_left_bracket_str (
2907 \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

```
2908 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
2909   \bool_if:NTF \l__stex_terms_brackets_done_bool {
2910     \bool_set_false:N \l__stex_terms_brackets_done_bool
2911     #2
2912   } {
2913     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
2914       \bool_if:NTF \l__stex_inarray_bool { #2 } {
2915         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
2916         \dobrackets { #2 }
2917       }
2918     }{ #2 }
2919   }
2920 }
```

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

**`\dobrackets`**

```
2921 \bool_new:N \l__stex_terms_brackets_done_bool
2922 %\RequirePackage{scalerel}
2923 \cs_new_protected:Npn \dobrackets #1 {
2924   %\ThisStyle{\if D\m@switch
2925   %   \exp_args:Nnx \use:nn
2926   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
2927   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
2928   %   \else
2929   %   \exp_args:Nnx \use:nn
2930   %   {
2931     \bool_set_true:N \l__stex_terms_brackets_done_bool
2932     \int_set:Nn \l__stex_terms_downprec \infprec
2933     \l__stex_terms_left_bracket_str
2934     #1
2935   }
2936   {
2937     \bool_set_false:N \l__stex_terms_brackets_done_bool
2938     \l__stex_terms_right_bracket_str
2939     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2940   }
2941   %\fi}
2942 }
```

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

**\withbrackets**

```
2943 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2944   \exp_args:Nnx \use:nn
2945   {
2946     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
2947     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
2948     #3
2949   }
2950   {
2951     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
2952     {\l__stex_terms_left_bracket_str}
2953     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
2954     {\l__stex_terms_right_bracket_str}
2955   }
2956 }
```

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

**\STEXinvisible**

```
2957 \cs_new_protected:Npn \STEXinvisible #1 {
2958   \stex_annotate_invisible:n { #1 }
2959 }
```

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

OMDoc terms:

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

```
2960 \cs_new_protected:Nn \_stex_term_oms:nnn {
2961   \stex_annotate:nnn{ OMID }{ #2 }{
2962     \stex_highlight_term:nn { #1 } { #3 }
2963   }
2964 }
2965
2966 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2967   \_stex_terms_maybe_brackets:nn { #3 }{
2968     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2969   }
2970 }
```

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

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

```
2971 \cs_new_protected:Nn \_stex_term_oma:nnn {
2972   \stex_annotate:nnn{ OMA }{ #2 }{
2973     \stex_highlight_term:nn { #1 } { #3 }
2974   }
2975 }
2976
2977 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2978   \_stex_terms_maybe_brackets:nn { #3 }{
2979     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2980   }
2981 }
```

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

`\_stex_term_math_omb:nnnn`

```

2982 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2983   \stex_annotate:nnn{ OMBIND }{ #2 }{
2984     \stex_highlight_term:nn { #1 } { #3 }
2985   }
2986 }
2987
2988 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2989   \__stex_terms_maybe_brackets:nn { #3 }{
2990     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2991   }
2992 }

```

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

`\_stex_term_math_arg:nnn`

```

2993 \cs_new_protected:Nn \_stex_term_arg:nn {
2994   \stex_unhighlight_term:n {
2995     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2996   }
2997 }
2998 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
2999   \exp_args:Nnx \use:nn
3000   { \int_set:Nn \l__stex_terms_downprec { #2 }
3001     \_stex_term_arg:nn { #1 }{ #3 }
3002   }
3003   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3004 }

```

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

`\_stex_term_math_assoc_arg:nnnn`

```

3005 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
3006   % TODO sequences
3007   \clist_set:Nn \l_tmpa_clist{ #3 }
3008   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3009     \tl_set:Nn \l_tmpa_tl { #3 }
3010   }{
3011     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3012     \clist_reverse:N \l_tmpa_clist
3013     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3014
3015     \clist_map_inline:Nn \l_tmpa_clist {
3016       \exp_args:NNNo \exp_args:NNNo \tl_set:No \l_tmpa_tl {
3017         \exp_args:Nno
3018         \l_tmpa_cs { ##1 } \l_tmpa_tl
3019       }
3020     }
3021   }
3022   \exp_args:Nnno
3023   \_stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
3024 }

```

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

`\stex_term_custom:nn`

```
3025 \cs_new_protected:Nn \stex_term_custom:nn {
3026   \str_set:Nn \l__stex_terms_custom_uri { #1 }
3027   \str_set:Nn \l_tmpa_str { #2 }
3028   \tl_clear:N \l_tmpa_tl
3029   \int_zero:N \l_tmpa_int
3030   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
3031   \__stex_terms_custom_loop:
3032 }
```

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

`\__stex_terms_custom_loop:`

```
3033 \cs_new_protected:Nn \__stex_terms_custom_loop: {
3034   \bool_set_false:N \l_tmpa_bool
3035   \bool_while_do:nn {
3036     \str_if_eq_p:ee X {
3037       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
3038     }
3039   }{
3040     \int_incr:N \l_tmpa_int
3041   }
3042
3043   \peek_charcode:NTF [ {
3044     % notation/text component
3045     \__stex_terms_custom_component:w
3046   } {
3047     \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
3048       % all arguments read => finish
3049       \__stex_terms_custom_final:
3050     } {
3051       % arguments missing
3052       \peek_charcode_remove:NTF * {
3053         % invisible, specific argument position or both
3054         \peek_charcode:NTF [ {
3055           % visible specific argument position
3056           \__stex_terms_custom_arg:wn
3057         } {
3058           % invisible
3059           \peek_charcode_remove:NTF * {
3060             % invisible specific argument position
3061             \__stex_terms_custom_arg_inv:wn
3062           } {
3063             % invisible next argument
3064             \__stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
3065           }
3066         }
3067       } {
3068         % next normal argument
3069         \__stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
3070       }
3071     }
3072   }
3073 }
```

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

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

```

3074 \cs_new_protected:Npn \_stex_terms_custom_arg_inv:wn [ #1 ] #2 {
3075   \bool_set_true:N \l_tmpa_bool
3076   \_stex_terms_custom_arg:wn [ #1 ] { #2 }
3077 }

```

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

\\_stex\_terms\_custom\_arg:wn

```

3078 \cs_new_protected:Npn \_stex_terms_custom_arg:wn [ #1 ] #2 {
3079   \str_set:Nx \l_tmpb_str {
3080     \str_item:Nn \l_tmpa_str { #1 }
3081   }
3082   \str_case:VnTF \l_tmpb_str {
3083     { X } {
3084       \msg_error:nnx{stex}{error/notationarg}{\l_stex_terms_custom_uri}
3085     }
3086     { i } { \_stex_terms_custom_set_X:n { #1 } }
3087     { b } { \_stex_terms_custom_set_X:n { #1 } }
3088     { a } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
3089     { B } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
3090   }{ }{
3091     \msg_error:nnx{stex}{error/notationarg}{\l_stex_terms_custom_uri}
3092   }
3093
3094   \bool_if:nTF \l_tmpa_bool {
3095     \tl_put_right:Nx \l_tmpa_tl {
3096       \stex_annotate_invisible:n {
3097         \stex_term_arg:nn { \int_eval:n { #1 } }
3098         \exp_not:n { { #2 } }
3099       }
3100     }
3101   } {
3102     \tl_put_right:Nx \l_tmpa_tl {
3103       \stex_term_arg:nn { \int_eval:n { #1 } }
3104       \exp_not:n { { #2 } }
3105     }
3106   }
3107
3108   \_stex_terms_custom_loop:
3109 }

```

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

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

```

3110 \cs_new_protected:Nn \_stex_terms_custom_set_X:n {
3111   \str_set:Nx \l_tmpa_str {
3112     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
3113     X
3114     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
3115   }
3116 }

```

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

\\_stex\_terms\_custom\_component:

```

3117 \cs_new_protected:Npn \_stex_terms_custom_component:w [ #1 ] {
3118   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
3119   \_stex_terms_custom_loop:
3120 }

```

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

\\_stex\_terms\_custom\_final:

```

3121 \cs_new_protected:Nn \_stex_terms_custom_final: {
3122   \int_compare:nNnTF \l_tmpb_int = 0 {
3123     \exp_args:Nnno \_stex_term_oms:nnn
3124   }{
3125     \str_if_in:NnTF \l_tmpa_str {b} {
3126       \exp_args:Nnno \_stex_term_ombind:nnn
3127     } {
3128       \exp_args:Nnno \_stex_term_oma:nnn
3129     }
3130   }
3131   { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
3132 }

```

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

\symref

\symname

```

3133 \NewDocumentCommand \symref { m m }{
3134   \let\compemph_uri_prev:\compemph@uri
3135   \let\compemph@uri\symrefemph@uri
3136   \STEXsymbol{#1}!{#2}
3137   \let\compemph@uri\compemph_uri_prev:
3138 }
3139
3140 \keys_define:nn { stex / symname } {
3141   post .str_set_x:N = \l_stex_symname_post_str
3142 }
3143
3144 \cs_new_protected:Nn \stex_symname_args:n {
3145   \str_clear:N \l_stex_symname_post_str
3146   \keys_set:nn { stex / symname } { #1 }
3147 }
3148
3149 \NewDocumentCommand \symname { 0{ } m }{
3150   \stex_symname_args:n { #1 }
3151   \stex_get_symbol:n { #2 }
3152   \str_set:Nx \l_tmpa_str {
3153     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3154   }
3155   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3156
3157   \let\compemph_uri_prev:\compemph@uri
3158   \let\compemph@uri\symrefemph@uri
3159   \exp_args:NNx \use:nn

```

```

3160 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }! [
3161   \l_tmpa_str \l_stex_symname_post_str
3162 ] }
3163 \let\compemph@uri\compemph_uri_prev:
3164 }

```

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

## 31.3 Notation Components

```

3165 <@@=stex_notationcomps>

```

`\stex_highlight_term:nn`

```

3166
3167 \str_new:N \l_stex_current_symbol_str
3168 \cs_new_protected:Nn \stex_highlight_term:nn {
3169   \exp_args:Nnx
3170   \use:nn {
3171     \str_set:Nx \l_stex_current_symbol_str { #1 }
3172     #2
3173   } {
3174     \str_set:Nx \exp_not:N \l_stex_current_symbol_str
3175     { \l_stex_current_symbol_str }
3176   }
3177 }
3178
3179 \cs_new_protected:Nn \stex_unhighlight_term:n {
3180   % \latexml_if:TF {
3181   %   #1
3182   % } {
3183   %   \rustex_if:TF {
3184   %     #1
3185   %   } {
3186     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
3187   % }
3188   % }
3189 }

```

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

```

\comp
\compemph@uri
\compemph
\defemph
\defemph@uri
\symrefemph
\symrefemph@uri
3190 \cs_new_protected:Npn \comp #1 {
3191   \str_if_empty:NF \l_stex_current_symbol_str {
3192     \rustex_if:TF {
3193       \stex_annotate:nnn { comp } { \l_stex_current_symbol_str } { #1 }
3194     } {
3195       \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3196     }
3197   }
3198 }
3199
3200 \cs_new_protected:Npn \compemph@uri #1 #2 {
3201   \compemph { #1 }
3202 }

```



```

3203
3204
3205 \cs_new_protected:Npn \compemph #1 {
3206     #1
3207 }
3208
3209 \cs_new_protected:Npn \defemph@uri #1 #2 {
3210     \defemph{#1}
3211 }
3212
3213 \cs_new_protected:Npn \defemph #1 {
3214     \textbf{#1}
3215 }
3216
3217 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3218     \symrefemph{#1}
3219 }
3220
3221 \cs_new_protected:Npn \symrefemph #1 {
3222     \textbf{#1}
3223 }

```

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

## `\ellipses`

```

3224 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix
\parrayline
\parraylineh
\parraycell
3225 \bool_new:N \l_stex_inarray_bool
3226 \bool_set_false:N \l_stex_inarray_bool
3227 \NewDocumentCommand \parray { m m } {
3228     \begin{group}
3229     \bool_set_true:N \l_stex_inarray_bool
3230     \begin{array}{#1}
3231         #2
3232     \end{array}
3233     \end{group}
3234 }
3235
3236 \NewDocumentCommand \prmatrix { m } {
3237     \begin{group}
3238     \bool_set_true:N \l_stex_inarray_bool
3239     \begin{matrix}
3240         #1
3241     \end{matrix}
3242     \end{group}
3243 }
3244
3245 \def \maybepline {
3246     \bool_if:NT \l_stex_inarray_bool {\hline}
3247 }
3248
3249 \def \parrayline #1 #2 {

```

```

3250   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3251 }
3252
3253 \def \pmrow #1 { \parrayline{ }{ #1 } }
3254
3255 \def \parraylineh #1 #2 {
3256   #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3257 }
3258
3259 \def \parraycell #1 {
3260   #1 \bool_if:NT \l_stex_inarray_bool {&}
3261 }

```

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

```

3262 \endpackage

```

## Chapter 32

# STEX -Structural Features Implementation

```
3263 <*package>
3264
3265 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3266
3267 <@@=stex_features>
3268
3269 Warnings and error messages
3270 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3271   Symbol~#1~can~not~be~assigned~in~copymodule~#2
3272 }
3273 \msg_new:nnn{stex}{error/interpretmodule/noddefinies}{
3274   Symbol~#1~not~assigned~in~interpretmodule~#2
3275 }
```

### 32.1 Imports with modification

```
3275 \cs_new_protected:Nn \stex_get_symbol_in_copymodule:n {
3276   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3277     \__stex_features_get_symbol_from_cs:n { #1 }
3278   }{
3279     % argument is a string
3280     % is it a command name?
3281     \cs_if_exist:cTF { #1 }{
3282       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3283       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3284       \str_if_empty:NNTF \l_tmpa_str {
3285         \exp_args:Nx \cs_if_eq:NNTF {
3286           \tl_head:N \l_tmpa_tl
3287         } \stex_invoke_symbol:n {
3288           \exp_args:No \__stex_features_get_symbol_from_cs:n { \use:c { #1 } }
3289         }{
3290           \__stex_features_get_symbol_from_string:n { #1 }
3291         }
3292       }
3293     }
3294   }
```

```

3291     }
3292   } {
3293     \__stex_features_get_symbol_from_string:n { #1 }
3294   }
3295   }{
3296     % argument is not a command name
3297     \__stex_features_get_symbol_from_string:n { #1 }
3298     % \l_stex_all_symbols_seq
3299   }
3300 }
3301 }
3302
3303 \cs_new_protected:Nn \__stex_features_get_symbol_from_string:n {
3304   \str_set:Nn \l_tmpa_str { #1 }
3305   \bool_set_false:N \l_tmpa_bool
3306   \bool_if:NF \l_tmpa_bool {
3307     \tl_set:Nn \l_tmpa_tl {
3308       \msg_set:nnn{stex}{error/unknownsymbol}{
3309         No~symbol~#1~found!
3310       }
3311       \msg_error:nn{stex}{error/unknownsymbol}
3312     }
3313     \str_set:Nn \l_tmpa_str { #1 }
3314     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3315     \seq_map_inline:Nn \l__stex_features_copymodule_fields_seq {
3316       \str_set:Nn \l_tmpb_str { ##1 }
3317       \str_if_eq:eeT { \l_tmpa_str } {
3318         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3319       } {
3320         \seq_map_break:n {
3321           \tl_set:Nn \l_tmpa_tl {
3322             \str_set:Nn \l_stex_get_symbol_uri_str {
3323               ##1
3324             }
3325             \__stex_features_get_symbol_check:
3326           }
3327         }
3328       }
3329     }
3330     \l_tmpa_tl
3331   }
3332 }
3333
3334 \cs_new_protected:Nn \__stex_features_get_symbol_from_cs:n {
3335   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3336   { \tl_tail:N \l_tmpa_tl }
3337   \tl_if_single:NTF \l_tmpa_tl {
3338     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3339       \exp_after:wN \str_set:Nn \exp_after:wN
3340         \l_stex_get_symbol_uri_str \l_tmpa_tl
3341       \__stex_features_get_symbol_check:
3342     }{
3343       % TODO
3344       % tail is not a single group

```

```

3345     }
3346   }{
3347     % TODO
3348     % tail is not a single group
3349   }
3350 }
3351
3352 \cs_new_protected:Nn \__stex_features_get_symbol_check: {
3353   \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq {?} \l_stex_get_symbol_uri_str
3354   \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 3 {
3355     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
3356     \str_set:Nx \l_tmpa_str {\seq_use:Nn \l_tmpa_seq ?}
3357     \seq_if_in:NoF \l__stex_features_copymodule_modules_seq \l_tmpa_str {
3358       \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3359         \l_stex_current_copymodule_name_str\Allowed:~\seq_use:Nn \l__stex_features_copymodule_modules_seq \l_tmpa_str
3360       }
3361     }
3362   }{
3363     \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3364       \l_stex_current_copymodule_name_str~(inexplicably)
3365     }
3366   }
3367 }
3368
3369 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
3370   \stex_import_module_uri:nn { #1 } { #2 }
3371   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3372   \stex_import_require_module:nnnn
3373     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3374     { \l_stex_import_path_str } { \l_stex_import_name_str }
3375   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3376   \seq_set_eq:NN \l__stex_features_copymodule_modules_seq \l_stex_collect_imports_seq
3377   \seq_clear:N \l__stex_features_copymodule_fields_seq
3378   \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3379     \seq_map_inline:cn {c_stex_module_###1_constants}{
3380       \exp_args:NNx \seq_put_right:Nn \l__stex_features_copymodule_fields_seq {
3381         ###1 ? ####1
3382       }
3383     }
3384   }
3385   \seq_clear:N \l_tmpa_seq
3386   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3387     name      = \l_stex_current_copymodule_name_str ,
3388     module    = \l_stex_current_module_str ,
3389     from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3390     includes  = \l_tmpa_seq ,
3391     fields    = \l_tmpa_seq
3392   }
3393   \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3394     as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3395   \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_features_copymodule_modules_seq
3396     \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_features_copymodule_fields_seq {,~}
3397   \stex_if_smsmode:F {
3398     \begin{stex_annotate_env} {#4} {

```

```

3399     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3400   }
3401   \stex_annotate_invisible:nnn{from}{\l_stex_import_ns_str ?\l_stex_import_name_str}{\}
3402 }
3403 \bool_set_eq:NN \l__stex_features_oldhtml_bool \l_stex_html_do_output_bool
3404 \bool_set_false:N \l_stex_html_do_output_bool
3405 }
3406 \cs_new_protected:Nn \stex_copymodule_end:n {
3407   \def \l_tmpa_cs ##1 ##2 {#1}
3408   \bool_set_eq:NN \l_stex_html_do_output_bool \l__stex_features_oldhtml_bool
3409   \tl_clear:N \l_tmpa_tl
3410   \tl_clear:N \l_tmpb_tl
3411   \prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3412   \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3413     \seq_map_inline:cn {c_stex_module_##1_constants}{
3414       \tl_clear:N \l_tmpc_tl
3415       \l_tmpa_cs{##1}{####1}
3416       \str_if_exist:cTF {l__stex_features_copymodule_##1?####1_name_str} {
3417         \tl_put_right:Nx \l_tmpa_tl {
3418           \prop_set_from_keyval:cn {
3419             l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_#
3420           }{
3421             \exp_after:wN \prop_to_keyval:N \csname
3422               l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_#
3423             \endcsname
3424           }
3425           \seq_clear:c {
3426             l_stex_symdecl_
3427             \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_name_s
3428             _notations
3429           }
3430         }
3431         \tl_put_right:Nx \l_tmpc_tl {
3432           \stex_copy_notations:nn {\l_stex_current_module_str ? \use:c{l__stex_features_copy
3433           \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_features_copymodule_##1?####1_n
3434         }
3435         \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{l__stex_features_
3436         \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3437           \tl_put_right:Nx \l_tmpc_tl {
3438             \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?#
3439           }
3440           \tl_put_right:Nx \l_tmpa_tl {
3441             \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3442             \stex_invoke_symbol:n {
3443               \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_na
3444             }
3445           }
3446         }
3447       }
3448     }{
3449       \tl_put_right:Nx \l_tmpc_tl {
3450         \stex_copy_notations:nn {\l_stex_current_module_str ? \l_stex_current_copymodule_n
3451       }
3452       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1_prop}

```

```

3453 \prop_put:Nnx \l_tmpa_prop { name }{ \l_stex_current_copymodule_name_str / ####1 }
3454 \prop_put:Nnx \l_tmpa_prop { module }{ \l_stex_current_module_str }
3455 \tl_put_right:Nx \l_tmpa_tl {
3456   \prop_set_from_keyval:cn {
3457     l_stex_symdecl\l_stex_current_module_str ? \l_stex_current_copymodule_name_str
3458   }{
3459     \prop_to_keyval:N \l_tmpa_prop
3460   }
3461   \seq_clear:c {
3462     l_stex_symdecl_
3463     \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3464     _notations
3465   }
3466 }
3467 \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \l_stex_current_copymodule
3468 \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3469   \tl_put_right:Nx \l_tmpc_tl {
3470     \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?#
3471   }
3472   \tl_put_right:Nx \l_tmpa_tl {
3473     \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3474       \stex_invoke_symbol:n {
3475         \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3476       }
3477     }
3478   }
3479 }
3480 }
3481 \tl_if_exist:cT {l__stex_features_copymodule_##1?####1_def_tl}{
3482   \tl_put_right:Nx \l_tmpc_tl {
3483     \stex_annotate_invisible:nnn{definien}{\use:c{l__stex_features_copymodule_##1?#
3484   }
3485 }
3486 \tl_put_right:Nx \l_tmpb_tl {
3487   \stex_annotate:nnn{assignment} {##1?####1} { \l_tmpc_tl }
3488 }
3489 }
3490 }
3491 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3492 \tl_put_left:Nx \l_tmpa_tl {
3493   \prop_set_from_keyval:cn {
3494     l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _pro
3495   }{
3496     \prop_to_keyval:N \l_stex_current_copymodule_prop
3497   }
3498 }
3499 \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl
3500 \stex_debug:nn{copymodule}{result:\meaning \l_tmpa_tl}
3501 \exp_args:Nx \stex_do_aftergroup:n {
3502   \exp_args:No \exp_not:n \l_tmpa_tl
3503 }
3504 \l_tmpb_tl
3505 \stex_if_smsmode:F {
3506   \end{stex_annotate_env}

```

```

3507 }
3508 }
3509
3510 \NewDocumentEnvironment {copymodule} { 0{} m m}{
3511   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ structure }
3512   \stex_deactivate_macro:Nn \symdecl {module~environments}
3513   \stex_deactivate_macro:Nn \symdef {module~environments}
3514   \stex_deactivate_macro:Nn \notation {module~environments}
3515   \stex_reactivate_macro:N \assign
3516   \stex_reactivate_macro:N \renamedec1
3517   \stex_reactivate_macro:N \donotcopy
3518   \stex_smsmode_do:
3519 }{
3520   \stex_copymodule_end:n {}
3521 }
3522
3523 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
3524   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ realization }
3525   \stex_deactivate_macro:Nn \symdecl {module~environments}
3526   \stex_deactivate_macro:Nn \symdef {module~environments}
3527   \stex_deactivate_macro:Nn \notation {module~environments}
3528   \stex_reactivate_macro:N \assign
3529   \stex_reactivate_macro:N \renamedec1
3530   \stex_reactivate_macro:N \donotcopy
3531   \stex_smsmode_do:
3532 }{
3533   \stex_copymodule_end:n {
3534     \tl_if_exist:cF {
3535       l__stex_features_copymodule_##1?##2_def_tl
3536     }{
3537       \msg_error:nnxx{stex}{error/interpretmodule/nodedefiniens}{
3538         ##1?##2
3539       }{\l_stex_current_copymodule_name_str}
3540     }
3541   }
3542 }
3543
3544 \NewDocumentCommand \donotcopy { 0{} m}{
3545   \stex_import_module_uri:nn { #1 } { #2 }
3546   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3547   \seq_map_inline:Nn \l_stex_collect_imports_seq {
3548     \seq_remove_all:Nn \l_stex_features_copymodule_modules_seq { ##1 }
3549     \seq_map_inline:cn {c_stex_module_##1_constants}{
3550       \seq_remove_all:Nn \l_stex_features_copymodule_fields_seq { ##1 ? #####1 }
3551       \bool_lazy_any_p:nT {
3552         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_name_str}}
3553         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_macroname_str}}
3554         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_def_tl}}
3555       }{
3556         % TODO throw error
3557       }
3558     }
3559   }
3560 }

```



```

3561 \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
3562 \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_ns_str ?\l_stex_import_name_str }
3563 \prop_put:Nnx \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
3564 }
3565
3566 \NewDocumentCommand \assign { m m }{
3567   \stex_get_symbol_in_copymodule:n {#1}
3568   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
3569   \tl_set:cn {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _def_tl}{#2}
3570 }
3571
3572 \keys_define:nn { stex / renamedec1 } {
3573   name .str_set_x:N = \l_stex_renamedec1_name_str
3574 }
3575 \cs_new_protected:Nn \__stex_features_renamedec1_args:n {
3576   \str_clear:N \l_stex_renamedec1_name_str
3577
3578   \keys_set:nn { stex / renamedec1 } { #1 }
3579 }
3580
3581 \NewDocumentCommand \renamedec1 { O{} m m }{
3582   \__stex_features_renamedec1_args:n { #1 }
3583   \stex_get_symbol_in_copymodule:n {#2}
3584   \stex_debug:nn{renamedec1}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
3585   \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
3586   \str_if_empty:NTF \l_stex_renamedec1_name_str {
3587     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3588       \l_stex_get_symbol_uri_str
3589     } }
3590   } {
3591     \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex_r
3592     \stex_debug:nn{renamedec1}{@~\l_stex_current_module_str ? \l_stex_renamedec1_name_str}
3593     \prop_set_eq:cc {l_stex_symdec1_
3594       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3595       _prop
3596     }{l_stex_symdec1_ \l_stex_get_symbol_uri_str _prop}
3597     \seq_set_eq:cc {l_stex_symdec1_
3598       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3599       _notations
3600     }{l_stex_symdec1_ \l_stex_get_symbol_uri_str _notations}
3601     \prop_put:cnx {l_stex_symdec1_
3602       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3603       _prop
3604     }{ name }{ \l_stex_renamedec1_name_str }
3605     \prop_put:cnx {l_stex_symdec1_
3606       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3607       _prop
3608     }{ module }{ \l_stex_current_module_str }
3609     \exp_args:NNx \seq_put_left:Nn \l__stex_features_copymodule_fields_seq {
3610       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3611     }
3612     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3613       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3614     } }

```

```

3615 }
3616 }
3617 %\NewDocumentCommand \notation_in_copymodules: { 0{} m } {
3618 % \_stex_notation_args:n { #1 }
3619 % \tl_clear:N \l_stex_symdecl_definiens_tl
3620 % \stex_get_symbol_in_copymodule:n { #2 }
3621 % \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
3622 % % todo
3623 %}
3624 \stex_deactivate_macro:Nn \assign {copymodules}
3625 \stex_deactivate_macro:Nn \renamedekl {copymodules}
3626 \stex_deactivate_macro:Nn \donotcopy {copymodules}
3627
3628
3629 \seq_new:N \l_stex_implicit_morphisms_seq
3630 \NewDocumentCommand \implicitmorphism { 0{} m m }{
3631 \stex_import_module_uri:nn { #1 } { #2 }
3632 \stex_debug:nn{implicits}{
3633 Implicit~morphism:~
3634 \l_stex_module_ns_str ? \l__stex_features_name_str
3635 }
3636 \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
3637 \l_stex_module_ns_str ? \l__stex_features_name_str
3638 }{
3639 \msg_error:nnn{stex}{error/conflictingmodules}{
3640 \l_stex_module_ns_str ? \l__stex_features_name_str
3641 }
3642 }
3643
3644 % TODO
3645
3646
3647
3648 \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
3649 \l_stex_module_ns_str ? \l__stex_features_name_str
3650 }
3651 }
3652

```

## 32.2 The feature environment

structural@feature

```

3653
3654 \NewDocumentEnvironment{structural@feature}{ m m m }{
3655 \stex_if_in_module:F {
3656 \msg_set:nnn{stex}{error/nomodule}{
3657 Structural~Feature~has~to~occur~in~a~module:\\
3658 Feature~#2~of~type~#1\\
3659 In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
3660 }
3661 \msg_error:nn{stex}{error/nomodule}
3662 }
3663

```

```

3664 \str_set:Nx \l_stex_module_name_str {
3665   \prop_item:Nn \l_stex_current_module_prop
3666     { name } / #2 - feature
3667 }
3668
3669 \str_set:Nx \l_stex_module_ns_str {
3670   \prop_item:Nn \l_stex_current_module_prop
3671     { ns }
3672 }
3673
3674
3675 \str_clear:N \l_tmpa_str
3676 \seq_clear:N \l_tmpa_seq
3677 \tl_clear:N \l_tmpa_tl
3678 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
3679   origname = #2,
3680   name      = \l_stex_module_name_str ,
3681   ns        = \l_stex_module_ns_str ,
3682   imports   = \exp_not:o { \l_tmpa_seq } ,
3683   constants = \exp_not:o { \l_tmpa_seq } ,
3684   content   = \exp_not:o { \l_tmpa_tl } ,
3685   file      = \exp_not:o { \g_stex_currentfile_seq } ,
3686   lang      = \l_stex_module_lang_str ,
3687   sig       = \l_tmpa_str ,
3688   meta      = \l_tmpa_str ,
3689   feature   = #1 ,
3690 }
3691
3692 \stex_if_smsmode:F {
3693   \begin{stex_annotate_env}{ feature:#1 }{}
3694   \stex_annotate_invisible:nnn{header}{}{ #3 }
3695 }
3696 }{
3697   \str_set:Nx \l_tmpa_str {
3698     c_stex_feature_
3699     \prop_item:Nn \l_stex_current_module_prop { ns } ?
3700     \prop_item:Nn \l_stex_current_module_prop { name }
3701     _prop
3702   }
3703   \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
3704   \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
3705   \stex_if_smsmode:TF {
3706     \exp_args:Nx \stex_add_to_sms:n {
3707       \prop_gset_from_keyval:cn {
3708         c_stex_feature_
3709         \prop_item:Nn \l_stex_current_module_prop { ns } ?
3710         \prop_item:Nn \l_stex_current_module_prop { name }
3711         _prop
3712       } {
3713         origname = #2,
3714         name      = \prop_item:cn { \l_tmpa_str } { name } ,
3715         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
3716         imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
3717         constants = \prop_item:cn { \l_tmpa_str } { constants } ,

```

```

3718         content = \prop_item:cn { \l_tmpa_str } { content } ,
3719         file     = \prop_item:cn { \l_tmpa_str } { file } ,
3720         lang     = \prop_item:cn { \l_tmpa_str } { lang } ,
3721         sig      = \prop_item:cn { \l_tmpa_str } { sig } ,
3722         meta     = \prop_item:cn { \l_tmpa_str } { meta } ,
3723         feature  = \prop_item:cn { \l_tmpa_str } { feature }
3724     }
3725 }
3726 } {
3727     \end{stex_annotate_env}
3728 }
3729 }
3730

```

## 32.3 Features

structure

```

3731
3732 \prop_new:N \l_stex_all_structures_prop
3733
3734 \keys_define:nn { stex / features / structure } {
3735     name .str_set_x:N = \l__stex_features_structure_name_str ,
3736 }
3737
3738 \cs_new_protected:Nn \__stex_features_structure_args:n {
3739     \str_clear:N \l__stex_features_structure_name_str
3740     \keys_set:nn { stex / features / structure } { #1 }
3741 }
3742
3743 %\stex_new_feature:nnnn { structure } { 0{ } m } {
3744 %   \__stex_features_structure_args:n { ##1 }
3745 %   \str_if_empty:NT \l__stex_features_structure_name_str {
3746 %       \str_set:Nx \l__stex_features_structure_name_str { ##2 }
3747 %   }
3748 %} {
3749 %
3750 %}
3751
3752 \NewDocumentEnvironment{mathstructure}{0{ } m }{
3753     \__stex_features_structure_args:n { #1 }
3754     \str_if_empty:NT \l__stex_features_structure_name_str {
3755         \str_set:Nx \l__stex_features_structure_name_str { #2 }
3756     }
3757     \exp_args:Nnnx
3758     \begin{structural@feature}{ structure }
3759         { \l__stex_features_structure_name_str }{}
3760         \seq_clear:N \l_tmpa_seq
3761         \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
3762     \stex_smsmode_do:
3763 }{
3764     \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
3765     \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
3766     \str_set:Nx \l_tmpa_str {

```

```

3767 \prop_item:Nn \l_stex_current_module_prop { ns } ?
3768 \prop_item:Nn \l_stex_current_module_prop { name }
3769 }
3770 \seq_map_inline:Nn \l_tmpa_seq {
3771 \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
3772 }
3773 \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
3774 \exp_args:Nnx
3775 \AddToHookNext { env / mathstructure / after }{
3776 \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}}{
3777 \_stex_term_math_oms:nxxx { \l_tmpa_str }{}{0}{}{
3778 }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
3779 \STEXexport {
3780 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3781 {\prop_item:Nn \l_stex_current_module_prop { origname }}
3782 {\l_tmpa_str}
3783 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3784 {#2}{\l_tmpa_str}
3785 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3786 % \prop_item:Nn \l_stex_current_module_prop { origname },
3787 % \l_tmpa_str
3788 % }
3789 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3790 % #2,\l_tmpa_str
3791 % }
3792 % \tl_set:cx { #2 } {
3793 % \stex_invoke_structure:n { \l_tmpa_str }
3794 }
3795 }
3796
3797 \end{structural@feature}
3798 % \g_stex_last_feature_prop
3799 }

```

\instantiate

```

3800 \seq_new:N \l__stex_features_structure_field_seq
3801 \str_new:N \l__stex_features_structure_field_str
3802 \str_new:N \l__stex_features_structure_def_tl
3803 \prop_new:N \l__stex_features_structure_prop
3804 \NewDocumentCommand \instantiate { m O{} m }{
3805 \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
3806 \prop_set_eq:Nc \l__stex_features_structure_prop {
3807 c_stex_feature_\l_tmpa_str _prop
3808 }
3809 \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
3810 \seq_map_inline:Nn \l__stex_features_structure_field_seq {
3811 \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
3812 \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3813 \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
3814 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
3815 {} \l_tmpa_tl
3816 \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3817 \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
3818 \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl

```

```

3819     \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3820   }{
3821     \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
3822     \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3823     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
3824     \l_tmpa_tl
3825     \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3826       \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
3827       \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3828     }{
3829       \tl_clear:N \l_tmpb_tl
3830     }
3831   }
3832 }{
3833   \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
3834   \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3835     \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
3836     \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
3837     \tl_clear:N \l_tmpa_tl
3838   }{
3839     % TODO throw error
3840   }
3841 }
3842 % \l_tmpa_str: name
3843 % \l_tmpa_tl: definiens
3844 % \l_tmpb_tl: notation
3845 \tl_if_empty:NT \l__stex_features_structure_field_str {
3846   % TODO throw error
3847 }
3848 \str_clear:N \l_tmpb_str
3849
3850 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3851 \seq_map_inline:Nn \l_tmpa_seq {
3852   \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
3853   \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
3854   \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
3855     \seq_map_break:n {
3856       \str_set:Nn \l_tmpb_str { ####1 }
3857     }
3858   }
3859 }
3860 \prop_get:cnN { l_stex_symdecl_ \l_tmpb_str _prop } {args}
3861 \l_tmpb_str
3862
3863 \tl_if_empty:NNTF \l_tmpb_tl {
3864   \tl_if_empty:NF \l_tmpa_tl {
3865     \exp_args:Nx \use:n {
3866       \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fe
3867     }
3868   }
3869 }{
3870   \tl_if_empty:NNTF \l_tmpa_tl {
3871     \exp_args:Nx \use:n {
3872       \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\

```

```

3873     }
3874
3875   }{
3876     \exp_args:Nx \use:n {
3877       \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
3878       \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
3879     }
3880   }
3881 }
3882 % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
3883 % \prop_item:Nn \l_stex_current_module_prop {name} ?
3884 % #3/\l__stex_features_structure_field_str
3885 % \par
3886 % \expandafter\present\csname
3887 %   l_stex_symdecl_
3888 %   \prop_item:Nn \l_stex_current_module_prop {ns} ?
3889 %   \prop_item:Nn \l_stex_current_module_prop {name} ?
3890 %   #3/\l__stex_features_structure_field_str
3891 %   _prop
3892 % \endcsname
3893 }
3894
3895 \tl_clear:N \l__stex_features_structure_def_tl
3896
3897 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3898 \seq_map_inline:Nn \l_tmpa_seq {
3899   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3900   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3901   \exp_args:Nx \use:n {
3902     \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
3903
3904     }
3905   }
3906
3907   \prop_if_exist:cF {
3908     l_stex_symdecl_
3909     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3910     \prop_item:Nn \l_stex_current_module_prop {name} ?
3911     #3/\l_tmpa_str
3912     _prop
3913   }{
3914     \prop_get:cnN { l_stex_symdecl_ ##1 _prop } {args}
3915     \l_tmpb_str
3916     \exp_args:Nx \use:n {
3917       \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
3918     }
3919   }
3920 }
3921
3922 \symdecl*[type={\STEXsymbol{module-type}}{
3923   \_stex_term_math_oms:nnnn {
3924     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3925     \prop_item:Nn \l__stex_features_structure_prop {name}
3926   }-}{0}-{}

```

```

3927   }}]{#3}
3928
3929   % TODO: -> sms file
3930
3931   \tl_set:cx{ #3 }{
3932     \stex_invoke_structure:nnn {
3933       \prop_item:Nn \l_stex_current_module_prop {ns} ?
3934       \prop_item:Nn \l_stex_current_module_prop {name} ? #3
3935     } {
3936       \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3937       \prop_item:Nn \l__stex_features_structure_prop {name}
3938     }
3939   }
3940   \stex_smsmode_do:
3941 }

```

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

\stex\_invoke\_structure:nnn

```

3942 % #1: URI of the instance
3943 % #2: URI of the instantiated module
3944 \cs_new_protected:Nn \stex_invoke_structure:nnn {
3945   \tl_if_empty:nTF{ #3 }{
3946     \prop_set_eq:Nc \l__stex_features_structure_prop {
3947       c_stex_feature_ #2 _prop
3948     }
3949     \tl_clear:N \l_tmpa_tl
3950     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3951     \seq_map_inline:Nn \l_tmpa_seq {
3952       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3953       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3954       \cs_if_exist:cT {
3955         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
3956       }{
3957         \tl_if_empty:NF \l_tmpa_tl {
3958           \tl_put_right:Nn \l_tmpa_tl {,}
3959         }
3960         \tl_put_right:Nx \l_tmpa_tl {
3961           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
3962         }
3963       }
3964     }
3965     \exp_args:No \mathstrut \l_tmpa_tl
3966   }{
3967     \stex_invoke_symbol:n{#1/#3}
3968   }
3969 }

```

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

```

3970 </package>

```



## Chapter 33

# STEX -Statements Implementation

```
3971 <*package>
3972
3973 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3974
3975 <@@=stex_statements>
    Warnings and error messages
3976
\titleemph
3977 \def\titleemph#1{\textbf{#1}}
    (End definition for \titleemph. This function is documented on page ??.)
```

### 33.1 Definitions

```
definiendum
3978 \keys_define:nn {stex / definiendum }{
3979   post      .tl_set:N      = \l__stex_statements_definiendum_post_tl,
3980   root      .str_set_x:N   = \l__stex_statements_definiendum_root_str,
3981   gfa       .str_set_x:N   = \l__stex_statements_definiendum_gfa_str
3982 }
3983 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
3984   \str_clear:N \l__stex_statements_definiendum_root_str
3985   \tl_clear:N \l__stex_statements_definiendum_post_tl
3986   \str_clear:N \l__stex_statements_definiendum_gfa_str
3987   \keys_set:nn { stex / definiendum }{ #1 }
3988 }
3989 \NewDocumentCommand \definiendum { 0{} m m } {
3990   \__stex_statements_definiendum_args:n { #1 }
3991   \stex_get_symbol:n { #2 }
3992   \stex_ref_new_sym_target:n \l__stex_get_symbol_uri_str
3993   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
3994     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
3995       \tl_set:Nn \l_tmpa_tl { #3 }

```

```

3996   } {
3997     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
3998     \tl_set:Nn \l_tmpa_tl {
3999       \l__stex_statements_definiendum_root_str\l__stex_statements_definiendum_post_tl
4000     }
4001   }
4002 } {
4003   \tl_set:Nn \l_tmpa_tl { #3 }
4004 }
4005
4006 % TODO root
4007 \rustex_if:TF {
4008   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
4009 } {
4010   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
4011 }
4012 }
4013 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

**definame**

```

4014
4015 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
4016
4017 \NewDocumentCommand \definame { 0{ } m } {
4018   \__stex_statements_definiendum_args:n { #1 }
4019   % TODO: root
4020   \stex_get_symbol:n { #2 }
4021   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4022   \str_set:Nx \l_tmpa_str {
4023     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4024   }
4025   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4026   \rustex_if:TF {
4027     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4028       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4029     }
4030   } {
4031     \defemph@uri {
4032       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4033     } { \l_stex_get_symbol_uri_str }
4034   }
4035 }
4036 \stex_deactivate_macro:Nn \definame {definition~environments}
4037
4038 \NewDocumentCommand \Definame { 0{ } m } {
4039   \__stex_statements_definiendum_args:n { #1 }
4040   \stex_get_symbol:n { #2 }
4041   \str_set:Nx \l_tmpa_str {
4042     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4043   }
4044   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4045   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str

```

```

4046 \rustex_if:TF {
4047   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4048     \l_tmpa_str\l__stex_statements_definiendum_post_tl
4049   }
4050 } {
4051   \defemph@uri {
4052     \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
4053   } { \l_stex_get_symbol_uri_str }
4054 }
4055 }
4056 \stex_deactivate_macro:Nn \Definame {definition-environments}
4057
4058 \NewDocumentCommand \Symname { 0{ } m }{
4059   \stex_symname_args:n { #1 }
4060   \stex_get_symbol:n { #2 }
4061   \str_set:Nx \l_tmpa_str {
4062     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4063   }
4064   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4065   \let\compemph_uri_prev:\compemph@uri
4066   \let\compemph@uri\symrefemph@uri
4067   \exp_args:NNx \use:nn
4068   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }! [
4069     \exp_after:wN \stex_capitalize:n \l_tmpa_str
4070     \l_stex_symname_post_str
4071   ] }
4072   \let\compemph@uri\compemph_uri_prev:
4073 }

```

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

#### sdefinition

```

4074
4075 \keys_define:nn {stex / sdefinition }{
4076   type      .str_set_x:N = \sdefinitiontype,
4077   id        .str_set_x:N = \sdefinitionid,
4078   name      .str_set_x:N = \sdefinitionname,
4079   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4080   title     .tl_set:N     = \sdefinitiontitle
4081 }
4082 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
4083   \str_clear:N \sdefinitiontype
4084   \str_clear:N \sdefinitionid
4085   \str_clear:N \sdefinitionname
4086   \clist_clear:N \l__stex_statements_sdefinition_for_clist
4087   \tl_clear:N \sdefinitiontitle
4088   \keys_set:nn { stex / sdefinition }{ #1 }
4089 }
4090
4091 \NewDocumentEnvironment{sdefinition}{0{}}{
4092   \__stex_statements_sdefinition_args:n{ #1 }
4093   \stex_reactivate_macro:N \definiendum
4094   \stex_reactivate_macro:N \definame
4095   \stex_reactivate_macro:N \Definame

```

```

4096 \stex_if_smsmode:F{
4097   \seq_clear:N \l_tmpa_seq
4098   \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4099     \str_if_eq:nnF{ ##1 }{}{
4100       \stex_get_symbol:n { ##1 }
4101       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4102         \l_stex_get_symbol_uri_str
4103       }
4104     }
4105   }
4106   \exp_args:Nnnx
4107   \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {,}}
4108   \str_if_empty:NF \sdefinitiontype {
4109     \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{}
4110   }
4111   \clist_set:N \l_tmpa_clist \sdefinitiontype
4112   \tl_clear:N \l_tmpa_tl
4113   \clist_map_inline:Nn \l_tmpa_clist {
4114     \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
4115       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
4116     }
4117   }
4118   \tl_if_empty:NTF \l_tmpa_tl {
4119     \__stex_statements_sdefinition_start:
4120   }{
4121     \l_tmpa_tl
4122   }
4123 }
4124 \stex_ref_new_doc_target:n \sdefinitionid
4125 \stex_smsmode_do:
4126 ){
4127   \str_if_empty:NF \sdefinitionname { \symdecl*{\sdefinitionname} }
4128   \stex_if_smsmode:F {
4129     \clist_set:N \l_tmpa_clist \sdefinitiontype
4130     \tl_clear:N \l_tmpa_tl
4131     \clist_map_inline:Nn \l_tmpa_clist {
4132       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
4133         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
4134       }
4135     }
4136     \tl_if_empty:NTF \l_tmpa_tl {
4137       \__stex_statements_sdefinition_end:
4138     }{
4139       \l_tmpa_tl
4140     }
4141     \end{stex_annotate_env}
4142   }
4143 }

```

\stexpatchdefinition

```

4144 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
4145   \par\noindent\titllemph{Definition\tl_if_empty:NF \sdefinitiontitle {
4146     ~(\sdefinitiontitle)
4147   }~}

```

```

4148 }
4149 \cs_new_protected:Nn \__stex_statements_sdefinition_end: { \par \medskip }
4150
4151 \newcommand\stexpatchdefinition[3] [] {
4152   \str_set:Nx \l_tmpa_str{ #1 }
4153   \str_if_empty:NTF \l_tmpa_str {
4154     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
4155     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
4156   }{
4157     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
4158     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
4159   }
4160 }

```

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

\inlinedef inline:

```

4161 \keys_define:nn {stex / inlinedef }{
4162   type      .str_set_x:N = \sdefinitiontype,
4163   id        .str_set_x:N = \sdefinitionid,
4164   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4165   name      .str_set_x:N = \sdefinitionname
4166 }
4167 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
4168   \str_clear:N \sdefinitiontype
4169   \str_clear:N \sdefinitionid
4170   \str_clear:N \sdefinitionname
4171   \clist_clear:N \l__stex_statements_sdefinition_for_clist
4172   \keys_set:nn { stex / inlinedef }{ #1 }
4173 }
4174 \NewDocumentCommand \inlinedef { 0{} m } {
4175   \begingroup
4176   \__stex_statements_inlinedef_args:n{ #1 }
4177   \stex_ref_new_doc_target:n \sdefinitionid
4178   \stex_reactivate_macro:N \definiendum
4179   \stex_reactivate_macro:N \definame
4180   \stex_reactivate_macro:N \Definame
4181   \stex_if_smsmode:TF{
4182     \str_if_empty:NF \sdefinitionname { \symdecl*{\sdefinitionname} }
4183   }{
4184     \seq_clear:N \l_tmpa_seq
4185     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4186       \str_if_eq:nnF{ ##1 }{ }{
4187         \stex_get_symbol:n { ##1 }
4188         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4189           \l_stex_get_symbol_uri_str
4190         }
4191       }
4192     }
4193     \exp_args:Nnx
4194     \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
4195       \str_if_empty:NF \sdefinitiontype {
4196         \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{ }
4197       }

```

```

4198     #2
4199     \str_if_empty:NF \sdefinitionname { \symdecl*{\sdefinitionname} }
4200   }
4201 }
4202 \endgroup
4203 \stex_smsmode_do:
4204 }

```

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

## 33.2 Assertions

sassertion

```

4205
4206 \keys_define:nn {stex / sassertion }{
4207   type      .str_set_x:N = \sassertiontype,
4208   id        .str_set_x:N = \sassertionid,
4209   title     .tl_set:N    = \sassertiontitle ,
4210   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4211   name      .str_set_x:N = \sassertionname
4212 }
4213 \cs_new_protected:Nn \l__stex_statements_sassertion_args:n {
4214   \str_clear:N \sassertiontype
4215   \str_clear:N \sassertionid
4216   \str_clear:N \sassertionname
4217   \clist_clear:N \l__stex_statements_sassertion_for_clist
4218   \tl_clear:N \sassertiontitle
4219   \keys_set:nn { stex / sassertion }{ #1 }
4220 }
4221
4222 %\tl_new:N \g__stex_statements_aftergroup_tl
4223
4224 \NewDocumentEnvironment{sassertion}{0{}}{
4225   \l__stex_statements_sassertion_args:n{ #1 }
4226   \stex_if_smsmode:F {
4227     \seq_clear:N \l_tmpa_seq
4228     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4229       \str_if_eq:nnF{ ##1 }{}{
4230         \stex_get_symbol:n { ##1 }
4231         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4232           \l_stex_get_symbol_uri_str
4233         }
4234       }
4235     }
4236     \exp_args:Nnnx
4237     \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}
4238     \str_if_empty:NF \sassertiontype {
4239       \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
4240     }
4241     \clist_set:Nn \l_tmpa_clist \sassertiontype
4242     \tl_clear:N \l_tmpa_tl
4243     \clist_map_inline:Nn \l_tmpa_clist {
4244       \tl_if_exist:cT {\l__stex_statements_sassertion_##1_start:}{

```

```

4245         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
4246     }
4247 }
4248 \tl_if_empty:NTF \l_tmpa_tl {
4249     \__stex_statements_sassertion_start:
4250 }{
4251     \l_tmpa_tl
4252 }
4253 }
4254 \stex_ref_new_doc_target:n \sassertionid
4255 \stex_smsmode_do:
4256 }{
4257     \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
4258     \stex_if_smsmode:F {
4259         \clist_set:Nn \l_tmpa_clist \sassertiontype
4260         \tl_clear:N \l_tmpa_tl
4261         \clist_map_inline:Nn \l_tmpa_clist {
4262             \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
4263                 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
4264             }
4265         }
4266         \tl_if_empty:NTF \l_tmpa_tl {
4267             \__stex_statements_sassertion_end:
4268         }{
4269             \l_tmpa_tl
4270         }
4271         \end{stex_annotate_env}
4272     }
4273 }

```

`\stexpatchassertion`

```

4274
4275 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
4276     \par\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
4277         (\sassertiontitle)
4278     }~}
4279 }
4280 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
4281
4282 \newcommand\stexpatchassertion[3] [] {
4283     \str_set:Nx \l_tmpa_str{ #1 }
4284     \str_if_empty:NTF \l_tmpa_str {
4285         \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
4286         \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
4287     }{
4288         \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
4289         \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
4290     }
4291 }

```

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

`\inlineass inline:`

```

4292 \keys_define:nn {stex / inlineass }{

```

```

4293 type      .str_set_x:N = \sassertiontype,
4294 id         .str_set_x:N = \sassertionid,
4295 for        .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4296 name       .str_set_x:N = \sassertionname
4297 }
4298 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
4299   \str_clear:N \sassertiontype
4300   \str_clear:N \sassertionid
4301   \str_clear:N \sassertionname
4302   \clist_clear:N \l__stex_statements_sassertion_for_clist
4303   \keys_set:nn { stex / inlineass }{ #1 }
4304 }
4305 \NewDocumentCommand \inlineass { O{} m } {
4306   \beginngroup
4307     \__stex_statements_inlineass_args:n{ #1 }
4308     \stex_ref_new_doc_target:n \sassertionid
4309     \stex_if_smsmode:TF{
4310       \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
4311     }{
4312       \seq_clear:N \l_tmpa_seq
4313       \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4314         \str_if_eq:nnF{ ##1 }{}{
4315           \stex_get_symbol:n { ##1 }
4316           \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4317             \l_stex_get_symbol_uri_str
4318           }
4319         }
4320       }
4321       \exp_args:Nnx
4322       \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{
4323         \str_if_empty:NF \sassertiontype {
4324           \stex_annotate_invisible:nnn{type}{\sassertiontype}{ }
4325         }
4326         #2
4327         \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
4328       }
4329     }
4330   \endgroup
4331   \stex_smsmode_do:
4332 }

```

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

### 33.3 Examples

`sexample`

```

4333
4334 \keys_define:nn {stex / sexample }{
4335   type      .str_set_x:N = \exampletype,
4336   id        .str_set_x:N = \sexampleid,
4337   title     .tl_set:N     = \sexampletile,
4338   for       .clist_set:N = \l__stex_statements_sexample_for_clist,
4339 }

```



```

4340 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
4341   \str_clear:N \sexamplotype
4342   \str_clear:N \sexampleid
4343   \tl_clear:N \sexampletile
4344   \clist_clear:N \l__stex_statements_sexample_for_clist
4345   \keys_set:nn { stex / sexample }{ #1 }
4346 }
4347
4348 \NewDocumentEnvironment{sexample}{0{}}{
4349   \__stex_statements_sexample_args:n{ #1 }
4350   \stex_if_smsmode:F {
4351     \seq_clear:N \l_tmpa_seq
4352     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4353       \str_if_eq:nnF{ ##1 }{{}{
4354         \stex_get_symbol:n { ##1 }
4355         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4356           \l_stex_get_symbol_uri_str
4357         }
4358       }
4359     }
4360     \exp_args:Nnnx
4361     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
4362     \str_if_empty:NF \sexamplotype {
4363       \stex_annotate_invisible:nnn{type}{\sexamplotype}{}
4364     }
4365     \clist_set:No \l_tmpa_clist \sexamplotype
4366     \tl_clear:N \l_tmpa_tl
4367     \clist_map_inline:Nn \l_tmpa_clist {
4368       \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
4369         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
4370       }
4371     }
4372     \tl_if_empty:NTF \l_tmpa_tl {
4373       \__stex_statements_sexample_start:
4374     }{
4375       \l_tmpa_tl
4376     }
4377   }
4378   \stex_ref_new_doc_target:n \sexampleid
4379   \stex_smsmode_do:
4380 }{
4381   \str_if_empty:NF \sexamplename { \symdecl*{\sexamplename} }
4382   \stex_if_smsmode:F {
4383     \clist_set:No \l_tmpa_clist \sexamplotype
4384     \tl_clear:N \l_tmpa_tl
4385     \clist_map_inline:Nn \l_tmpa_clist {
4386       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
4387         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
4388       }
4389     }
4390     \tl_if_empty:NTF \l_tmpa_tl {
4391       \__stex_statements_sexample_end:
4392     }{
4393       \l_tmpa_tl

```

```

4394     }
4395     \end{stex_annotate_env}
4396   }
4397 }

```

`\stexpatchexample`

```

4398
4399 \cs_new_protected:Nn \__stex_statements_sexample_start: {
4400   \par\noindent\titllemph{Example~\tl_if_empty:NF \sexampltitle {
4401     (\sexampltitle)
4402   }~}
4403 }
4404 \cs_new_protected:Nn \__stex_statements_sexample_end: { \par\medskip}
4405
4406 \newcommand\stexpatchexample[3][] {
4407   \str_set:Nx \l_tmpa_str{ #1 }
4408   \str_if_empty:NTF \l_tmpa_str {
4409     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
4410     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
4411   }{
4412     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
4413     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
4414   }
4415 }

```

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

`\inlineex` inline:

```

4416 \keys_define:nn {stex / inlineex }{
4417   type      .str_set_x:N = \sexampltype,
4418   id        .str_set_x:N = \sexampleid,
4419   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
4420   name      .str_set_x:N = \sexamplname
4421 }
4422 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
4423   \str_clear:N \sexampltype
4424   \str_clear:N \sexampleid
4425   \str_clear:N \sexamplname
4426   \clist_clear:N \l__stex_statements_sexample_for_clist
4427   \keys_set:nn { stex / inlineex }{ #1 }
4428 }
4429 \NewDocumentCommand \inlineex { 0{ } m } {
4430   \begingroup
4431   \__stex_statements_inlineex_args:n{ #1 }
4432   \stex_ref_new_doc_target:n \sexampleid
4433   \stex_if_smsmode:TF{
4434     \str_if_empty:NF \sexamplname { \symdecl*{\examplename} }
4435   }{
4436     \seq_clear:N \l_tmpa_seq
4437     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4438       \str_if_eq:nnF{ ##1 }{}{
4439         \stex_get_symbol:n { ##1 }
4440         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4441           \l_stex_get_symbol_uri_str

```

```

4442     }
4443   }
4444 }
4445 \exp_args:Nnx
4446 \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpa_seq {,}}{
4447   \str_if_empty:NF \sexamplotype {
4448     \stex_annotate_invisible:nnn{type}{\sexamplotype}{ }
4449   }
4450   #2
4451   \str_if_empty:NF \sexamplename { \symdecl*{\sexamplename} }
4452 }
4453 }
4454 \endgroup
4455 \stex_smsmode_do:
4456 }

```

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

## 33.4 Logical Paragraphs

`sparagraph`

```

4457 \keys_define:nn { stex / sparagraph } {
4458   id      .str_set:N = \sparagraphid ,
4459   title   .tl_set:N  = \l_stex_sparagraph_title_tl ,
4460   type     .str_set:N = \sparagraphtype ,
4461   for      .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
4462   from     .tl_set:N  = \sparagraphfrom ,
4463   to       .tl_set:N  = \sparagraphto ,
4464   start    .tl_set:N  = \l_stex_sparagraph_start_tl ,
4465   name     .str_set:N  = \sparagraphname
4466 }
4467
4468 \cs_new_protected:Nn \stex_sparagraph_args:n {
4469   \tl_clear:N \l_stex_sparagraph_title_tl
4470   \tl_clear:N \sparagraphfrom
4471   \tl_clear:N \sparagraphto
4472   \tl_clear:N \l_stex_sparagraph_start_tl
4473   \str_clear:N \sparagraphid
4474   \str_clear:N \sparagraphtype
4475   \clist_clear:N \l__stex_statements_sparagraph_for_clist
4476   \str_clear:N \sparagraphname
4477   \keys_set:nn { stex / sparagraph } { #1 }
4478 }
4479 \newif\if@in@omtext\@in@omtextfalse
4480
4481 \NewDocumentEnvironment {sparagraph} { 0{} } {
4482   \stex_sparagraph_args:n { #1 }
4483   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4484     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
4485   }{
4486     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
4487   }
4488   \@in@omtexttrue

```

```

4489 \stex_if_smsmode:F {
4490   \seq_clear:N \l_tmpa_seq
4491   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
4492     \str_if_eq:nnF{ ##1 }{}{
4493       \stex_get_symbol:n { ##1 }
4494       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4495         \l_stex_get_symbol_uri_str
4496       }
4497     }
4498   }
4499   \exp_args:Nnnx
4500   \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
4501   \str_if_empty:NF \sparagraphtype {
4502     \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}
4503   }
4504   \str_if_empty:NF \sparagraphfrom {
4505     \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
4506   }
4507   \str_if_empty:NF \sparagraphto {
4508     \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
4509   }
4510   \clist_set:No \l_tmpa_clist \sparagraphtype
4511   \tl_clear:N \l_tmpa_tl
4512   \clist_map_inline:Nn \sparagraphtype {
4513     \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
4514       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
4515     }
4516   }
4517   \tl_if_empty:NTF \l_tmpa_tl {
4518     \__stex_statements_sparagraph_start:
4519   }{
4520     \l_tmpa_tl
4521   }
4522 }
4523 \stex_ref_new_doc_target:n \sparagraphid
4524 \stex_smsmode_do:
4525 \ignorespacesandpars
4526 }{
4527   \stex_if_smsmode:F {
4528     \clist_set:No \l_tmpa_clist \sparagraphtype
4529     \tl_clear:N \l_tmpa_tl
4530     \clist_map_inline:Nn \l_tmpa_clist {
4531       \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
4532         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
4533       }
4534     }
4535     \str_if_empty:NF \sparagraphname { \symdecl*{\sparagraphname} }
4536     \tl_if_empty:NTF \l_tmpa_tl {
4537       \__stex_statements_sparagraph_end:
4538     }{
4539       \l_tmpa_tl
4540     }
4541     \end{stex_annotate_env}
4542   }

```

```

4543 }

\stexpatchparagraph

4544
4545 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
4546   \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4547     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
4548       \titleemph{\l_stex_sparagraph_title_tl}:~
4549     }
4550   }{
4551     \titleemph{\l_stex_sparagraph_start_tl}~
4552   }
4553 }
4554 \cs_new_protected:Nn \__stex_statements_sparagraph_end: { \par\medskip}
4555
4556 \newcommand\stexpatchparagraph[3] [] {
4557   \str_set:Nx \l_tmpa_str{ #1 }
4558   \str_if_empty:NTF \l_tmpa_str {
4559     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
4560     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
4561   }{
4562     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
4563     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
4564   }
4565 }
4566
4567 \keys_define:nn { stex / inlinepara} {
4568   id      .str_set_x:N = \sparagraphid ,
4569   type    .str_set_x:N = \sparagraphtype ,
4570   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
4571   from    .tl_set:N    = \sparagraphfrom ,
4572   to      .tl_set:N    = \sparagraphto ,
4573   name    .str_set:N   = \sparagraphname
4574 }
4575 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
4576   \tl_clear:N \sparagraphfrom
4577   \tl_clear:N \sparagraphto
4578   \str_clear:N \sparagraphid
4579   \str_clear:N \sparagraphtype
4580   \clist_clear:N \l__stex_statements_sparagraph_for_clist
4581   \str_clear:N \sparagraphname
4582   \keys_set:nn { stex / inlinepara }{ #1 }
4583 }
4584 \NewDocumentCommand \inlinepara { 0{} m } {
4585   \begingroup
4586   \__stex_statements_inlinepara_args:n{ #1 }
4587   \stex_ref_new_doc_target:n \sparagraphid
4588   \stex_if_smsmode:TF{
4589     \str_if_empty:NF \sparagraphname { \symdecl*\sparagraphname} }
4590   }{
4591     \seq_clear:N \l_tmpa_seq
4592     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
4593       \str_if_eq:nnF{ ##1 }{ }{
4594         \stex_get_symbol:n { ##1 }

```

```

4595     \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4596         \l_stex_get_symbol_uri_str
4597     }
4598 }
4599 }
4600 \exp_args:Nnx
4601 \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
4602     \str_if_empty:NF \sparagraphtype {
4603         \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}}
4604     }
4605     \str_if_empty:NF \sparagraphfrom {
4606         \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}}
4607     }
4608     \str_if_empty:NF \sparagraphto {
4609         \stex_annotate_invisible:nnn{to}{\sparagraphto}{}}
4610     }
4611     #2
4612     \str_if_empty:NF \sparagraphname { \symdecl*{\sparagraphname} }
4613 }
4614 }
4615 \endgroup
4616 \stex_smsmode_do:
4617 }
4618

```

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

symboldoc

```

4619 \NewDocumentEnvironment{symboldoc}{m}{
4620     \seq_set_split:Nnn \l_tmpa_seq , { #1 }
4621     \seq_clear:N \l_tmpb_seq
4622     \seq_map_inline:Nn \l_tmpa_seq {
4623         \str_if_eq:nnF{ ##1 }{}{
4624             \stex_get_symbol:n { ##1 }
4625             \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
4626                 \l_stex_get_symbol_uri_str
4627             }
4628         }
4629     }
4630     \par
4631     \exp_args:Nnnx
4632     \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
4633 }{
4634     \end{stex_annotate_env}
4635 }
4636 \</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>

```
4637 \package
4638 \@@=stex_sproof
4639
4640 %%%%%%%%%%%%%%%%% sproof.dtx %%%%%%%%%%%%%%%%%
4641
```

### 34.2 Proofs

We first define some keys for the proof environment.

```
4642 \keys_define:nn { stex / spf } {
4643   id          .str_set:N = \l__stex_sproof_spf_id_str,
4644   display     .tl_set:N  = \l__stex_sproof_spf_display_tl,
4645   for         .tl_set:N  = \l__stex_sproof_spf_for_tl ,
4646   from        .tl_set:N  = \l__stex_sproof_spf_from_tl ,
4647   proofend    .tl_set:N  = \l__stex_sproof_spf_proofend_tl,
4648   type        .tl_set:N  = \l__stex_sproof_spf_type_tl,
4649   title       .tl_set:N  = \l__stex_sproof_spf_title_tl,
4650   continues   .tl_set:N  = \l__stex_sproof_spf_continues_tl,
4651   functions   .tl_set:N  = \l__stex_sproof_spf_functions_tl,
4652   method      .tl_set:N  = \l__stex_sproof_spf_method_tl
4653 }
4654 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
4655   \str_clear:N \l__stex_sproof_spf_id_str
4656   \tl_clear:N \l__stex_sproof_spf_display_tl
4657   \tl_clear:N \l__stex_sproof_spf_for_tl
4658   \tl_clear:N \l__stex_sproof_spf_from_tl
4659   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
4660   \tl_clear:N \l__stex_sproof_spf_type_tl
4661   \tl_clear:N \l__stex_sproof_spf_title_tl
4662 }
```

---

<sup>13</sup>EDNOTE: need an implementation for L<sup>A</sup>T<sub>E</sub>X<sub>M</sub>L

```

4662 \tl_clear:N \l__stex_sproof_spf_continues_tl
4663 \tl_clear:N \l__stex_sproof_spf_functions_tl
4664 \tl_clear:N \l__stex_sproof_spf_method_tl
4665 \keys_set:nn { stex / spf }{ #1 }
4666 }

```

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

```

4667 \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.

```

4668 \newcount\count_ten
4669 \newenvironment{pst@with@label}[1]{
4670   \edef\pst@label{#1}
4671   \advance\count_ten by 1\relax
4672   \count_ten=1
4673 }{
4674   \advance\count_ten by -1\relax
4675 }

```

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

```

4676 \def\the@pst@label{
4677   \pst@make@label\pst@label{\number\count_ten}\l__stex_sproof_pstlabel_postfix_tl
4678 }

```

(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.

```

4679 \keys_define:nn { stex / pstlabel }{
4680   prefix      .tl_set:N   = \l__stex_sproof_pstlabel_prefix_tl,
4681   delimiter   .tl_set:N   = \l__stex_sproof_pstlabel_delimiter_tl,
4682   postfix     .tl_set:N   = \l__stex_sproof_pstlabel_postfix_tl
4683 }
4684 \cs_new_protected:Nn \__stex_sproof_pstlabel_args:n {

```

---

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



```

4685 \tl_set:Nn \l__stex_sproof_pstlabel_prefix_tl {P}
4686 \tl_set:Nn \l__stex_sproof_pstlabel_delimiter_tl {.}
4687 \tl_clear:N \l__stex_sproof_pstlabel_postfix_tl
4688 }
4689 \__stex_sproof_pstlabel_args:n {}
4690 \newcommand\setpstlabelstyle[1]{
4691   \__stex_sproof_pstlabel_args:n {#1}
4692 }
4693 \newcommand\setpstlabelstyledefault{%
4694   \__stex_sproof_pstlabel_args:n{prefix=P,delimiter=.,postfix={}}
4695 }

```

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

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

```

4696 \ExplSyntaxOff
4697 \def\pst@make@label@long#1#2{\@for\@I:=#1\do{\expandafter\expandafter\expandafter\@I\csname
4698 \def\pst@make@label@angles#1#2{\ensuremath{\@for\@I:=#1\do{\rangle}}#2}
4699 \def\pst@make@label@short#1#2{#2}
4700 \def\pst@make@label@empty#1#2{}
4701 \ExplSyntaxOn
4702 \def\pstlabelstyle#1{%
4703   \def\pst@make@label{\use:c{pst@make@label@#1}}%
4704 }%
4705 \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.

```

4706 \def\next@pst@label{%
4707   \global\advance\count\count10 by 1%
4708 }%

```

(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

```

4709 \def\sproof@box{
4710   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
4711 }
4712 \def\spf@proofend{\sproof@box}
4713 \def\sproofend{
4714   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
4715     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
4716   }
4717 }
4718 \def\sProofEndSymbol#1{\def\sproof@box{#1}}

```

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

**spf@\*@kw**

```

4719 \def\spf@proofsketch@kw{Proof Sketch}
4720 \def\spf@proof@kw{Proof}
4721 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

4722 \AddToHook{begindocument}{
4723   \ltx@ifpackageloaded{babel}{
4724     \makeatletter
4725     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4726     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4727       \input{sproof-ngerman.ldf}
4728     }
4729     \clist_if_in:NnT \l_tmpa_clist {finnish}{
4730       \input{sproof-finnish.ldf}
4731     }
4732     \clist_if_in:NnT \l_tmpa_clist {french}{
4733       \input{sproof-french.ldf}
4734     }
4735     \clist_if_in:NnT \l_tmpa_clist {russian}{
4736       \input{sproof-russian.ldf}
4737     }
4738     \makeatother
4739   }{}
4740 }

```

`spfsketch`

```

4741 \newcommand\spfsketch[2][]{
4742   \__stex_sproof_spf_args:n{#1}
4743   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4744     \titleemph{
4745       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
4746         \spf@proofsketch@kw
4747       }{
4748         \l__stex_sproof_spf_type_tl
4749       }
4750     }:
4751   }
4752   {~#2}
4753   %\sref@label@id{this \ifx\spf@type\@empty\spf@proofsketch@kw\else\spf@type\fi}
4754   \sproofend
4755 }

```

(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>

```

4756 \newenvironment{spfeq}[2][]{
4757   \__stex_sproof_spf_args:n{#1}
4758   %\sref@target
4759   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4760     \titleemph{
4761       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
4762         \spf@proof@kw
4763       }{

```

<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

```

4764         \l__stex_sproof_spf_type_tl
4765     }
4766     }:
4767 }
4768 {-#2}
4769 \begin{displaymath}\begin{array}{rcll}
4770 }{
4771 \end{array}\end{displaymath}
4772 }

```

(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.

```

4773 \newenvironment{spf@proof}[2] []{
4774   \l__stex_sproof_spf_args:n{#1}
4775   %\sref@target
4776   \count_ten=10
4777   \par\noindent
4778   \tl_if_eq:NNTF \l__stex_sproof_spf_display_tl\spf@flow{
4779     \titleemph{
4780       \tl_if_empty:NNTF \l__stex_sproof_spf_type_tl {
4781         \spf@proof@kw
4782       }{
4783         \l__stex_sproof_spf_type_tl
4784       }
4785     }:
4786   }
4787   {-#2}
4788   %\sref@label{id{this \ifx\spf@type\empty\spf@proof@kw\else\spf@type\fi}
4789   \def\pst@label{}
4790   \newcount\pst@count% initialize the labeling mechanism
4791   \begin{description}\begin{pst@with@label}{\l__stex_sproof_pstlabel_prefix_tl}
4792   }{
4793     \end{pst@with@label}\end{description}
4794   }
4795   \newenvironment{sproof}[2] []{\begin{spf@proof}[#1]{#2}}{\sproofend\end{spf@proof}}
4796   \newenvironment{sProof}[2] []{\begin{spf@proof}[#1]{#2}}{\end{spf@proof}}

```

**\spfidea**

```

4797 \newcommand\spfidea[2] []{
4798   \l__stex_sproof_spf_args:n{#1}
4799   \titleemph{
4800     \tl_if_empty:NNTF \l__stex_sproof_spf_type_tl {Proof~Idea}{
4801       \l__stex_sproof_spf_type_tl
4802     }:
4803   }-#2
4804   \sproofend
4805 }

```

(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
4806 \newenvironment{spfstep}[1][]{
4807   \_stex_sproof_spf_args:n{#1}
4808   \@in@omtexttrue
4809   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4810     \item[\the@pst@label]
4811   }
4812   \tl_if_empty:NF \l__stex_sproof_spf_title_tl {
4813     {(\titleemph{\l__stex_sproof_spf_title_tl})\enspace}
4814   }
4815   %\sref@label{id{\pst@label}}
4816   \ignorespacesandpars
4817 }{
4818   \next@pst@label\ignorespacesandpars
4819 }

```

**sproofcomment**

```

4820 \newenvironment{sproofcomment}[1][]{
4821   \_stex_sproof_spf_args:n{#1}
4822   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4823     \item[\the@pst@label]
4824   }
4825 }{
4826   \next@pst@label
4827 }

```

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.

```

4828 \newenvironment{subproof}[2][]{
4829   \_stex_sproof_spf_args:n{#1}
4830   \def\@test{#2}
4831   \ifx\@test\empty\else
4832     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4833       \item[\the@pst@label]
4834     }{#2}
4835   \fi
4836   \begin{pst@with@label}{\pst@label,\number\count_ten}
4837 }{
4838   \end{pst@with@label}\next@pst@label
4839 }

```

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

```

4840 \newenvironment{spfcases}[2][]{
4841   \def\@test{#1}
4842   \ifx\@test\empty
4843     \begin{subproof}[method=by-cases]{#2}

```

---

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

```

4844 \else
4845   \begin{subproof}[#1,method=by-cases]{#2}
4846 \fi
4847 }{
4848   \end{subproof}
4849 }

```

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

```

4850 \newenvironment{spfcase}[2] [] {
4851   \__stex_sproof_spf_args:n{#1}
4852   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4853     \item[\the@pst@label]
4854   }
4855   \def\@test{#2}
4856   \ifx\@test\@empty
4857   \else
4858     {\titleemph{#2}:~}
4859   \fi
4860   \begin{pst@with@label}{\pst@label,\number\count_ten}
4861 }{
4862   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4863     \sproofend
4864   }
4865   \end{pst@with@label}
4866   \next@pst@label
4867 }

```

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

```

4868 \newcommand\spfcasesketch[3] [] {
4869   \__stex_sproof_spf_args:n{#1}
4870   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4871     \item[\the@pst@label]
4872   }
4873   \def\@test{#2}
4874   \ifx\@test\@empty
4875   \else
4876     {\titleemph{#2}:~}
4877   \fi#3
4878   \next@pst@label
4879 }%

```

### 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.

```

4880 \keys_define:nn { stex / just }{
4881   id      .str_set:x:N = \l__stex_sproof_just_id_str,
4882   method  .tl_set:N    = \l__stex_sproof_just_method_tl,
4883   premises .tl_set:N    = \l__stex_sproof_just_premises_tl,
4884   args    .tl_set:N    = \l__stex_sproof_just_args_tl
4885 }

```

EdN:17

The next three environments and macros are purely semantic, so we ignore the keyval arguments for now and only display the content.<sup>17</sup>

`justification`

```
4886 \newenvironment{justification}[1] [] {}{}
```

`\premise`

```
4887 \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.

```
4888 \newcommand\justarg[2] [] {#2}
```

```
4889 \end{package}
```

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

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

---

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

## Chapter 35

# STEX -Others Implementation

```
4890 <*package>
4891
4892 %%%%%%%%%% others.dtx %%%%%%%%%%
4893
4894 <@@=stex_others>
      Warnings and error messages
4895 % None

\MSC Math subject classifier

4896 \NewDocumentCommand \MSC {m} {
4897   % TODO
4898 }

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

4899 \@ifpackageloaded{tikzinput}{
4900   \RequirePackage{stex-tikzinput}
4901 }{}
4902 </package>
```

## Chapter 36

# STEX -Metatheory Implementation

```
4903 \*package>
4904 \@@=stex_modules>
4905
4906 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
4907
4908 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
4909 \begingroup
4910 \stex_module_setup:nn{
4911   ns=\c_stex_metatheory_ns_str,
4912   meta=NONE
4913 }{Metatheory}
4914 \stex_reactivate_macro:N \symdecl
4915 \stex_reactivate_macro:N \notation
4916 \stex_reactivate_macro:N \symdef
4917 \ExplSyntaxOff
4918 \csname stex_suppress_html:n\endcsname{
4919   % is-a (a:A, a \in A, a is an A, etc.)
4920   \symdecl[args=ai]{isa}
4921   \notation[typed]{isa}{#1 \comp{:} #2}{##1 \comp, ##2}
4922   \notation[in]{isa}{#1 \comp\in #2}{##1 \comp, ##2}
4923   \notation[pred]{isa}{#2\comp(#1 \comp)}{##1 \comp, ##2}
4924
4925   % bind (\forall, \Pi, \lambda etc.)
4926   \symdecl[args=Bi]{bind}
4927   \notation[forall]{bind}{\comp\forall #1.\;#2}{##1 \comp, ##2}
4928   \notation[\Pi]{bind}{\comp\prod_{#1}#2}{##1 \comp, ##2}
4929   \notation[deffun]{bind}{\comp( #1 \comp{ }\;\to\; ) #2}{##1 \comp, ##2}
4930
4931   % dummy variable
4932   \symdecl{dummyvar}
4933   \notation[underscore]{dummyvar}{\comp\_}
4934   \notation[dot]{dummyvar}{\comp\cdot}
4935   \notation[dash]{dummyvar}{\comp{\rm --}}
4936
4937   %fromto (function space, Hom-set, implication etc.)
```



```

4938 \symdecl[args=ai]{fromto}
4939 \notation[xarrow]{fromto}{#1 \comp\to #2}{##1 \comp\times ##2}
4940 \notation[arrow]{fromto}{#1 \comp\to #2}{##1 \comp\to ##2}
4941
4942 % mapto (lambda etc.)
4943 \%symdecl[args=Bi]{mapto}
4944 \%notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
4945 \%notation[lambda]{mapto}{\comp\lambda #1 \comp.; #2}{#1 \comp, #2}
4946 \%notation[lambdau]{mapto}{\comp\lambda_{#1} \comp.; #2}{#1 \comp, #2}
4947
4948 % function/operator application
4949 \symdecl[args=ia]{apply}
4950 \notation[prec=0;0x\infpres,parens]{apply}{#1 \comp( #2 \comp)}{##1 \comp, ##2}
4951 \notation[prec=0;0x\infpres,lambda]{apply}{#1 \; #2 }{##1 \; ; ##2}
4952
4953 % ‘type’ of all collections (sets, classes, types, kinds)
4954 \symdecl{collection}
4955 \notation[U]{collection}{\comp{\mathcal{U}}}
4956 \notation[set]{collection}{\comp{\textsf{Set}}}
4957
4958 % sequences
4959 \symdecl[args=1]{seqtype}
4960 \notation[kleene]{seqtype}{#1^{\comp\ast}}
4961
4962 \symdef[args=2,li,prec=nobrackets]{sequence-index}{#1}_{#2}
4963 \notation[ui,prec=nobrackets]{sequence-index}{#1}^{#2}
4964
4965 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{,\ellipses}}{##1\comp,##2}
4966 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{,\ellipses},#2}{##1\comp,##2}
4967 \symdef[args=aia,prec=nobrackets]{aseqfromtovia}{#1\comp{,\ellipses},#2\comp{,\ellipses},#3}
4968
4969 % letin (‘let’, local definitions, variable substitution)
4970 \symdecl[args=bii]{letin}
4971 \notation[let]{letin}{\comp{\rm let}}{#1\comp{=}#2;\comp{\rm in}}{#3}
4972 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
4973 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
4974
4975 % structures
4976 \symdecl*[args=1]{module-type}
4977 \notation{module-type}{\mathtt{MOD} #1}
4978 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
4979 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{##1 \comp, ##2}
4980
4981 }
4982 \ExplSyntaxOn
4983 \stex_add_to_current_module:n{
4984   \let\appa\apply
4985   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
4986   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
4987   \def\livar{\csname sequence-index\endcsname[li]}
4988   \def\uivar{\csname sequence-index\endcsname[ui]}
4989   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
4990   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
4991   \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}

```

```
4992 }  
4993 \_stex_modules_end_module:  
4994 \endgroup  
4995 </package>
```

## Chapter 37

# Tikzinput Implementation

```
4996 <*package>
4997
4998 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
4999
5000 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
5001 \RequirePackage{l3keys2e}
5002
5003 \keys_define:nn { tikzinput } {
5004   image .bool_set:N = \c_tikzinput_image_bool,
5005   image .default:n = false ,
5006   unknown .code:n = {}
5007 }
5008
5009 \ProcessKeysOptions { tikzinput }
5010
5011 \bool_if:NTF \c_tikzinput_image_bool {
5012   \RequirePackage{graphicx}
5013
5014   \providecommand\usetikzlibrary[]{}
5015   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
5016 }{
5017   \RequirePackage{tikz}
5018   \RequirePackage{standalone}
5019
5020   \newcommand \tikzinput [2] [] {
5021     \setkeys{Gin}{#1}
5022     \ifx \Gin@ewidth \Gin@exclamation
5023       \ifx \Gin@eheight \Gin@exclamation
5024         \input { #2 }
5025       \else
5026         \resizebox{!}{ \Gin@eheight }{
5027           \input { #2 }
5028         }
5029       \fi
5030     \else
5031       \ifx \Gin@eheight \Gin@exclamation
5032         \resizebox{ \Gin@ewidth }{!}{
5033           \input { #2 }
```

```

5034     }
5035     \else
5036         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
5037             \input { #2 }
5038         }
5039     \fi
5040 \fi
5041 }
5042 }
5043
5044 \newcommand \ctikzinput [2] [] {
5045     \begin{center}
5046         \tikzinput [ #1 ] { #2 }
5047     \end{center}
5048 }
5049
5050 \@ifpackageloaded{stex}{
5051     \RequirePackage{stex-tikzinput}
5052 }{}
5053
5054 </package>
5055 <*stex>
5056 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
5057 \RequirePackage{stex}
5058 \RequirePackage{tikzinput}
5059
5060 \newcommand\mhtikzinput [2] [] {%
5061     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
5062     \stex_in_repository:nn\Gin@mhrepos{
5063         \tikzinput [ #1 ] {\mhp{##1}{#2}}
5064     }
5065 }
5066 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [ #1 ] { #2 }\end{center}}
5067 </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.

```
5068 \*cls)
5069 \<@@=document_structure>
5070 \ProvidesExplClass{document-structure}{2022/02/10}{3.0}{Modular Document Structure Class}
5071 \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`

```
5072 \keys_define:nn{ document-structure / pkg }{
5073   class      .str_set_x:N = \c_document_structure_class_str,
5074   minimal    .bool_set:N = \c_document_structure_minimal_bool,
5075   report     .code:n      = {
5076     \ClassWarning{document-structure}{the option 'report' is deprecated, use 'class=report',
5077     \str_set:Nn \c_document_structure_class_str {report}
5078   },
5079   book       .code:n      = {
5080     \ClassWarning{document-structure}{the option 'book' is deprecated, use 'class=book', ins
5081     \str_set:Nn \c_document_structure_class_str {book}
5082   },
5083   bookpart   .code:n      = {
5084     \ClassWarning{document-structure}{the option 'bookpart' is deprecated, use 'class=book,t
5085     \str_set:Nn \c_document_structure_class_str {book}
5086     \str_set:Nn \c_document_structure_topsect_str {chapter}
5087   },
```

```

5088 docopt      .str_set_x:N = \c_document_structure_docopt_str,
5089 unknown     .code:n      = {
5090   \PassOptionsToPackage{ \CurrentOption }{ document-structure }
5091 }
5092 }
5093 \ProcessKeysOptions{ document-structure / pkg }
5094 \str_if_empty:NT \c_document_structure_class_str {
5095   \str_set:Nn \c_document_structure_class_str {article}
5096 }
5097 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
5098   {\c_document_structure_class_str}
5099

```

### 38.3 Beefing up the document environment

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

```

5100 \RequirePackage{document-structure}
5101 \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>

```

5102 \keys_define:nn { document-structure / document }{
5103   id .str_set_x:N = \c_document_structure_document_id_str
5104 }
5105 \let\__document_structure_orig_document=\document
5106 \renewcommand{\document}[1][]{
5107   \keys_set:nn{ document-structure / document }{ #1 }
5108   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
5109   \__document_structure_orig_document
5110 }

```

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

```

5111 }
5112 \</cls>

```

### 38.4 Implementation: document-structure Package

```

5113 \<*package>
5114 \ProvidesExplPackage{document-structure}{2022/02/10}{3.0}{Modular Document Structure}
5115 \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

```

5116
5117 \keys_define:nn{ document-structure / pkg }{
5118   class      .str_set_x:N = \c_document_structure_class_str,
5119   topsect     .str_set_x:N = \c_document_structure_topsect_str,
5120   % showignores .bool_set:N = \c_document_structure_showignores_bool,
5121 }
5122 \ProcessKeysOptions{ document-structure / pkg }
5123 \str_if_empty:NT \c_document_structure_class_str {
5124   \str_set:Nn \c_document_structure_class_str {article}
5125 }
5126 \str_if_empty:NT \c_document_structure_topsect_str {
5127   \str_set:Nn \c_document_structure_topsect_str {section}
5128 }

```

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

```

5129 \RequirePackage{xspace}
5130 \RequirePackage{comment}
5131 \AddToHook{begindocument}{
5132   \ltx@ifpackageloaded{babel}{
5133     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5134     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5135       \makeatletter\input{omdoc-ngerman.ldf}\makeatother
5136     }
5137   }{}
5138 }

```

`\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.

```

5139 \int_new:N \l_document_structure_section_level_int
5140 \str_case:VnF \c_document_structure_topsect_str {
5141   {part}}{
5142     \int_set:Nn \l_document_structure_section_level_int {0}
5143   }
5144   {chapter}{
5145     \int_set:Nn \l_document_structure_section_level_int {1}
5146   }
5147 }{
5148   \str_case:VnF \c_document_structure_class_str {
5149     {book}}{
5150       \int_set:Nn \l_document_structure_section_level_int {0}
5151     }
5152     {report}}{
5153       \int_set:Nn \l_document_structure_section_level_int {0}
5154     }
5155   }{
5156     \int_set:Nn \l_document_structure_section_level_int {2}
5157   }
5158 }

```

## 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

```
5159 \def\current@section@level{document}%
5160 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
5161 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
5162 \cs_new_protected:Npn \skipomgroup {
5163   \ifcase\l_document_structure_section_level_int
5164   \or\stepcounter{part}
5165   \or\stepcounter{chapter}
5166   \or\stepcounter{section}
5167   \or\stepcounter{subsection}
5168   \or\stepcounter{subsubsection}
5169   \or\stepcounter{paragraph}
5170   \or\stepcounter{subparagraph}
5171   \fi
5172 }
```

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

`blindomgroup`

```
5173 \newcommand\at@begin@blindomgroup[1]{%
5174 \newenvironment{blindomgroup}
5175 {
5176   \int_incr:N\l_document_structure_section_level_int
5177   \at@begin@blindomgroup\l_document_structure_section_level_int
5178 }{}}
```

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

```
5179 \newcommand\omgroup@nonum[2]{%
5180   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
5181   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
5182 }
```

*(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.

```
5183 \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.



```

5184 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
5185   \@nameuse{#1}{#2}
5186 }{
5187   \cs_if_exist:NTF\rdfmata@sectioning{
5188     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
5189   }{
5190     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
5191   }
5192 }
5193 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
5194 }

```

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

omgroup

```

5195 \keys_define:nn { document-structure / omgroupp }{
5196   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
5197   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
5198   creators     .clist_set:N = \l__document_structure_omgroup_creators_clist,
5199   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
5200   srccite      .tl_set:N   = \l__document_structure_omgroup_srccite_tl,
5201   type         .tl_set:N   = \l__document_structure_omgroup_type_tl,
5202   short        .tl_set:N   = \l__document_structure_omgroup_short_tl,
5203   display      .tl_set:N   = \l__document_structure_omgroup_display_tl,
5204   intro        .tl_set:N   = \l__document_structure_omgroup_intro_tl,
5205   loadmodules  .bool_set:N = \l__document_structure_omgroup_loadmodules_bool
5206 }
5207 \cs_new_protected:Nn \__document_structure_omgroup_args:n {
5208   \str_clear:N \l__document_structure_omgroup_id_str
5209   \str_clear:N \l__document_structure_omgroup_date_str
5210   \clist_clear:N \l__document_structure_omgroup_creators_clist
5211   \clist_clear:N \l__document_structure_omgroup_contributors_clist
5212   \tl_clear:N \l__document_structure_omgroup_srccite_tl
5213   \tl_clear:N \l__document_structure_omgroup_type_tl
5214   \tl_clear:N \l__document_structure_omgroup_short_tl
5215   \tl_clear:N \l__document_structure_omgroup_display_tl
5216   \tl_clear:N \l__document_structure_omgroup_intro_tl
5217   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
5218   \keys_set:nn { document-structure / omgroupp } { #1 }
5219 }

```

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 omgroupp, i.e. after the section heading.

```

5220 \newif\if@mainmatter\@mainmattertrue
5221 \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.

```

5222 \keys_define:nn { document-structure / sectioning }{
5223   name .str_set_x:N = \l__document_structure_sect_name_str ,
5224   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
5225   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
5226   clear .default:n = {true} ,
5227   num .bool_set:N = \l__document_structure_sect_num_bool ,

```

```

5228   num      .default:n    = {true}
5229 }
5230 \cs_new_protected:Nn \__document_structure_sect_args:n {
5231   \str_clear:N \l__document_structure_sect_name_str
5232   \str_clear:N \l__document_structure_sect_ref_str
5233   \bool_set_false:N \l__document_structure_sect_clear_bool
5234   \bool_set_false:N \l__document_structure_sect_num_bool
5235   \keys_set:nn { document-structure / sectioning } { #1 }
5236 }
5237 \newcommand\omdoc@sectioning[3][]{
5238   \__document_structure_sect_args:n {#1}
5239   \let\omdoc@sect@name\l__document_structure_sect_name_str
5240   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
5241   \if@mainmatter% numbering not overridden by frontmatter, etc.
5242     \bool_if:NTF \l__document_structure_sect_num_bool {
5243       \omgroup@num{#2}{#3}
5244     }{
5245       \omgroup@nonum{#2}{#3}
5246     }
5247     \def\current@section@level{\omdoc@sect@name}
5248   \else
5249     \omgroup@nonum{#2}{#3}
5250   \fi
5251 }% 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.

```

5252 \newcommand\omgroup@redefine@addtocontents[1]{%
5253 %\edef\__document_structureimport{#1}%
5254 %\@for\@I:=\__document_structureimport\do{%
5255 %\edef\@path{\csname module@\@I @path\endcsname}%
5256 %\@ifundefined{tf@toc}\relax%
5257 %   {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
5258 %\ifx\hyper@anchor\undefined% hyperref.sty loaded?
5259 %\def\addcontentsline##1##2##3{%
5260 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
5261 %\else% hyperref.sty not loaded
5262 %\def\addcontentsline##1##2##3{%
5263 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
5264 %\fi
5265 }% 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.

```

5266 \newenvironment{omgroup}[2][]{% keys, title
5267 {
5268   \__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.

```

5269 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
5270   \omgroup@redefine@addtocontents{
5271     \@ifundefined{module@id}\used@modules%

```

```

5272     %{\ifundefined{module@}\module@id @path}{\used@modules}\module@id}
5273   }
5274 }

```

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

```

5275 \int_incr:N\l_document_structure_section_level_int
5276 \ifcase\l_document_structure_section_level_int
5277   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
5278   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
5279   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
5280   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
5281   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
5282   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
5283   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
5284 \fi
5285 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
5286 \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
5287 }% for customization
5288 {}

```

and finally, we localize the sections

```

5289 \newcommand\omdoc@part@kw{Part}
5290 \newcommand\omdoc@chapter@kw{Chapter}
5291 \newcommand\omdoc@section@kw{Section}
5292 \newcommand\omdoc@subsection@kw{Subsection}
5293 \newcommand\omdoc@subsubsection@kw{Subsubsection}
5294 \newcommand\omdoc@paragraph@kw{paragraph}
5295 \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`

```

5296 \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).

```

5297 \cs_if_exist:NTF\frontmatter{
5298   \let\__document_structure_orig_frontmatter\frontmatter
5299   \let\frontmatter\relax
5300 }{
5301   \tl_set:Nn\__document_structure_orig_frontmatter{
5302     \clearpage
5303     \@mainmatterfalse
5304     \pagenumbering{roman}
5305   }
5306 }

```

```

5307 \cs_if_exist:NTF\backmatter{
5308   \let\__document_structure_orig_backmatter\backmatter
5309   \let\backmatter\relax
5310 }{
5311   \tl_set:Nn\__document_structure_orig_backmatter{
5312     \clearpage
5313     \@mainmatterfalse
5314     \pagenumbering{roman}
5315   }
5316 }

```

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.

```

5317 \newenvironment{frontmatter}{
5318   \__document_structure_orig_frontmatter
5319 }{
5320   \cs_if_exist:NTF\mainmatter{
5321     \mainmatter
5322   }{
5323     \clearpage
5324     \@mainmattertrue
5325     \pagenumbering{arabic}
5326   }
5327 }

```

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

```

5328 \newenvironment{backmatter}{
5329   \__document_structure_orig_backmatter
5330 }{
5331   \cs_if_exist:NTF\mainmatter{
5332     \mainmatter
5333   }{
5334     \clearpage
5335     \@mainmattertrue
5336     \pagenumbering{arabic}
5337   }
5338 }

```

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

5339 \@mainmattertrue\pagenumbering{arabic}

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

```

5340 \def \c__document_structure_document_str{document}
5341 \newcommand\afterprematurestop{}
5342 \def\prematurestop@endomgroup{
5343   \unless\ifx\@currentvir\c__document_structure_document_str
5344     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
5345     \expandafter\prematurestop@endomgroup
5346   \fi
5347 }

```

```

5348 \providecommand\prematurestop{
5349   \message{Stopping~sTeX~processing~prematurely}
5350   \prematurestop@endomgroup
5351   \afterprematurestop
5352   \end{document}
5353 }

```

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

## 38.8 Global Variables

**\setSGvar** set a global variable

```

5354 \RequirePackage{etoolbox}
5355 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

**\useSGvar** use a global variable

```

5356 \newrobustcmd\useSGvar[1]{%
5357   \@ifundefined{sTeX@Gvar@#1}
5358   {\PackageError{document-structure}
5359     {The sTeX Global variable #1 is undefined}
5360     {set it with \protect\setSGvar}}
5361   \@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.

```

5362 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
5363   \@ifundefined{sTeX@Gvar@#1}
5364   {\PackageError{document-structure}
5365     {The sTeX Global variable #1 is undefined}
5366     {set it with \protect\setSGvar}}
5367   {\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.

```
5368 \*cls)
5369 \@@=notesslides)
5370 \ProvidesExplClass{notesslides}{2022/02/10}{3.0}{notesslides Class}
5371 \RequirePackage{l3keys2e,expl-keystr-compatible}
5372
5373 \keys_define:nn{notesslides / cls}{
5374   class .code:n = {
5375     \PassOptionsToClass{\CurrentOption}{document-structure}
5376     \str_if_eq:nnT{#1}{book}{
5377       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5378     }
5379     \str_if_eq:nnT{#1}{report}{
5380       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5381     }
5382   },
5383   notes .bool_set:N = \c__notesslides_notes_bool ,
5384   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
5385   unknown .code:n = {
5386     \PassOptionsToClass{\CurrentOption}{document-structure}
5387     \PassOptionsToClass{\CurrentOption}{beamer}
5388     \PassOptionsToPackage{\CurrentOption}{notesslides}
5389   }
5390 }
5391 \ProcessKeysOptions{ notesslides / cls }
5392 \bool_if:NTF \c__notesslides_notes_bool {
5393   \PassOptionsToPackage{notes=true}{notesslides}
5394 }{
5395   \PassOptionsToPackage{notes=false}{notesslides}
5396 }
5397 \</cls)
```

now we do the same for the notesslides package.

```

5398 <*package>
5399 \ProvidesExplPackage{notesslides}{2022/02/10}{3.0}{notesslides Package}
5400 \RequirePackage{l3keys2e,expl-keystr-compat}
5401
5402 \keys_define:nn{notesslides / pkg}{
5403   topsect      .str_set_x:N = \c__notesslides_topsect_str,
5404   defaulttopsect .str_set_x:N = \c__notesslides_defaulttopsec_str,
5405   notes        .bool_set:N = \c__notesslides_notes_bool ,
5406   slides       .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
5407   sectocframes .bool_set:N = \c__notesslides_sectocframes_bool ,
5408   frameimages  .bool_set:N = \c__notesslides_frameimages_bool ,
5409   fiboxed      .bool_set:N = \c__notesslides_fiboxed_bool ,
5410   nopproblems  .bool_set:N = \c__notesslides_nopproblems_bool,
5411   unknown      .code:n      = {
5412     \PassOptionsToClass{\CurrentOption}{stex}
5413     \PassOptionsToClass{\CurrentOption}{tikzinput}
5414   }
5415 }
5416 \ProcessKeysOptions{ notesslides / pkg }
5417 \newif\ifnotes
5418 \bool_if:NTF \c__notesslides_notes_bool {
5419   \notesttrue
5420 }{
5421   \notesfalse
5422 }
5423

```

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

```

5424 \str_if_empty:NTF \c__notesslides_topsect_str {
5425   \str_set_eq:NN \__notesslides_topsect \c__notesslides_defaulttopsec_str
5426 }{
5427   \str_set_eq:NN \__notesslides_topsect \c__notesslides_topsect_str
5428 }
5429 </package>

```

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

```

5430 <*cls>
5431 \bool_if:NTF \c__notesslides_notes_bool {
5432   \LoadClass{document-structure}
5433 }{
5434   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
5435   \newcounter{Item}
5436   \newcounter{paragraph}
5437   \newcounter{subparagraph}
5438   \newcounter{Hfootnote}
5439   \RequirePackage{document-structure}
5440 }

```

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

```

5441 \RequirePackage{notesslides}
5442 </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.

```

5443 \*package>
5444 \bool_if:NT \c__notesslides_notes_bool {
5445   \RequirePackage{a4wide}
5446   \RequirePackage{marginnote}
5447   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
5448   \RequirePackage{mdframed}
5449   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
5450   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
5451 }
5452 \RequirePackage{stex-tikzinput}
5453 \RequirePackage{etoolbox}
5454 \RequirePackage{amssymb}
5455 \RequirePackage{amsmath}
5456 \RequirePackage{comment}
5457 \RequirePackage{textcomp}
5458 \RequirePackage{url}
5459 \RequirePackage{graphicx}
5460 \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>

```

5461 \bool_if:NT \c__notesslides_notes_bool {
5462   \renewcommand\usetheme[2][\usepackage[#1]{beamernotestheme#2}]
5463 }

```

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

```

5464 \newcounter{slide}
5465 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
5466 \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.

```

5467 \bool_if:NTF \c__notesslides_notes_bool {
5468   \renewenvironment{note}{\ignorespaces}{\ignorespaces}
5469 }{
5470   \excludecomment{note}
5471 }

```

---

<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.

```
5472 \bool_if:NT \c__notesslides_notes_bool {
5473   \newlength{\slideframewidth}
5474   \setlength{\slideframewidth}{1.5pt}
```

**frame** We first define the keys.

```
5475 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
5476   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
5477     \bool_set_true:N #1
5478   }{
5479     \bool_set_false:N #1
5480   }
5481 }
5482 \keys_define:nn{notesslides / frame}{
5483   label .str_set_x:N = \l__notesslides_frame_label_str,
5484   allowframebreaks .code:n = {
5485     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
5486   },
5487   allowdisplaybreaks .code:n = {
5488     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
5489   },
5490   fragile .code:n = {
5491     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
5492   },
5493   shrink .code:n = {
5494     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
5495   },
5496   squeeze .code:n = {
5497     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
5498   },
5499   t .code:n = {
5500     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
5501   },
5502 }
5503 \cs_new_protected:Nn \__notesslides_frame_args:n {
5504   \str_clear:N \l__notesslides_frame_label_str
5505   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
5506   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
5507   \bool_set_true:N \l__notesslides_frame_fragile_bool
5508   \bool_set_true:N \l__notesslides_frame_shrink_bool
5509   \bool_set_true:N \l__notesslides_frame_squeeze_bool
5510   \bool_set_true:N \l__notesslides_frame_t_bool
5511   \keys_set:nn { notesslides / frame }{ #1 }
5512 }
```

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

```
5513 \renewenvironment{frame}[1][]{
5514   \__notesslides_frame_args:n{#1}
5515   \sffamily
5516   \stepcounter{slide}
5517   \def\@currentlabel{\theslide}
5518   \str_if_empty:NF \l__notesslides_frame_label_str {
5519     \label{\l__notesslides_frame_label_str}
```

5520 }

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

5521 \def\itemize@level{outer}  
5522 \def\itemize@outer{outer}  
5523 \def\itemize@inner{inner}  
5524 \renewcommand\newpage{\addtocounter{framenum}{1}}  
5525 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}  
5526 \renewenvironment{itemize}{  
5527 \ifx\itemize@level\itemize@outer  
5528 \def\itemize@label{\$\rhd\$}  
5529 \fi  
5530 \ifx\itemize@level\itemize@inner  
5531 \def\itemize@label{\$\scriptstyle\rhd\$}  
5532 \fi  
5533 \begin{list}  
5534 {\itemize@label}  
5535 {\setlength{\labelsep}{.3em}  
5536 \setlength{\labelwidth}{.5em}  
5537 \setlength{\leftmargin}{1.5em}  
5538 }  
5539 \edef\itemize@level{\itemize@inner}  
5540 }{  
5541 \end{list}  
5542 }

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

5543 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth=100pt]  
5544 }{  
5545 \medskip\miko@slidelabel\end{mdframed}  
5546 }

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

\frametitle

5547 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip  
5548 }

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

EdN:21

\pause 21

5549 \bool\_if:NT \c\_notesslides\_notes\_bool {  
5550 \newcommand\pause{  
5551 }

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

nparagraph

5552 \bool\_if:NTF \c\_notesslides\_notes\_bool {  
5553 \newenvironment{nparagraph}[1][\begin{sparagraph}[#1]}{\end{sparagraph}}  
5554 }{  
5555 \excludecomment{nparagraph}  
5556 }

---

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

nomgroup

```
5557 \bool_if:NTF \c__notesslides_notes_bool {
5558   \newenvironment{nomgroup}[2] [] {\begin{omgroup}[#1]{#2}}{\end{omgroup}}
5559 }{
5560   \excludecomment{nomgroup}
5561 }
```

ndefinition

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

nassertion

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

nsproof

```
5572 \bool_if:NTF \c__notesslides_notes_bool {
5573   \newenvironment{nsproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
5574 }{
5575   \excludecomment{nsproof}
5576 }
```

nexample

```
5577 \bool_if:NTF \c__notesslides_notes_bool {
5578   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}{\end{sexample}}
5579 }{
5580   \excludecomment{nexample}
5581 }
```

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

```
5582 \def\inputref@preskip{\smallskip}
5583 \def\inputref@postskip{\medskip}
```

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

\inputref\*

```
5584 \let\orig@inputref\inputref
5585 \def\inputref{@ifstar\ninputref\orig@inputref}
5586 \newcommand\ninputref[2] [] {
5587   \bool_if:NT \c__notesslides_notes_bool {
5588     \orig@inputref[#1]{#2}
5589   }
5590 }
```

(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>}`.

```
5591 \newlength{\slidelogoheight}
5592
5593 \bool_if:NTF \c__notesslides_notes_bool {
5594   \setlength{\slidelogoheight}{.4cm}
5595 }{
5596   \setlength{\slidelogoheight}{1cm}
5597 }
5598 \newsavebox{\slidelogo}
5599 \sbox{\slidelogo}{\TeX}
5600 \newrobustcmd{\setslidelogo}[1]{
5601   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
5602 }
```

(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.

```
5603 \def\source{Michael Kohlhase}% customize locally
5604 \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.

```
5605 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
5606 \newsavebox{\cclogo}
5607 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
5608 \newif\ifcchref\cchreffalse
5609 \AtBeginDocument{
5610   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
5611 }
5612 \def\licensing{
5613   \ifcchref
5614     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
5615   \else
5616     {\usebox{\cclogo}}
5617   \fi
5618 }
5619 \newrobustcmd{\setlicensing}[2][]{
5620   \def\@url{#1}
5621   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
5622   \ifx\@url\@empty
5623     \def\licensing{{\usebox{\cclogo}}}
5624   \else
5625     \def\licensing{
```

```

5626     \ifcchref
5627     \href{#1}{\usebox{\cclogo}}
5628   \else
5629     {\usebox{\cclogo}}
5630   \fi
5631 }
5632 \fi
5633 }

```

(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>

```

5634 \newrobustcmd\miko@slidelabel{
5635   \vbox to \slidelogoheight{
5636     \vss\hbox to \slidewidth
5637     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
5638   }
5639 }

```

(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.

```

5640 \def\Gin@mhrepos{}
5641 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
5642 \define@key{Gin}{label}{\def\currentlabel{\arabic{slide}}\label{#1}}
5643 \newrobustcmd\frameimage[2][]{
5644   \stepcounter{slide}
5645   \bool_if:NT \c__notesslides_frameimages_bool {
5646     \def\Gin@ewidth{}\setkeys{Gin}{#1}
5647     \bool_if:NF \c__notesslides_notes_bool { \vfill }
5648     \begin{center}
5649       \bool_if:NTF \c__notesslides_fiboxed_bool {
5650         \fbox{
5651           \ifx\Gin@ewidth\@empty
5652             \ifx\Gin@mhrepos\@empty
5653               \mhgraphics[width=\slidewidth,#1]{#2}
5654             \else
5655               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5656             \fi
5657           \else% \Gin@ewidth empty
5658             \ifx\Gin@mhrepos\@empty
5659               \mhgraphics[#1]{#2}
5660             \else
5661               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5662             \fi
5663           \fi% \Gin@ewidth empty
5664         }
5665       }{
5666         \ifx\Gin@ewidth\@empty

```

---

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

```

5667         \ifx\Gin@mhrepos\@empty
5668             \mhgraphics[width=\slidewidth,#1]{#2}
5669         \else
5670             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5671         \fi
5672         \ifx\Gin@mhrepos\@empty
5673             \mhgraphics[#1]{#2}
5674         \else
5675             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5676         \fi
5677     \fi% Gin@ewidth empty
5678 }
5679 \end{center}
5680 \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
5681 \bool_if:NF \c__notesslides_notes_bool { \vfill }
5682 }
5683 } % 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.

```

5684 \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

```

5685 \AddToHook{begindocument}{
5686     \definecolor{green}{rgb}{0,.5,0}
5687     \definecolor{purple}{cmk}{.3,1,0,.17}
5688 }

```

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`.

```

5689 % \def\STpresent#1{\textcolor{blue}{#1}}
5690 \def\defemph#1{\textcolor{magenta}{#1}}
5691 \def\symrefemph#1{\textcolor{cyan}{#1}}
5692 \def\compemph#1{\textcolor{blue}{#1}}
5693 \def\titleemph#1{\textcolor{blue}{#1}}
5694 \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.

```

5695 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
5696 \def\smalltextwarning{
5697     \pgfuseimage{miko@small@dbend}
5698     \xspace
5699 }
5700 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}

```

```

5701 \newrobustcmd\textwarning{
5702   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
5703   \xspace
5704 }
5705 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}
5706 \newrobustcmd\bigtextwarning{
5707   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
5708   \xspace
5709 }

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

5710 \newrobustcmd\putgraphicsat[3]{
5711   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
5712 }
5713 \newrobustcmd\putat[2]{
5714   \begin{picture}(0,0)\put(#1){#2}\end{picture}
5715 }

```

## 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`.

```

5716 \bool_if:NT \c__notesslides_sectocframes_bool {
5717   \str_if_eq:VnTF \__notesslidesstopsect{part}{
5718     \newcounter{chapter}\counterwithin*{section}{chapter}
5719   }{
5720     \str_if_eq:VnT\__notesslidesstopsect{chapter}{
5721       \newcounter{chapter}\counterwithin*{section}{chapter}
5722     }
5723   }
5724 }

```

`\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

5725 \def\part@prefix{}
5726 \@ifpackageloaded{document-structure}{
5727   \str_case:VnF \__notesslidesstopsect {
5728     {part}{
5729       \int_set:Nn \l_document_structure_section_level_int {0}
5730       \def\thesection{\arabic{chapter}.\arabic{section}}
5731       \def\part@prefix{\arabic{chapter}.}
5732     }
5733     {chapter}{
5734       \int_set:Nn \l_document_structure_section_level_int {1}
5735       \def\thesection{\arabic{chapter}.\arabic{section}}
5736       \def\part@prefix{\arabic{chapter}.}
5737     }
5738   }{
5739     \int_set:Nn \l_document_structure_section_level_int {2}
5740     \def\part@prefix{}

```

```

5741 }
5742 }
5743
5744 \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

```

5745 \renewenvironment{omgroup}[2][]{
5746   \__document_structure_omgroup_args:n { #1 }
5747   \int_incr:N \l_document_structure_section_level_int
5748   \bool_if:NT \c__notesslides_sectocframes_bool {
5749     \stepcounter{slide}
5750     \begin{frame}[noframenumbering]
5751       \vfill\Large\centering
5752       \red{
5753         \ifcase\l_document_structure_section_level_int\or
5754           \stepcounter{part}
5755           \def\__notesslideslabel{\omdoc@part@kw~\Roman{part}}
5756           \def\currentsectionlevel{\omdoc@part@kw}
5757         \or
5758           \stepcounter{chapter}
5759           \def\__notesslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
5760           \def\currentsectionlevel{\omdoc@chapter@kw}
5761         \or
5762           \stepcounter{section}
5763           \def\__notesslideslabel{\part@prefix\arabic{section}}
5764           \def\currentsectionlevel{\omdoc@section@kw}
5765         \or
5766           \stepcounter{subsection}
5767           \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
5768           \def\currentsectionlevel{\omdoc@subsection@kw}
5769         \or
5770           \stepcounter{subsubsection}
5771           \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
5772           \def\currentsectionlevel{\omdoc@subsubsection@kw}
5773         \or
5774           \stepcounter{paragraph}
5775           \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{paragraph}}
5776           \def\currentsectionlevel{\omdoc@paragraph@kw}
5777         \else
5778           \def\__notesslideslabel{}
5779           \def\currentsectionlevel{\omdoc@paragraph@kw}
5780         \fi% end ifcase
5781         \__notesslideslabel%\sref@label@id\__notesslideslabel
5782         \quad #2%
5783       }%
5784       \vfill%
5785       \end{frame}%
5786     }
5787     \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str%
5788   }{}

```



5789 }

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

```
5790 \def\inserttheorembodyfont{\normalfont}
5791 %\bool_if:NF \c__notesslides_notes_bool {
5792 % \defbeamertemplate{theorem begin}{miko}
5793 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
5794 % \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
5795 % \inserttheorempunctuation\inserttheorembodyfont\hspace}
5796 % \defbeamertemplate{theorem end}{miko}{}}
```

and we set it as the default one.

```
5797 % \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.

```
5798 % \expandafter\def\csname Parent2\endcsname{}
5799 %}
5800
5801 \AddToHook{begindocument}{% this does not work for some reasons
5802 \setbeamertemplate{theorems}[ams style]
5803 }
5804 \bool_if:NT \c__notesslides_notes_bool {
5805 \renewenvironment{columns}[1][{}]{%
5806 \par\noindent%
5807 \begin{minipage}%
5808 \slidewidth\centering\leavevmode%
5809 }{%
5810 \end{minipage}\par\noindent%
5811 }%
5812 \newsavebox\columnbox%
5813 \renewenvironment<>{column}[2][{}]{%
5814 \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
5815 }{%
5816 \end{minipage}\end{lrbox}\usebox\columnbox%
5817 }%
5818 }
5819 \bool_if:NTF \c__notesslides_noproblems_bool {
5820 \newenvironment{problems}{}{}
5821 }{
5822 \excludecomment{problems}
5823 }
```

## 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.

```
5824 \gdef\printexcursions{}
5825 \newcommand\excursionref[2]{% label, text
5826 \bool_if:NT \c__notesslides_notes_bool {
5827 \begin{sparagraph}[title=Excursion]
```

```

5828     #2 \sref[fallback=the appendix]{#1}.
5829 \end{sparagraph}
5830 }
5831 }
5832 \newcommand\activate@excursion[2][]{
5833 \gappto\printexcursions{\inputref[#1]{#2}}
5834 }
5835 \newcommand\excursion[4][]{% repos, label, path, text
5836 \bool_if:NT \c__notesslides_notes_bool {
5837 \activate@excursion[#1]{#3}\excursionref{#2}{#4}
5838 }
5839 }

```

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

\excursiongroup

```

5840 \keys_define:nn{notesslides / excursiongroup }{
5841 id .str_set_x:N = \l__notesslides_excursion_id_str,
5842 intro .tl_set:N = \l__notesslides_excursion_intro_tl,
5843 mhrepos .str_set_x:N = \l__notesslides_excursion_mhrepos_str
5844 }
5845 \cs_new_protected:Nn \__notesslides_excursion_args:n {
5846 \tl_clear:N \l__notesslides_excursion_intro_tl
5847 \str_clear:N \l__notesslides_excursion_id_str
5848 \str_clear:N \l__notesslides_excursion_mhrepos_str
5849 \keys_set:nn {notesslides / excursiongroup }{ #1 }
5850 }
5851 \newcommand\excursiongroup[1][]{
5852 \__notesslides_excursion_args:n{ #1 }
5853 \ifdefempty\printexcursions{}% only if there are excursions
5854 {\begin{note}
5855 \begin{omgroup}[#1]{Excursions}%
5856 \ifdefempty\l__notesslides_excursion_intro_tl{\{
5857 \inputref[\l__notesslides_excursion_mhrepos_str]{
5858 \l__notesslides_excursion_intro_tl
5859 }
5860 }
5861 \printexcursions%
5862 \end{omgroup}
5863 \end{note}}
5864 }
5865 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{\fi
5866 \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.

```
5867 <*package>
5868 <@@=problems>
5869 \ProvidesExplPackage{problem}{2019/03/20}{1.3}{Semantic Markup for Problems}
5870 \RequirePackage{l3keys2e,expl-keystr-compatible}
5871
5872 \keys_define:nn { problem / pkg }{
5873   notes      .default:n    = { true },
5874   notes      .bool_set:N   = \c__problems_notes_bool,
5875   gnotes     .default:n    = { true },
5876   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
5877   hints      .default:n    = { true },
5878   hints      .bool_set:N   = \c__problems_hints_bool,
5879   solutions  .default:n    = { true },
5880   solutions  .bool_set:N   = \c__problems_solutions_bool,
5881   pts        .default:n    = { true },
5882   pts        .bool_set:N   = \c__problems_pts_bool,
5883   min        .default:n    = { true },
5884   min        .bool_set:N   = \c__problems_min_bool,
5885   boxed      .default:n    = { true },
5886   boxed      .bool_set:N   = \c__problems_boxed_bool,
5887   unknown    .code:n       = {}
5888 }
5889 \newif\ifsolutions
5890
5891 \ProcessKeysOptions{ problem / pkg }
5892 \bool_if:NTF \c__problems_solutions_bool {
5893   \solutionstrue
5894 }{
5895   \solutionsfalse
5896 }
```

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

```
5897 \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.

```
5898 \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.

```
5899 \def\prob@problem@kw{Problem}
5900 \def\prob@solution@kw{Solution}
5901 \def\prob@hint@kw{Hint}
5902 \def\prob@note@kw{Note}
5903 \def\prob@gnote@kw{Grading}
5904 \def\prob@pt@kw{pt}
5905 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
5906 \AddToHook{begindocument}{
5907   \ltx@ifpackageloaded{babel}{
5908     \makeatletter
5909     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5910     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5911       \input{problem-ngerman.ldf}
5912     }
5913     \clist_if_in:NnT \l_tmpa_clist {finnish}{
5914       \input{problem-finnish.ldf}
5915     }
5916     \clist_if_in:NnT \l_tmpa_clist {french}{
5917       \input{problem-french.ldf}
5918     }
5919     \clist_if_in:NnT \l_tmpa_clist {russian}{
5920       \input{problem-russian.ldf}
5921     }
5922     \makeatother
5923   }{ }
5924 }
```

## 40.2 Problems and Solutions

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

```
5925 \keys_define:nn{ problem / problem }{
5926   id      .str_set:x:N = \l__problems_prob_id_str,
5927   pts     .tl_set:N    = \l__problems_prob_pts_tl,
5928   min     .tl_set:N    = \l__problems_prob_min_tl,
5929   title   .tl_set:N    = \l__problems_prob_title_tl,
5930   type    .tl_set:N    = \l__problems_prob_type_tl,
5931   refnum  .int_set:N    = \l__problems_prob_refnum_int
5932 }
5933 \cs_new_protected:Nn \__problems_prob_args:n {
```

```

5934 \str_clear:N \l__problems_prob_id_str
5935 \tl_clear:N \l__problems_prob_pts_tl
5936 \tl_clear:N \l__problems_prob_min_tl
5937 \tl_clear:N \l__problems_prob_title_tl
5938 \tl_clear:N \l__problems_prob_type_tl
5939 \int_zero_new:N \l__problems_prob_refnum_int
5940 \keys_set:nn { problem / problem }{ #1 }
5941 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
5942   \let\l__problems_prob_refnum_int\undefined
5943 }
5944 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

5945 \newcounter{problem}
5946 \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.

```

5947 \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

```

5948 \newcommand\prob@number{
5949   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
5950     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
5951   }{
5952     \int_if_exist:NTF \l__problems_prob_refnum_int {
5953       \prob@label{\int_use:N \l__problems_prob_refnum_int }
5954     }{
5955       \prob@label\theproblem
5956     }
5957   }
5958 }

```

(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.

```

5959 \newcommand\prob@title[3]{%
5960   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
5961     #2 \l__problems_inclprob_title_tl #3
5962   }{
5963     \tl_if_exist:NTF \l__problems_prob_title_tl {
5964       #2 \l__problems_prob_title_tl #3
5965     }{
5966       #1
5967     }
5968   }
5969 }

```

(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.

```

5970 \def\prob@heading{
5971   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
5972   %\sref@label{id}\prob@problem@kw~\prob@number}{~}
5973 }

```

(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**

```

5974 \newenvironment{sproblem}[1][{}]{
5975   \__problems_prob_args:n{#1}%\sref@target%
5976   \@in@omtexttrue% we are in a statement (for inline definitions)
5977   \stepcounter{problem}\record@problem
5978   \def\current@section@level{\prob@problem@kw}
5979   \tl_if_exist:NTF \l__problems_inclprob_type_tl {
5980     \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
5981   }{
5982     \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
5983   }
5984   \str_if_exist:NTF \l__problems_inclprob_id_str {
5985     \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
5986   }{
5987     \str_set_eq:NN \sproblemid \l__problems_prob_id_str
5988   }
5989
5990
5991   \clist_set:No \l_tmpa_clist \sproblemtype
5992   \tl_clear:N \l_tmpa_tl
5993   \clist_map_inline:Nn \l_tmpa_clist {
5994     \tl_if_exist:cT {\__problems_sproblem_##1_start:}{
5995       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_start:}}
5996     }
5997   }
5998   \tl_if_empty:NTF \l_tmpa_tl {
5999     \__problems_sproblem_start:
6000   }{
6001     \l_tmpa_tl
6002   }
6003   \stex_ref_new_doc_target:n \sproblemid
6004 }{
6005   \clist_set:No \l_tmpa_clist \sproblemtype
6006   \tl_clear:N \l_tmpa_tl
6007   \clist_map_inline:Nn \l_tmpa_clist {
6008     \tl_if_exist:cT {\__problems_sproblem_##1_end:}{
6009       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_end:}}
6010     }

```

```

6011 }
6012 \tl_if_empty:NTF \l_tmpa_tl {
6013   \__problems_sproblem_end:
6014 }{
6015   \l_tmpa_tl
6016 }
6017
6018
6019 \smallskip
6020 }
6021
6022
6023 \cs_new_protected:Nn \__problems_sproblem_start: {
6024   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
6025 }
6026 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
6027
6028 \newcommand\stexpatchproblem[3][] {
6029   \str_set:Nx \l_tmpa_str{ #1 }
6030   \str_if_empty:NTF \l_tmpa_str {
6031     \tl_set:Nn \__problems_sproblem_start: { #2 }
6032     \tl_set:Nn \__problems_sproblem_end: { #3 }
6033   }{
6034     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
6035     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
6036   }
6037 }
6038
6039
6040 \bool_if:NT \c__problems_boxed_bool {
6041   \surroundwithmdframed{problem}
6042 }

```

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

```

6043 \def\record@problem{
6044   \protected@write\@auxout{}
6045   {
6046     \string\@problem{\prob@number}
6047     {
6048       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
6049         \l__problems_inclprob_pts_tl
6050       }{
6051         \l__problems_prob_pts_tl
6052       }
6053     }%
6054     {
6055       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
6056         \l__problems_inclprob_min_tl
6057       }{
6058         \l__problems_prob_min_tl
6059       }
6060     }
6061   }
6062 }

```

(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).

```
6063 \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.

```
6064 \keys_define:nn { problem / solution }{
6065   id               .str_set_x:N = \l__problems_solution_id_str ,
6066   for              .tl_set:N   = \l__problems_solution_for_tl ,
6067   height           .dim_set:N  = \l__problems_solution_height_dim ,
6068   creators         .clist_set:N = \l__problems_solution_creators_clist ,
6069   contributors     .clist_set:N = \l__problems_solution_contributors_clist ,
6070   srccite          .tl_set:N   = \l__problems_solution_srccite_tl
6071 }
6072 \cs_new_protected:Nn \__problems_solution_args:n {
6073   \str_clear:N \l__problems_solution_id_str
6074   \tl_clear:N \l__problems_solution_for_tl
6075   \tl_clear:N \l__problems_solution_srccite_tl
6076   \clist_clear:N \l__problems_solution_creators_clist
6077   \clist_clear:N \l__problems_solution_contributors_clist
6078   \dim_zero:N \l__problems_solution_height_dim
6079   \keys_set:nn { problem / solution }{ #1 }
6080 }
```

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

```
6081 \newcommand\@startsolution[1][{}]{
6082   \__problems_solution_args:n { #1 }
6083   \@in@omtexttrue% we are in a statement.
6084   \bool_if:NF \c__problems_boxed_bool { \hrule }
6085   \smallskip\noindent
6086   {\textbf\prob@solution@kw : \enspace}
6087   \begin{small}
6088   \def\current@section@level{\prob@solution@kw}
6089   \ignorespacesandpars
6090 }
```

**\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.

```
6091 \newcommand\startsolutions{
6092   \specialcomment{solution}{\@startsolution}{
6093     \bool_if:NF \c__problems_boxed_bool {
6094       \hrule\medskip
6095     }
6096     \end{small}%
6097   }
6098   \bool_if:NT \c__problems_boxed_bool {
6099     \surroundwithmdframed{solution}
6100   }
6101 }
```



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

\stopsolutions

```
6102 \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.

```
6103 \ifsolutions
```

```
6104 \startsolutions
```

```
6105 \else
```

```
6106 \stopsolutions
```

```
6107 \fi
```

exnote

```
6108 \bool_if:NTF \c__problems_notes_bool {
```

```
6109 \newenvironment{exnote}[1][]{
```

```
6110 \par\smallskip\hrule\smallskip
```

```
6111 \noindent\textbf{\prob@note@kw : }\small
```

```
6112 }{
```

```
6113 \smallskip\hrule
```

```
6114 }
```

```
6115 }{
```

```
6116 \excludecomment{exnote}
```

```
6117 }
```

hint

```
6118 \bool_if:NTF \c__problems_notes_bool {
```

```
6119 \newenvironment{hint}[1][]{
```

```
6120 \par\smallskip\hrule\smallskip
```

```
6121 \noindent\textbf{\prob@hint@kw :~ }\small
```

```
6122 }{
```

```
6123 \smallskip\hrule
```

```
6124 }
```

```
6125 \newenvironment{exhint}[1][]{
```

```
6126 \par\smallskip\hrule\smallskip
```

```
6127 \noindent\textbf{\prob@hint@kw :~ }\small
```

```
6128 }{
```

```
6129 \smallskip\hrule
```

```
6130 }
```

```
6131 }{
```

```
6132 \excludecomment{hint}
```

```
6133 \excludecomment{exhint}
```

```
6134 }
```

gnote

```
6135 \bool_if:NTF \c__problems_notes_bool {
```

```
6136 \newenvironment{gnote}[1][]{
```

```
6137 \par\smallskip\hrule\smallskip
```

```
6138 \noindent\textbf{\prob@gnote@kw : }\small
```

```
6139 }{
```

```
6140 \smallskip\hrule
```

```
6141 }
```

```
6142 }{
```

```
6143 \excludecomment{gnote}
```

```
6144 }
```

## 40.3 Multiple Choice Blocks

```

6145 \newenvironment{mcb}{
6146   \begin{enumerate}
6147 }{
6148   \end{enumerate}
6149 }

```

we define the keys for the mcc macro

```

6150 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
6151   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
6152     \bool_set_true:N #1
6153   }{
6154     \bool_set_false:N #1
6155   }
6156 }
6157 \keys_define:nn { problem / mcc }{
6158   id          .str_set:x:N = \l__problems_mcc_id_str ,
6159   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
6160   T           .default:n    = { true } ,
6161   T           .bool_set:N   = \l__problems_mcc_t_bool ,
6162   F           .default:n    = { true } ,
6163   F           .bool_set:N   = \l__problems_mcc_f_bool ,
6164   Ttext       .code:n       = {
6165     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
6166   } ,
6167   Ftext       .code:n       = {
6168     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
6169   }
6170 }
6171 \cs_new_protected:Nn \l__problems_mcc_args:n {
6172   \str_clear:N \l__problems_mcc_id_str
6173   \tl_clear:N \l__problems_mcc_feedback_tl
6174   \bool_set_true:N \l__problems_mcc_t_bool
6175   \bool_set_true:N \l__problems_mcc_f_bool
6176   \bool_set_true:N \l__problems_mcc_Ttext_bool
6177   \bool_set_false:N \l__problems_mcc_Ftext_bool
6178   \keys_set:nn { problem / mcc }{ #1 }
6179 }

```

\mcc

```

6180 \newcommand\mcc[2][] {
6181   \l__problems_mcc_args:n{ #1 }
6182   \item #2
6183   \ifsolutions
6184     \\\
6185     \bool_if:NT \l__problems_mcc_t_bool {
6186       % TODO!
6187       % \ifcsstring{mcc@T}{T}{\mcc@Ttext}%
6188     }
6189     \bool_if:NT \l__problems_mcc_f_bool {

```

---

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

```

6190      % TODO!
6191      % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
6192    }
6193    \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
6194      !
6195    }{
6196      \l__problems_mcc_feedback_tl
6197    }
6198    \fi
6199  } %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.

```

6200
6201 \keys_define:nn{ problem / inclproblem }{
6202   id      .str_set:N = \l__problems_inclprob_id_str,
6203   pts     .tl_set:N  = \l__problems_inclprob_pts_tl,
6204   min     .tl_set:N  = \l__problems_inclprob_min_tl,
6205   title   .tl_set:N  = \l__problems_inclprob_title_tl,
6206   refnum  .int_set:N  = \l__problems_inclprob_refnum_int,
6207   type    .tl_set:N  = \l__problems_inclprob_type_tl,
6208   mhrepos .str_set:N = \l__problems_inclprob_mhrepos_str
6209 }
6210 \cs_new_protected:Nn \l__problems_inclprob_args:n {
6211   \str_clear:N \l__problems_prob_id_str
6212   \tl_clear:N \l__problems_inclprob_pts_tl
6213   \tl_clear:N \l__problems_inclprob_min_tl
6214   \tl_clear:N \l__problems_inclprob_title_tl
6215   \tl_clear:N \l__problems_inclprob_type_tl
6216   \int_zero_new:N \l__problems_inclprob_refnum_int
6217   \str_clear:N \l__problems_inclprob_mhrepos_str
6218   \keys_set:nn { problem / inclproblem }{ #1 }
6219   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
6220     \let\l__problems_inclprob_pts_tl\undefined
6221   }
6222   \tl_if_empty:NT \l__problems_inclprob_min_tl {
6223     \let\l__problems_inclprob_min_tl\undefined
6224   }
6225   \tl_if_empty:NT \l__problems_inclprob_title_tl {
6226     \let\l__problems_inclprob_title_tl\undefined
6227   }
6228   \tl_if_empty:NT \l__problems_inclprob_type_tl {
6229     \let\l__problems_inclprob_type_tl\undefined
6230   }
6231   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
6232     \let\l__problems_inclprob_refnum_int\undefined
6233   }
6234 }

```

```

6235
6236 \cs_new_protected:Nn \__problems_inclprob_clear: {
6237   \let\l__problems_inclprob_id_str\undefined
6238   \let\l__problems_inclprob_pts_tl\undefined
6239   \let\l__problems_inclprob_min_tl\undefined
6240   \let\l__problems_inclprob_title_tl\undefined
6241   \let\l__problems_inclprob_type_tl\undefined
6242   \let\l__problems_inclprob_refnum_int\undefined
6243   \let\l__problems_inclprob_mhrepos_str\undefined
6244 }
6245 \__problems_inclprob_clear:
6246
6247 \newcommand\includeproblem[2][ ]{
6248   \__problems_inclprob_args:n{ #1 }
6249   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
6250     \input{#2}
6251   }{
6252     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
6253       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
6254     }
6255   }
6256   \__problems_inclprob_clear:
6257 }

```

(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.

```

6258 \AddToHook{enddocument}{
6259   \bool_if:NT \c__problems_pts_bool {
6260     \message{Total:~\arabic{pts}~points}
6261   }
6262   \bool_if:NT \c__problems_min_bool {
6263     \message{Total:~\arabic{min}~minutes}
6264   }
6265 }

```

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

```

6266 \def\pts#1{
6267   \bool_if:NT \c__problems_pts_bool {
6268     \marginpar{#1~\prob@pt@kw}
6269   }
6270 }
6271 \def\min#1{
6272   \bool_if:NT \c__problems_min_bool {
6273     \marginpar{#1~\prob@min@kw}
6274   }
6275 }

```

`\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.

```

6276 \newcounter{pts}
6277 \def\show@pts{
6278   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
6279     \bool_if:NT \c__problems_pts_bool {
6280       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
6281       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
6282     }
6283   }{
6284     \tl_if_exist:NT \l__problems_prob_pts_tl {
6285       \bool_if:NT \c__problems_pts_bool {
6286         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
6287         \addtocounter{pts}{\l__problems_prob_pts_tl}
6288       }
6289     }
6290   }
6291 }

```

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

`\show@min`

```

6292 \newcounter{min}
6293 \def\show@min{
6294   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
6295     \bool_if:NT \c__problems_min_bool {
6296       \marginpar{\l__problems_inclprob_min_tl\ min}
6297       \addtocounter{min}{\l__problems_inclprob_min_tl}
6298     }
6299   }{
6300     \tl_if_exist:NT \l__problems_prob_min_tl {
6301       \bool_if:NT \c__problems_min_bool {
6302         \marginpar{\l__problems_prob_min_tl\ min}
6303         \addtocounter{min}{\l__problems_prob_min_tl}
6304       }
6305     }
6306   }
6307 }
6308 </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.

```
6309 <@@=hwexam>
6310 <*cls>
6311 \ProvidesExplClass{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
6312 \RequirePackage{l3keys2e,expl-keystr-compatible}
6313 \DeclareOption*{
6314   \PassOptionsToClass{\CurrentOption}{document-structure}
6315   \PassOptionsToPackage{\CurrentOption}{stex}
6316   \PassOptionsToPackage{\CurrentOption}{hwexam}
6317   \PassOptionsToPackage{\CurrentOption}{tikzinput}
6318 }
6319 \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.

```
6320 \LoadClass{document-structure}
6321 \RequirePackage{stex}
6322 \RequirePackage{hwexam}
6323 \RequirePackage{tikzinput}
6324 \RequirePackage{graphicx}
6325 \RequirePackage{a4wide}
6326 \RequirePackage{amssymb}
6327 \RequirePackage{amstext}
6328 \RequirePackage{amsmath}
```

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

```

6329 \newcommand\assig@default@type{\hwexam@assignment@kw}
6330 \def\document@hwexamtype{\assig@default@type}
6331 <@@=document_structure>
6332 \keys_define:nn { document-structure / document }{
6333 id .str_set_x:N = \c_document_structure_document_id_str,
6334 hwexamtype .tl_set:N = \document@hwexamtype
6335 }
6336 <@@=hwexam>
6337 </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.

```
6338 \*package>
6339 \ProvidesExplPackage{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
6340 \RequirePackage{l3keys2e,expl-keystr-compat}
6341
6342 \newif\iftest\testfalse
6343 \DeclareOption{test}{\testtrue}
6344 \newif\ifmultiple\multiplefalse
6345 \DeclareOption{multiple}{\multipletrue}
6346 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
6347 \ProcessOptions
```

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

```
6348 \RequirePackage{keyval}[1997/11/10]
6349 \RequirePackage{problem}
```

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

```
6350 \newcommand\hwexam@assignment@kw{Assignment}
6351 \newcommand\hwexam@given@kw{Given}
6352 \newcommand\hwexam@due@kw{Due}
6353 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
6354 blank~for~extra~space}
6355 \def\hwexam@minutes@kw{minutes}
6356 \newcommand\correction@probs@kw{prob.}
6357 \newcommand\correction@pts@kw{total}
6358 \newcommand\correction@reached@kw{reached}
6359 \newcommand\correction@sum@kw{Sum}
6360 \newcommand\correction@grade@kw{grade}
6361 \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

```

6362 \AddToHook{begindocument}{
6363 \ltx@ifpackageloaded{babel}{
6364 \makeatletter
6365 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6366 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6367 \input{hwexam-ngerman.ldf}
6368 }
6369 \clist_if_in:NnT \l_tmpa_clist {finnish}{
6370 \input{hwexam-finnish.ldf}
6371 }
6372 \clist_if_in:NnT \l_tmpa_clist {french}{
6373 \input{hwexam-french.ldf}
6374 }
6375 \clist_if_in:NnT \l_tmpa_clist {russian}{
6376 \input{hwexam-russian.ldf}
6377 }
6378 \makeatother
6379 }{}
6380 }
6381

```

## 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.

```

6382 \newcounter{assignment}
6383 \numberproblemsin{assignment}
6384 \renewcommand\prob@label[1]{\assignment@number.#1}

```

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

```

6385 \keys_define:nn { hwexam / assignment } {
6386 id .str_set:N = \l__hwexam_assign_id_str,
6387 number .int_set:N = \l__hwexam_assign_number_int,
6388 title .tl_set:N = \l__hwexam_assign_title_tl,
6389 type .tl_set:N = \l__hwexam_assign_type_tl,
6390 given .tl_set:N = \l__hwexam_assign_given_tl,
6391 due .tl_set:N = \l__hwexam_assign_due_tl,
6392 loadmodules .code:n = {
6393 \bool_set_true:N \l__hwexam_assign_loadmodules_bool
6394 }
6395 }
6396 \cs_new_protected:Nn \__hwexam_assignment_args:n {
6397 \str_clear:N \l__hwexam_assign_id_str
6398 \int_set:Nn \l__hwexam_assign_number_int {-1}
6399 \tl_clear:N \l__hwexam_assign_title_tl
6400 \tl_clear:N \l__hwexam_assign_type_tl
6401 \tl_clear:N \l__hwexam_assign_given_tl
6402 \tl_clear:N \l__hwexam_assign_due_tl
6403 \bool_set_false:N \l__hwexam_assign_loadmodules_bool

```

```

6404 \keys_set:nn { hwexam / assignment }{ #1 }
6405 }

```

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.

```

6406 \newcommand\given@due[2]{
6407 \bool_lazy_all:nF {
6408 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
6409 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
6410 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
6411 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
6412 }{ #1 }
6413
6414 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
6415 \tl_if_empty:NF \l__hwexam_assign_given_tl {
6416 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
6417 }
6418 }{
6419 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
6420 }
6421
6422 \bool_lazy_or:nnF {
6423 \bool_lazy_and_p:nn {
6424 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
6425 }{
6426 \tl_if_empty_p:V \l__hwexam_assign_due_tl
6427 }
6428 }{
6429 \bool_lazy_and_p:nn {
6430 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
6431 }{
6432 \tl_if_empty_p:V \l__hwexam_assign_due_tl
6433 }
6434 }{ ,~ }
6435
6436 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
6437 \tl_if_empty:NF \l__hwexam_assign_due_tl {
6438 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
6439 }
6440 }{
6441 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
6442 }
6443
6444 \bool_lazy_all:nF {
6445 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
6446 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
6447 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
6448 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
6449 }{ #2 }
6450 }

```

`\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.

```

6451 \newcommand\assignment@title[3]{
6452 \tl_if_empty:NTF \l__hwexam_inclasssign_title_tl {
6453 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
6454 #1
6455 }{
6456 #2\l__hwexam_assign_title_tl#3
6457 }
6458 }{
6459 #2\l__hwexam_inclasssign_title_tl#3
6460 }
6461 }

```

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

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

```

6462 \newcommand\assignment@number{
6463 \int_compare:nNnTF \l__hwexam_inclasssign_number_int = {-1} {
6464 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
6465 \arabic{assignment}
6466 } {
6467 \int_use:N \l__hwexam_assign_number_int
6468 }
6469 }{
6470 \int_use:N \l__hwexam_inclasssign_number_int
6471 }
6472 }

```

(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.

```

6473 \newenvironment{assignment}[1][]{
6474 \__hwexam_assignment_args:n { #1 }
6475 %\sref@target
6476 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
6477 \global\stepcounter{assignment}
6478 }{
6479 \global\setcounter{assignment}{\int_use:N\l__hwexam_assign_number_int}
6480 }
6481 \setcounter{problem}{0}
6482 \def\current@section@level{\document@hwexamtype}
6483 %\sref@label@id{\document@hwexamtype \thesection}
6484 \begin{@assignment}
6485 }{
6486 \end{@assignment}
6487 }

```

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

```

6488 \def\ass@title{
6489 \protect\document@hwexamtype~\arabic{assignment}
6490 \assignment@title{}\{;\}{} -- \given@due{}\}
6491 }
6492 \ifmultiple
6493 \newenvironment{@assignment}{
6494 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
6495 \begin{omgroup}[loadmodules]{\ass@title}
6496 }{
6497 \begin{omgroup}{\ass@title}
6498 }
6499 }{
6500 \end{omgroup}
6501 }

```

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

```

6502 \else
6503 \newenvironment{@assignment}{
6504 \begin{center}\bf
6505 \Large@title\strut\\
6506 \document@hwexamtype~\arabic{assignment}\assignment@title{}\{;\}{}{\}
6507 \large\given@due{--;\}{}{;\}{}
6508 \end{center}
6509 }{}
6510 \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.

```

6511 \keys_define:nn { hwexam / inclassignment } {
6512 %id .str_set_x:N = \l__hwexam_assign_id_str,
6513 number .int_set:N = \l__hwexam_inclassign_number_int,
6514 title .tl_set:N = \l__hwexam_inclassign_title_tl,
6515 type .tl_set:N = \l__hwexam_inclassign_type_tl,
6516 given .tl_set:N = \l__hwexam_inclassign_given_tl,
6517 due .tl_set:N = \l__hwexam_inclassign_due_tl,
6518 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
6519 }
6520 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
6521 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
6522 \tl_clear:N \l__hwexam_inclassign_title_tl
6523 \tl_clear:N \l__hwexam_inclassign_type_tl
6524 \tl_clear:N \l__hwexam_inclassign_given_tl
6525 \tl_clear:N \l__hwexam_inclassign_due_tl
6526 \str_clear:N \l__hwexam_inclassign_mhrepos_str
6527 \keys_set:nn { hwexam / inclassignment }{ #1 }
6528 }
6529 \__hwexam_inclassignment_args:n {}
6530
6531 \newcommand\inputassignment[2][{}]{

```

```

6532 \_hwexam_inclassnment_args:n { #1 }
6533 \str_if_empty:NTF \l__hwexam_inclassn_mhrepos_str {
6534 \input{#2}
6535 }{
6536 \stex_in_repository:nn{\l__hwexam_inclassn_mhrepos_str}{
6537 \input{\mhp{path}\l__hwexam_inclassn_mhrepos_str}{#2}}
6538 }
6539 }
6540 \_hwexam_inclassnment_args:n {}
6541 }
6542 \newcommand\includeassignment[2][ ]{
6543 \newpage
6544 \inputassignment[#1]{#2}
6545 }

```

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

## 42.4 Typesetting Exams

\quizheading

```

6546 \ExplSyntaxOff
6547 \newcommand\quizheading[1]{%
6548 \def\@tas{#1}%
6549 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
6550 \ifx\@tas\@empty\else%
6551 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
6552 \fi%
6553 }
6554 \ExplSyntaxOn

```

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

\testheading

```

6555
6556 \def\hwexamheader{\input{hwexam-default.header}}
6557
6558 \def\hwexamminutes{
6559 \tl_if_empty:NTF \testheading@duration {
6560 {\testheading@min}~\hwexam@minutes@kw
6561 }{
6562 \testheading@duration
6563 }
6564 }
6565
6566 \keys_define:nn { hwexam / testheading } {
6567 min .tl_set:N = \testheading@min,
6568 duration .tl_set:N = \testheading@duration,
6569 reqpts .tl_set:N = \testheading@reqpts,
6570 tools .tl_set:N = \testheading@tools
6571 }
6572 \cs_new_protected:Nn \_hwexam_testheading_args:n {
6573 \tl_clear:N \testheading@min
6574 \tl_clear:N \testheading@duration

```

```

6575 \tl_clear:N \testheading@reqpts
6576 \tl_clear:N \testheading@tools
6577 \keys_set:nn { hwexam / testheading }{ #1 }
6578 }
6579 \newenvironment{testheading}[1][]{
6580   \__hwexam_testheading_args:n{ #1 }
6581   \newcount\check@time\check@time=\testheading@min
6582   \advance\check@time by -\theassignment@totalmin
6583   \newif\if@bonuspoints
6584   \tl_if_empty:NTF \testheading@reqpts {
6585     \@bonuspointsfalse
6586   }{
6587     \newcount\bonus@pts
6588     \bonus@pts=\theassignment@totalpts
6589     \advance\bonus@pts by -\testheading@reqpts
6590     \edef\bonus@pts{\the\bonus@pts}
6591     \@bonuspointstrue
6592   }
6593   \edef\check@time{\the\check@time}
6594
6595   \makeatletter\hwexamheader\makeatother
6596 }{
6597   \newpage
6598 }

```

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

\testspace

```

6599 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

6600 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

6601 \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.

```

6602 <@=problems>
6603 \renewcommand\@problem[3]{
6604   \stepcounter{assignment@probs}
6605   \def\__problemspts{#2}
6606   \ifx\__problemspts\@empty\else
6607     \addtocounter{assignment@totalpts}{#2}
6608   \fi
6609   \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
6610   \xdef\correction@probs{\correction@probs & #1}%
6611   \xdef\correction@pts{\correction@pts & #2}
6612   \xdef\correction@reached{\correction@reached &}

```

```

6613 }
6614 <@@=hwexam>

```

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

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

```

6615 \newcounter{assignment@probs}
6616 \newcounter{assignment@totalpts}
6617 \newcounter{assignment@totalmin}
6618 \def\correction@probs{\correction@probs@kw}
6619 \def\correction@pts{\correction@pts@kw}
6620 \def\correction@reached{\correction@reached@kw}
6621 \stepcounter{assignment@probs}
6622 \newcommand\correction@table{
6623 \resizebox{\textwidth}{!}{%
6624 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
6625 &\multicolumn{\theassignment@probs}{c|}|%|
6626 {\footnotesize\correction@forgrading@kw} &\\ \hline
6627 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
6628 \correction@pts & \theassignment@totalpts & \\ \hline
6629 \correction@reached & & \[.7cm]\hline
6630 \end{tabular}}
6631 </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}

```