

# The sTeX3 Package \*

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

2022-03-05

## Abstract

sTeX is a collection of L<sup>A</sup>T<sub>E</sub>X packages 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 2022-03-05)

# Contents

|           |  |           |
|-----------|--|-----------|
| <b>I</b>  | <b>Manual</b>  | <b>1</b>  |
| <b>1</b>  | <b>What is <math>\text{\texttt{sTeX}}</math>?</b>  | <b>2</b>  |
| <b>2</b>  | <b>Quickstart</b>  | <b>3</b>  |
| 2.1       | Setup  | 3         |
| 2.1.1     | The $\text{\texttt{sTeX}}$ IDE   | 3         |
| 2.1.2     | Manual Setup   | 3         |
| 2.2       | A First $\text{\texttt{sTeX}}$ Document  | 4         |
| 2.2.1     | OMDoc/xhtml Conversion   | 7         |
| <b>3</b>  | <b>Creating <math>\text{\texttt{sTeX}}</math> Content</b>                                  | <b>9</b>  |
| 3.1       | How Knowledge is Organized in $\text{\texttt{sTeX}}$                                       | 9         |
| 3.2       | $\text{\texttt{sTeX}}$ Archives  | 10        |
| 3.2.1     | The Local MathHub-Directory  | 10        |
| 3.2.2     | The Structure of $\text{\texttt{sTeX}}$ Archives   | 10        |
| 3.2.3     | MANIFEST.MF-Files  | 11        |
| 3.2.4     | Using Files in $\text{\texttt{sTeX}}$ Archives Directly                                    | 12        |
| 3.3       | Module, Symbol and Notation Declarations   | 13        |
| 3.3.1     | The <code>smodule</code> -Environment  | 13        |
| 3.3.2     | Declaring New Symbols and Notations  | 14        |
| 3.4       | Using Semantic Macros and Referencing Symbols  | 17        |
| 3.5       | Advanced Structuring Mechanisms  | 17        |
| 3.6       | Primitive Symbols (The $\text{\texttt{sTeX}}$ Metatheory)                                  | 19        |
| <b>4</b>  | <b><math>\text{\texttt{sTeX}}</math> Statements (Definitions, Theorems, Examples, ...)</b> | <b>20</b> |
| <b>5</b>  | <b>Additional Packages</b>   | <b>21</b> |
| 5.1       | Modular Document Structuring   | 21        |
| 5.2       | Slides and Course Notes  | 21        |
| 5.3       | Homework, Problems and Exams   | 21        |
| <b>6</b>  | <b>Stuff</b>   | <b>22</b> |
| 6.0.1     | Semantic Macros and Notations  | 22        |
|           | Other Argument Types   | 25        |
|           | Precedences  | 26        |
| 6.0.2     | Archives and Imports   | 27        |
|           | Namespaces   | 27        |
|           | Paths in Import-Statements   | 27        |
| <b>II</b> | <b>Documentation</b>   | <b>29</b> |
| <b>7</b>  | <b><math>\text{\texttt{sTeX}}</math>-Basics</b>  | <b>30</b> |
| 7.1       | Macros and Environments  | 30        |
| 7.1.1     | HTML Annotations   | 30        |
| 7.1.2     | Babel Languages  | 31        |
| 7.1.3     | Auxiliary Methods  | 31        |

|            |   |           |
|------------|---|-----------|
| <b>8</b>   | <b><code>sTeX</code>-MathHub</b>                              | <b>32</b> |
| 8.1        | Macros and Environments                                       | 32        |
| 8.1.1      | Files, Paths, URIs  | 32        |
| 8.1.2      | MathHub Archives  | 33        |
| 8.1.3      | Using Content in Archives                                     | 34        |
| <b>9</b>   | <b><code>sTeX</code>-References</b>                           | <b>35</b> |
| 9.1        | Macros and Environments                                       | 35        |
| 9.1.1      | Setting Reference Targets                                     | 35        |
| 9.1.2      | Using References  | 36        |
| <b>10</b>  | <b><code>sTeX</code>-Modules</b>                              | <b>37</b> |
| 10.1       | Macros and Environments                                       | 37        |
| 10.1.1     | The <code>smodule</code> environment                          | 39        |
| <b>11</b>  | <b><code>sTeX</code>-Module Inheritance</b>                   | <b>41</b> |
| 11.1       | Macros and Environments                                       | 41        |
| 11.1.1     | SMS Mode  | 41        |
| 11.1.2     | Imports and Inheritance                                       | 42        |
| <b>12</b>  | <b><code>sTeX</code>-Symbols</b>                              | <b>44</b> |
| 12.1       | Macros and Environments                                       | 44        |
| <b>13</b>  | <b><code>sTeX</code>-Terms</b>                                | <b>46</b> |
| 13.1       | Macros and Environments                                       | 46        |
| <b>14</b>  | <b><code>sTeX</code>-Structural Features</b>                  | <b>48</b> |
| 14.1       | Macros and Environments                                       | 48        |
| 14.1.1     | Structures  | 48        |
| <b>15</b>  | <b><code>sTeX</code>-Statements</b>                           | <b>49</b> |
| 15.1       | Macros and Environments                                       | 49        |
| <b>16</b>  | <b><code>sTeX</code>-Proofs: Structural Markup for Proofs</b> | <b>50</b> |
| 16.1       | Introduction  | 52        |
| 16.2       | The User Interface  | 53        |
| 16.2.1     | Package Options   | 53        |
| 16.2.2     | Proofs and Proof steps  | 53        |
| 16.2.3     | Justifications  | 53        |
| 16.2.4     | Proof Structure   | 55        |
| 16.2.5     | Proof End Markers   | 55        |
| 16.2.6     | Configuration of the Presentation                             | 55        |
| 16.3       | Limitations   | 56        |
| <b>17</b>  | <b><code>sTeX</code>-Metatheory</b>                           | <b>57</b> |
| 17.1       | Symbols   | 57        |
| <b>III</b> | <b>Extensions</b>   | <b>58</b> |

|  |           |
|--|-----------|
| <b>18 Tikzinput</b>  | <b>59</b> |
| 18.1 Macros and Environments . . . . .   | 59        |
| <b>19 document-structure: Semantic Markup for Open Mathematical Documents in L<sup>A</sup>T<sub>E</sub>X</b> | <b>60</b> |
| 19.1 Introduction . . . . .  | 60        |
| 19.2 The User Interface . . . . .  | 61        |
| 19.2.1 Package and Class Options . . . . .   | 61        |
| 19.2.2 Document Structure . . . . .  | 61        |
| 19.2.3 Ignoring Inputs . . . . .   | 63        |
| 19.2.4 Structure Sharing . . . . .   | 63        |
| 19.2.5 Global Variables . . . . .  | 63        |
| 19.2.6 Colors . . . . .  | 64        |
| 19.3 Limitations . . . . .   | 64        |
| <b>20 NotesSlides – Slides and Course Notes</b>  | <b>65</b> |
| 20.1 Introduction . . . . .  | 65        |
| 20.2 The User Interface . . . . .  | 65        |
| 20.2.1 Package Options . . . . .   | 65        |
| 20.2.2 Notes and Slides . . . . .  | 66        |
| 20.2.3 Header and Footer Lines of the Slides . . . . .   | 67        |
| 20.2.4 Frame Images . . . . .  | 67        |
| 20.2.5 Colors and Highlighting . . . . .   | 68        |
| 20.2.6 Front Matter, Titles, etc. . . . .  | 68        |
| 20.2.7 Excursions . . . . .  | 68        |
| 20.2.8 Miscellaneous . . . . .   | 69        |
| 20.3 Limitations . . . . .   | 69        |
| <b>21 problem.sty: An Infrastructure for formatting Problems</b>   | <b>70</b> |
| 21.1 Introduction . . . . .  | 70        |
| 21.2 The User Interface . . . . .  | 70        |
| 21.2.1 Package Options . . . . .   | 70        |
| 21.2.2 Problems and Solutions . . . . .  | 71        |
| 21.2.3 Multiple Choice Blocks . . . . .  | 72        |
| 21.2.4 Including Problems . . . . .  | 72        |
| 21.2.5 Reporting Metadata . . . . .  | 72        |
| 21.3 Limitations . . . . .   | 72        |
| <b>22 hwexam.sty/cls: An Infrastructure for formatting Assignments and Exams</b>                             | <b>74</b> |
| 22.1 Introduction . . . . .  | 75        |
| 22.2 The User Interface . . . . .  | 75        |
| 22.2.1 Package and Class Options . . . . .   | 75        |
| 22.2.2 Assignments . . . . .   | 75        |
| 22.2.3 Typesetting Exams . . . . .   | 75        |
| 22.2.4 Including Assignments . . . . .   | 76        |
| 22.3 Limitations . . . . .   | 76        |
| <b>IV Implementation</b>   | <b>78</b> |

|           |  |            |
|-----------|--|------------|
| <b>23</b> | <b>STeX-Basics Implementation</b>              | <b>79</b>  |
| 23.1      | The STeXDocument Class . . . . .               | 79         |
| 23.2      | Preliminaries . . . . .                        | 79         |
| 23.3      | Messages and logging . . . . .                 | 80         |
| 23.4      | HTML Annotations . . . . .                     | 81         |
| 23.5      | Babel Languages . . . . .                      | 84         |
| 23.6      | Auxiliary Methods . . . . .                    | 85         |
| <b>24</b> | <b>STeX-MathHub Implementation</b>             | <b>86</b>  |
| 24.1      | Generic Path Handling . . . . .                | 86         |
| 24.2      | PWD and kpsewhich . . . . .                    | 88         |
| 24.3      | File Hooks and Tracking . . . . .              | 89         |
| 24.4      | MathHub Repositories . . . . .                 | 90         |
| 24.5      | Using Content in Archives . . . . .            | 94         |
| <b>25</b> | <b>STeX-References Implementation</b>          | <b>99</b>  |
| 25.1      | Document URIs and URLs . . . . .               | 99         |
| 25.2      | Setting Reference Targets . . . . .            | 101        |
| 25.3      | Using References . . . . .                     | 103        |
| <b>26</b> | <b>STeX-Modules Implementation</b>             | <b>106</b> |
| 26.1      | The smodule environment . . . . .              | 110        |
| 26.2      | Invoking modules . . . . .                     | 115        |
| <b>27</b> | <b>STeX-Module Inheritance Implementation</b>  | <b>117</b> |
| 27.1      | SMS Mode . . . . .                             | 117        |
| 27.2      | Inheritance . . . . .                          | 120        |
| <b>28</b> | <b>STeX-Symbols Implementation</b>             | <b>125</b> |
| 28.1      | Symbol Declarations . . . . .                  | 125        |
| 28.2      | Notations . . . . .                            | 131        |
| 28.3      | Variables . . . . .                            | 141        |
| <b>29</b> | <b>STeX-Terms Implementation</b>               | <b>147</b> |
| 29.1      | Symbol Invocations . . . . .                   | 147        |
| 29.2      | Terms . . . . .                                | 154        |
| 29.3      | Notation Components . . . . .                  | 158        |
| 29.4      | Variables . . . . .                            | 160        |
| 29.5      | Sequences . . . . .                            | 162        |
| <b>30</b> | <b>STeX-Structural Features Implementation</b> | <b>163</b> |
| 30.1      | Imports with modification . . . . .            | 164        |
| 30.2      | The feature environment . . . . .              | 170        |
| 30.3      | Structure . . . . .                            | 171        |
| <b>31</b> | <b>STeX-Statements Implementation</b>          | <b>180</b> |
| 31.1      | Definitions . . . . .                          | 180        |
| 31.2      | Assertions . . . . .                           | 185        |
| 31.3      | Examples . . . . .                             | 188        |
| 31.4      | Logical Paragraphs . . . . .                   | 191        |

|   |            |
|---|------------|
| <b>32 The Implementation</b>                                    | <b>196</b> |
| 32.1 Package Options . . . . .                                  | 196        |
| 32.2 Proofs . . . . .   | 196        |
| 32.3 Justifications . . . . .                                   | 207        |
| <b>33 <math>\text{\LaTeX}</math>-Others Implementation</b>      | <b>209</b> |
| <b>34 <math>\text{\LaTeX}</math>-Metatheory Implementation</b>  | <b>210</b> |
| <b>35 Tikzinput Implementation</b>                              | <b>213</b> |
| <b>36 document-structure.sty Implementation</b>                 | <b>215</b> |
| 36.1 The document-structure Class . . . . .                     | 215        |
| 36.2 Class Options . . . . .                                    | 215        |
| 36.3 Beefing up the <code>document</code> environment . . . . . | 216        |
| 36.4 Implementation: document-structure Package . . . . .       | 216        |
| 36.5 Package Options . . . . .                                  | 216        |
| 36.6 Document Structure . . . . .                               | 218        |
| 36.7 Front and Backmatter . . . . .                             | 221        |
| 36.8 Global Variables . . . . .                                 | 223        |
| <b>37 NotesSlides – Implementation</b>                          | <b>224</b> |
| 37.1 Class and Package Options . . . . .                        | 224        |
| 37.2 Notes and Slides . . . . .                                 | 226        |
| 37.3 Header and Footer Lines . . . . .                          | 230        |
| 37.4 Frame Images . . . . .                                     | 231        |
| 37.5 Colors and Highlighting . . . . .                          | 232        |
| 37.6 Sectioning . . . . .                                       | 233        |
| 37.7 Excursions . . . . .                                       | 236        |
| <b>38 The Implementation</b>                                    | <b>237</b> |
| 38.1 Package Options . . . . .                                  | 237        |
| 38.2 Problems and Solutions . . . . .                           | 238        |
| 38.3 Multiple Choice Blocks . . . . .                           | 244        |
| 38.4 Including Problems . . . . .                               | 245        |
| 38.5 Reporting Metadata . . . . .                               | 246        |
| <b>39 Implementation: The hwexam Class</b>                      | <b>248</b> |
| 39.1 Class Options . . . . .                                    | 248        |
| <b>40 Implementation: The hwexam Package</b>                    | <b>250</b> |
| 40.1 Package Options . . . . .                                  | 250        |
| 40.2 Assignments . . . . .                                      | 251        |
| 40.3 Including Assignments . . . . .                            | 254        |
| 40.4 Typesetting Exams . . . . .                                | 255        |
| 40.5 Leftovers . . . . .  | 257        |

# Part I

## Manual



Boxes like this one contain implementation details that are not immediately interesting, but might be useful to know when debugging (or to avoid) errors.



Boxes like this one explain how some  $\text{\texttt{\textbackslash S T E X}}$  concept relates to the MMT/OMDoc system, philosophy or language.

# 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](#).  
sTeX is also available on CTAN and in TeXLive.
- 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 [section 3.2](#)).

- **The Mmt System** available [here](#)<sup>1</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.

- **sTeX Archives** If we only care about L<sup>A</sup>TeX 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.

- **RuSTeX** The MMT system will also set up RuSTeX for you, which is used to generate (semantically annotated) `xhtml` from tex sources. In lieu of using MMT, you can also download and use RuSTeX directly [here](#).

---

<sup>1</sup>EdNOTE: For now, we require the sTeX-branch, requiring manually compiling the MMT sources

## 2.2 A First $\text{\LaTeX}$ Document

Having set everything up, we can write a first  $\text{\LaTeX}$  document. As an example, we will use the `smglom/calculus` and `smglom/arithmetics` archives, which should be present in the designated MathHub-folder, and write a small fragment defining the *geometric series*:

**TODO:** use some  $\text{sTeX}$ -archive instead of `smglom`, use a convergence-notion that includes the limit, mark-up the theorem properly

```

1 \documentclass{article}
2 \usepackage{stex,xcolor,stexthm}
3
4 \begin{document}
5 \begin{smodule}{GeometricSeries}
6   \importmodule[smglom/calculus]{series}
7   \importmodule[smglom/arithmetics]{realarith}
8
9   \symdef{geometricSeries}[name=geometric-series]{\comp{S}}
10
11   \begin{sdefinition}[for=geometricSeries]
12     The \definame{geometricSeries} is the \symname{?series}
13     \[\defeq{\geometricSeries}{\definiens{
14       \infinitesum{\svar{n}}{1}{
15         \realdivide[frac]{1}{
16           \realpower{2}{\svar{n}}
17         }
18       }}
19     \].\]
20   \end{sdefinition}
21
22   \begin{sassertion}[name=geometricSeriesConverges,type=theorem]
23     The \symname{geometricSeries} \symname{converges} towards $1$.
24   \end{sassertion}
25 \end{smodule}
26 \end{document}

```

Compiling this document with `pdflatex` should yield the output

**Definition 0.1.** The **geometric series** is the **series**

$$S := \sum_{n=1}^{\infty} \frac{1}{2^n}.$$

**Theorem 0.2.** The **geometric series converges** towards 1.

Feel free to move your cursor over the various highlighted parts of the document – depending on your pdf viewer, this should yield some interesting (but possibly for now cryptic) information.

### Remark 2.2.1:

Note that all of the highlighting, tooltips, coloring and the environment headers come from `stexthm` – by default, the amount of additional packages loaded is kept to a minimum and all the presentations can be customized.

Let's investigate this document in detail now:

```
\begin{smodule}{GeometricSeries}
...
\end{smodule}
```

**smodule** First, we open a new *module* called `GeometricSeries`. This module is assigned a *globally unique* identifier (URI), which (depending on your pdf viewer) should pop up in a tooltip if you hover over the word **geometric series**.

```
\importmodule[smglom/calculus]{series}
\importmodule[smglom/arithmetics]{realarith}
```

**\importmodule** Next, we *import* two modules – `series` in the `smglom/calculus`-archive, and `realarith` in the `smglom/arithmetics`-archive. If we investigate these archives, we find the files `series.en.tex` and `realarith.en.tex` (respectively) in their respective **source**-folders, which contain the statements `\begin{smodule}{series}` and `\begin{smodule}{realarith}` (respectively).

The `\importmodule`-statements make all  $\text{\LaTeX}$  symbols and associated semantic macros (e.g. `\infinitesum`, `\realdive`, `\realpower`) in the desired module available. Additionally, they “export” these symbols to all further modules which include the *current* module – i.e. if in some future module we would put `\importmodule{GeometricSeries}`, we would also have `\infinitesum` etc. at our disposal.

**\usemodule** If we only want to *use* the content of some module `Foo`, e.g. in remarks or examples, but none of the symbols in our current module actually *depend* on the content of `Foo`, we can use `\usemodule` instead – like `\importmodule`, this will make the module content available, but will *not* export it to other modules.

```
\symdef{GeometricSeries}[name=geometric-series]{\comp{S}}
```

**\symdef** Next, we introduce a new *symbol* with name `geometric-series` and assign it the semantic macro `\geometricSeries`. `\symdef` also immediately assigns this symbol a *notation*, namely `S`.

**\comp** The macro `\comp` marks the `S` in the notation as a *notational component*, as opposed to e.g. arguments to `\geometricSeries`. It is the notational components that get highlighted and associated with the corresponding symbol (i.e. in this case `geometricSeries`). Since `\geometricSeries` takes no arguments, we can wrap the whole notation in a `\comp`.

```
\begin{sdefinition}[for=geometricSeries]
...
\end{sdefinition}
\begin{sassertion}[name=geometricSeriesConverges,type=theorem]
...
\end{sassertion}
```

What follows are two  $\text{\LaTeX}$ -statements (e.g. definitions, theorems, examples, proofs, ...). These are semantically marked-up variants of the usual environments, which take additional optional arguments (e.g. `for=`, `type=`, `name=`). Since many  $\text{\LaTeX}$  templates predefine environments like `definition` or `theorem` with different syntax, we use `sdefinition`, `sassertion`, `sexample` etc. instead. You can customize these environments to e.g. simply wrap around some predefined `theorem`-environment. That way, we can still use `sassertion` to provide semantic information, while being fully compatible with (and using the document presentation of) predefined environments.

In our case, the `stexthm`-package patches e.g. `\begin{sassertion}[type=theorem]` to use a `theorem`-environment defined (as usual) using `amsthm`.

The `\define{geometricSeries}` is the `\symname{?series}`

|   |  |
|---|--|
| <u><code>\symname</code></u>                                    | The <code>\symname</code> -command prints the name of a symbol, highlights it (based on customizable settings) and associates the text printed with the corresponding symbol. If you hover over the word <code>series</code> in the pdf output, you should see a tooltip showing the full URI of the symbol used.  |
| <u><code>\symref</code></u>                                     | The <code>\symname</code> -command is a special case of the more general <code>\symref</code> -command, which allows customizing the precise text associated with a symbol.  |
| <u><code>\define</code></u><br><u><code>\definiendum</code></u> | <p>The <code>sdefinition</code>-environment provides two additional macros, <code>\define</code> and <code>\definiendum</code> which behave similar to <code>\symname</code> and <code>\symref</code>, but explicitly mark the symbols as <i>being defined</i> in this environment, to allow for special highlighting.</p> <pre> \[\defeq{\geometricSeries}{\definiens{   \infinitesum{svar{n}}{1}{     \realdivide[frac]{1}{       \realpower{2}{svar{n}}     }   }} }\].\]</pre> <p>The next snippet – set in a math environment – uses several semantic macros imported from (or recursively via) <code>series</code> and <code>realarithmetics</code>, such as <code>\defeq</code>, <code>\infinitesum</code>, etc. In math mode, using a semantic macro inserts its (default) definition. A semantic macro can have several notations – in that case, we can explicitly choose a specific notation by providing its identifier as an optional argument; e.g. <code>\realdivide[frac]{a}{b}</code> will use the explicit notation named <code>frac</code> of the semantic macro <code>\realdivide</code>, which yields <math>\frac{a}{b}</math> instead of <math>a/b</math>.</p> |
| <u><code>\svar</code></u>                                       | The <code>\svar{n}</code> command marks up the <code>n</code> as a variable with name <code>n</code> and notation <code>n</code> .   |
| <u><code>\definiens</code></u>                                  | The <code>sdefinition</code> -environment additionally provides the <code>\definiens</code> -command, which allows for explicitly marking up its argument as the <i>definiens</i> of the symbol currently being defined.   |

## 2.2.1 OMDoc/xhtml Conversion

So, if we run `pdflatex` on our document, then  $\LaTeX$  yields pretty colors and tooltips<sup>1</sup>. But  $\LaTeX$  becomes a lot more powerful if we additionally convert our document to `xhtml`.

### TODO VSCode Plugin

Using `RuSTeX`, we can convert the document to `xhtml` using the command `rustex -i /path/to/file.tex -o /path/to/outfile.xhtml`. Investigating the resulting file, we notice additional semantic information resulting from our usage of semantic macros, `\symref` etc. Below is the (abbreviated) snippet inside our `\definiens` block:

```
<mrow resource="" property="stex:definiens">
  <mrow resource="...?series?infinitesum##" property="stex:OMBIND">
    <munderover displaystyle="true">
      <mo resource="...?series?infinitesum" property="stex:comp"> $\Sigma$ </mo>
      <mrow>
        <mi resource="1" property="stex:arg">n</mi>
        <mo class="rel">=</mo>
        <mi resource="2" property="stex:arg">1</mi>
      </mrow>
      <mi resource="...?series?infinitesum" property="stex:comp"> $\infty$ </mi>
    </munderover>
    <mrow resource="3" property="stex:arg">
      <mfrac resource="...?realarith?division#frac#" property="stex:OMA">
        <mi resource="1" property="stex:arg">1</mi>
        <mrow resource="2" property="stex:arg">
          <mo class="opening">(</mo>
          <msup resource="...realarith?exponentiation##" property="stex:OMA">
            <mi resource="1" property="stex:arg">2</mi>
            <mi resource="2" property="stex:arg">n</mi>
          </msup>
          <mo class="closing">)</mo>
        </mrow>
      </mfrac>
    </mrow>
  </mrow>
```

...containing all the semantic information. The MMT system can extract from this the following OPENMATH snippet:

```
<OMBIND>
  <OMID name="...?series?infinitesum"/>
  <OMV name="n"/>
  <OMLIT name="1"/>
  <OMA>
    <OMS name="...?realarith?division"/>
    <OMLIT name="1"/>
    <OMA>
      <OMS name="...realarith?exponentiation"/>
      <OMLIT name="2"/>
      <OMV name="n"/>
    </OMA>
  </OMA>
</OMBIND>
```

<sup>1</sup>...and hyperlinks for symbols, and indices, and allows reusing document fragments modularly, and...

...giving us the full semantics of the snippet, allowing for a plurality of knowledge management services – in particular when serving the `xhtml`.

**Remark 2.2.2:**

Note that the `html` when opened in a browser will look slightly different than the `pdf` when it comes to highlighting semantic content – that is because naturally `html` allows for much more powerful features than `pdf` does. Consequently, the `html` is intended to be served by a system like MMT, which can pick up on the semantic information and offer much more powerful highlighting, linking and similar features, and being customizable by *readers* rather than being prescribed by an author.

Additionally, not all browsers (most notably Chrome) support MATHML natively, and might require additional external JavaScript libraries such as MathJax to render mathematical formulas properly.

## Chapter 3

# Creating sTeX Content

We can use sTeX by simply including the package with `\usepackage{stex}`, or – primarily for individual fragments to be included in other documents – by using the sTeX document class with `\documentclass{stex}` which combines the `standalone` document class with the `stex` package.

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

**lang** (*<language>\**) Languages to load with the `babel` package.

**mathhub** (*<directory>*) MathHub folder to search for repositories – this is not necessary if the `MATHHUB` system variable is set.

**sms** (*<boolean>*) use *persisted* mode (not yet implemented).

**image** (*<boolean>*) passed on to `tikzinput`.

**debug** (*<log-prefix>\**) Logs debugging information with the given prefixes to the terminal, or all if `all` is given. Largely irrelevant for the majority of users.

### 3.1 How Knowledge is Organized in sTeX

sTeX content is organized on multiple levels:

- sTeX **archives** (see [section 3.2](#)) contain individual `.tex`-files.
- These may contain sTeX **modules**, introduced via `\begin{smodule}{ModuleName}`.
- Modules contain sTeX **symbol declarations**, introduced via `\symdecl{symbolname}`, `\symdef{symbolname}` and some other constructions. Most symbols have a *notation* that can be used via a *semantic macro* `\symbolname` generated by symbol declarations.
- sTeX **expressions** finally are built up from usages of semantic macros.

 M

 M

 T

• sTeX archives are simultaneously MMT archives, and the same directory structure is consequently used.

• sTeX modules correspond to OMDoc/MMT *theories*. `\importmodules` (and



similar constructions) induce an MMT **includes** and other *theory morphisms*, thus giving rise to a *theory graph* in the OMDoc sense.

- Symbol declarations induce OMDoc/MMT *constants*, with optional (formal) *type* and *definiens* components.
- Finally,  $\text{\texttt{\textit{S}T\textit{E}X}}$  expressions are converted to OMDoc/MMT terms, which use the syntax of OPENMATH.

## 3.2 $\text{\texttt{\textit{S}T\textit{E}X}}$ Archives

### 3.2.1 The Local MathHub-Directory

$\text{\texttt{\textit{S}T\textit{E}X}}$   $\text{\texttt{\textit{usemodule}}}$ ,  $\text{\texttt{\textit{importmodule}}}$ ,  $\text{\texttt{\textit{inputref}}}$  etc. allow for including content modularly without having to specify absolute paths, which would differ between users and machines. Instead,  $\text{\texttt{\textit{S}T\textit{E}X}}$  uses *archives* that determine the global namespaces for symbols and statements and make it possible for  $\text{\texttt{\textit{S}T\textit{E}X}}$  to find content referenced via such URIs.

All  $\text{\texttt{\textit{S}T\textit{E}X}}$  archives need to exist in the local MathHub-directory.  $\text{\texttt{\textit{S}T\textit{E}X}}$  knows where this folder is via one of three means:

1. If the  $\text{\texttt{\textit{S}T\textit{E}X}}$  package is loaded with the option  $\text{\texttt{mathhub=/path/to/mathhub}}$ , then  $\text{\texttt{\textit{S}T\textit{E}X}}$  will consider  $\text{\texttt{/path/to/mathhub}}$  as the local MathHub-directory.
2. If the  $\text{\texttt{mathhub}}$  package option is *not* set, but the macro  $\text{\texttt{\textit{mathhub}}}$  exists when the  $\text{\texttt{\textit{S}T\textit{E}X}}$ -package is loaded, then this macro is assumed to point to the local MathHub-directory; i.e.  $\text{\texttt{\textit{def}\textit{mathhub}\{ /path/to/mathhub \}\textit{usepackage}\{stex\}}}$  will set the MathHub-directory as  $\text{\texttt{path/to/mathhub}}$ .
3. Otherwise,  $\text{\texttt{\textit{S}T\textit{E}X}}$  will attempt to retrieve the system variable  $\text{\texttt{MATHHUB}}$ , 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.

### 3.2.2 The Structure of $\text{\texttt{\textit{S}T\textit{E}X}}$ Archives

An  $\text{\texttt{\textit{S}T\textit{E}X}}$  archive **group/name** needs to be stored in the directory  $\text{\texttt{/path/to/mathhub/group/name}}$ ; e.g. assuming your local MathHub-directory is set as  $\text{\texttt{/user/foo/MathHub}}$ , then in order for the **smglom/calculus**-archive to be found by the  $\text{\texttt{\textit{S}T\textit{E}X}}$  system, it needs to be in  $\text{\texttt{/user/foo/MathHub/smglom/calculus}}$ .

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  $\text{\texttt{\textit{S}T\textit{E}X}}$  will look for files included via  $\text{\texttt{\textit{libinput}}}$ .

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  $\text{\texttt{\textit{libinput}}}$  from all tex files in any archive in the **group/\***-group.



We recommend this additional directory structure in the `source`-folder of an  $\text{\TeX}$  archive:

- `/source/mod/` – individual  $\text{\TeX}$  modules, containing symbol declarations, notations, and `\begin{paragraph}` [`type=symdoc,for=...`] environments for “encyclopedic” symbol documentations
- `/source/def/` – definitions
- `/source/ex/` – examples
- `/source/thm/` – theorems, lemmata and proofs; preferably proofs in separate files to allow for multiple proofs for the same statement
- `/source/snip/` – individual text snippets such as remarks, explanations etc.
- `/source/frag/` – individual document fragments, ideally only `\inputref`ing snippets, definitions, examples etc. in some desirable order
- `/source/tikz/` – tikz images, as individual `.tex`-files
- `/source/pic/` – image files.

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

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

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

### 3.2.4 Using Files in $\text{\TeX}$ Archives Directly

Several macros provided by  $\text{\TeX}$  allow for directly including files in repositories. These are:

---

|                            |   |
|----------------------------|---|
| $\backslash\text{mhinput}$ | $\backslash\text{mhinput}$ [Some/Archive]{some/file} directly inputs the file some/file in the source-folder of Some/Archive. |
|----------------------------|---|

---

|                             |   |
|-----------------------------|---|
| $\backslash\text{inputref}$ | $\backslash\text{inputref}$ [Some/Archive]{some/file} behaves like $\backslash\text{mhinput}$ , but wraps the input in a $\backslash\text{begingroup} \dots \backslash\text{endgroup}$ . When converting to $\text{xhtml}$ , the file is not input at all, and instead an $\text{html}$ -annotation is inserted that references the file.<br>In the majority of cases $\backslash\text{inputref}$ is likely to be preferred over $\backslash\text{mhinput}$ . |
|-----------------------------|---|

---

|                            |   |
|----------------------------|---|
| $\backslash\text{ifinput}$ | Both $\backslash\text{mhinput}$ and $\backslash\text{inputref}$ set $\backslash\text{ifinput}$ to “true” during input. This allows for selectively including e.g. bibliographies only if the current file is not being currently included in a larger document. |
|----------------------------|---|

---

|                                     |   |
|-------------------------------------|---|
| $\backslash\text{addmhbibresource}$ | $\backslash\text{addmhbibresource}$ [Some/Archive]{some/file} searches for a file like $\backslash\text{mhinput}$ does, but calls $\backslash\text{addbibresource}$ to the result and looks for the file in the archive root directory directly, rather than the <code>source</code> directory. |
|-------------------------------------|---|

---

|                             |   |
|-----------------------------|---|
| $\backslash\text{libinput}$ | $\backslash\text{libinput}$ {some/file} searches for a file some/file in <ul style="list-style-type: none"><li>• the <code>lib</code>-directory of the current archive, and</li><li>• the <code>lib</code>-directory of a <code>meta-inf</code>-archive in (any of) the archive groups containing the current archive</li></ul> and include all found files in reverse order; e.g. $\backslash\text{libinput}\{\text{preamble}\}$ in a <code>.tex</code> -file in <code>smglom/calculus</code> will <i>first</i> input <code>../smglom/meta-inf/lib/preamble.tex</code> and then <code>../smglom/calculus/lib/preamble.tex</code> .<br>Will throw an error if <i>no</i> candidate for some/file is found. |
|-----------------------------|---|

---

|                                  |  |
|----------------------------------|--|
| $\backslash\text{libusepackage}$ | $\backslash\text{libusepackage}$ [package-options]{some/file} searches for a file some/file.sty in the same way that $\backslash\text{libinput}$ does, but will call $\backslash\text{usepackage}$ [package-options]{path/to/some/file} instead of $\backslash\text{input}$ .<br>Will throw an error if not <i>exactly one</i> candidate for some/file is found. |
|----------------------------------|--|

### Remark 3.2.1:

A good practice is to have individual  $\text{\TeX}$  fragments follow basically this document frame:

```
1 \documentclass{stex}
2 \libinput{preamble}
3 \begin{document}
4   ...
5   \ifinputref \else \libinput{postamble} \fi
6 \end{document}
```

Then the `preamble.tex` files can take care of loading the generally required packages, setting presentation customizations etc. (per archive or archive group or both), and `postamble.tex` can e.g. print the bibliography, index etc.

## 3.3 Module, Symbol and Notation Declarations

### 3.3.1 The `smodule`-Environment

`smodule` A new module is declared using the basic syntax

```
\begin{smodule}[options]{ModuleName}...\end{smodule}.
```

A module is required to declare any new formal content such as symbols or notations (but not variables, which may be introduced anywhere).

The `smodule`-environment takes several optional arguments, all of which are optional:

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

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

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

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

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

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

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

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

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

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

$\hookrightarrow$   
 $\rightarrow$   
 $\rightsquigarrow$

An  $\text{\TeX}$  module corresponds to an MMT/OMDoc *theory*.

By default, opening a module will produce no output whatsoever, e.g.:

### Example 1

Input:

```

1 \begin{smodule}[title={This is Some Module}]{SomeModule}
2   Hello World
3 \end{smodule}

```

Output:

```

Hello World

```

#### \stexpatchmodule

We can customize this behavior either for all modules or only for modules with a specific type using the command `\stexpatchmodule[optional-type]{begin-code}{end-code}`. Some optional parameters are then available in `\smodule*`-macros, specifically `\smoduletitle`, `\smoduletype` and `\smoduleid`. For example:

### Example 2

Input:

```

1 \stexpatchmodule[display]
2   {\textbf{Module (\smoduletitle)}\par}
3   {\par\noindent\textbf{End of Module (\smoduletitle)}}
4
5 \begin{smodule}[type=display,title={Some New Module}]{SomeModule2}
6   Hello World
7 \end{smodule}

```

Output:

```

Module (Some New Module)
  Hello World
End of Module (Some New Module)

```

## 3.3.2 Declaring New Symbols and Notations

Inside an `smodule` environment, we can declare new  $\text{\TeX}$  symbols.

#### \symdecl

The most basic command for doing so is using `\symdecl{symbolname}`. This introduces a new symbol with name `symbolname`, arity 0 and semantic macro `\symbolname`.

The starred variant `\symdecl*{symbolname}` will declare a symbol, but not introduce a semantic macro. If we don't want to supply a notation (for example to introduce concepts like “abelian”, which is not something that has a notation), the starred variant is likely to be what we want.

$\hookrightarrow M \rightarrow$  \symsdecl introduces a new OMDoc/MMT constant in the current module (=OM-Doc/MMT theory).  
 $\hookrightarrow M \rightarrow$   
 $\rightsquigarrow T \rightsquigarrow$

Without a semantic macro or a notation, the only meaningful way to reference a symbol is via `\symref`, `\symname` etc.

### Example 3

Input:

```
1 \symdecl*{foo}
2 Given a \symname{foo}, we can...
```

Output:

Given a `foo`, we can...

Obviously, most semantic macros should take actual *arguments*, implying that the symbol we introduce is an *operator* or *function*. We can let `\symdecl` know the *arity* (i.e. number of arguments) of a symbol like this:

### Example 4

Input:

```
1 \symdecl{binarysymbol}[args=2]
2 \symref{binarysymbol}{this} is a symbol taking two arguments.
```

Output:

`this` is a symbol taking two arguments.

•

\notation

In that case, we probably want to supply a notation as well, in which case we can finally actually use the semantic macro in math mode. We can do so using the `\notation` command, like this:

### Example 5

Input:

```
1 \notation{binarysymbol}{\text{First: }#1\text{; Second: }#2}  
2 $\text{binarysymbol}{a}{b}$
```

Output:

First:  $a$ ; Second:  $b$

`\comp`

Unfortunately, we have no highlighting whatsoever now. That is because we need to tell  $\TeX$  explicitly which parts of the notation are *notation components* which *should* be highlighted. We can do so with the `\comp` command.

We can introduce a new notation `highlight` for `\binarysymbol` that fixes this flaw, which we can subsequently use with `\binarysymbol[highlight]`:

#### Example 6

Input:

```
1 \notation{binarysymbol}[highlight]
2 {\comp{\text{First: }}#1\comp{\text{; Second: }}#2}
3 $\binarysymbol[highlight]{a}{b}$
```

Output:

First:  $a$ ; Second:  $b$



Ideally, `\comp` would not be necessary: Everything in a notation that is *not* an argument should be a notation component. Unfortunately, it is computationally expensive to determine where an argument begins and ends, and the argument markers `#n` may themselves be nested in other macro applications or  $\TeX$  groups, making it ultimately almost impossible to determine them automatically while also remaining compatible with arbitrary highlighting customizations (such as tooltips, hyperlinks, colors) that users might employ, and that are ultimately invoked by `\comp`.



Note that it is required that

1. the argument markers `#n` never occur inside a `\comp`, and
2. no semantic arguments may ever occur inside a notation.

Both criteria are not just required for technical reasons, but conceptionally meaningful:

The underlying principle is that the arguments to a semantic macro represent *arguments to the mathematical operation* represented by a symbol. For example, a semantic macro `\addition{a}{b}` taking two arguments would represent *the actual addition of (mathematical objects) a and b*. It should therefore be impossible for  $a$  or  $b$  to be part of a notation component of `\addition`.

Similarly, a semantic macro can not conceptually be part of the notation of `\addition`, since a semantic macro represents a *distinct mathematical concept* with *its own semantics*, whereas notations are syntactic representations of the very symbol to which the notation belongs.

If you want an argument to a semantic macro to be a purely syntactic parameter, then you are likely somewhat confused with respect to the distinction between the precise *syntax* and *semantics* of the symbol you are trying to declare (which happens quite often even to experienced  $\TeX$  users), and might want to give those another thought - quite likely, the macro you aim to implement does not actually represent a semantically meaningful mathematical concept, and you will want to



use `\def` and similar native L<sup>A</sup>T<sub>E</sub>X macro definitions rather than semantic macros.

`\symdef`

In the vast majority of cases where a symbol declaration should come with a semantic macro, we will want to supply a notation immediately. For that reason, the `\symdef` commands combines the functionality of both `\symdecl` and `\notation` with the optional arguments of both:

#### Example 7

Input:

```
1 \symdef{newbinarysymbol}[hl,args=2]
2   {\comp{\text{1.: }}#1\comp{\text{; 2.: }}#2}
3 $\newbinarysymbol{a}{b}$
```

Output:

1.: a; 2.: b

We just both declared a new symbol `newbinarysymbol` and immediately provided it with a notation with identifier `hl`. Since `hl` is the *first* (and so far, only) notation supplied for `newbinarysymbol`, using `\newbinarysymbol` without optional argument defaults to this notation.

### 3.4 Using Semantic Macros and Referencing Symbols

TODO: terms documentation

TODO: references documentation

TODO: inheritance documentation

### 3.5 Advanced Structuring Mechanisms

Given modules:

#### Example 8

Input:

```

1 \begin{smodule}{magma}
2   \symdef{universe}{\comp{\mathcal U}}
3   \symdef{operation}[args=2,op=\circ]{#1 \comp\circ #2}
4 \end{smodule}
5 \begin{smodule}{monoid}
6   \importmodule{magma}
7   \symdef{unit}{\comp e}
8 \end{smodule}
9 \begin{smodule}{group}
10  \importmodule{monoid}
11  \symdef{inverse}[args=1]{#1\comp{-1}}
12 \end{smodule}

```

Output:

.

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 9

Input:

```

1 \begin{smodule}{ring}
2   \begin{copymodule}{group}{addition}
3     \renamedekl[name=universe]{universe}{runiverse}
4     \renamedekl[name=plus]{operation}{rplus}
5     \renamedekl[name=zero]{unit}{rzero}
6     \renamedekl[name=uminus]{inverse}{ruminus}
7   \end{copymodule}
8   \notation*{rplus}[plus,op=+,prec=60]{#1 \comp+ #2}
9   \notation*{rzero}[zero]{\comp0}
10  \notation*{ruminus}[uminus,op=-]{\comp- #1}
11  \begin{copymodule}{monoid}{multiplication}
12    \assign{universe}{\runiverse}
13    \renamedekl[name=times]{operation}{rtimes}
14    \renamedekl[name=one]{unit}{rone}
15  \end{copymodule}
16  \notation*{rtimes}[cdot,op=\cdot,prec=50]{#1 \comp\cdot #2}
17  \notation*{rone}[one]{\comp1}
18  Test: $\rtimes a\{rplus c\{rtimes de\}}$
19 \end{smodule}

```

Output:

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

TODO: explain donotclone

#### Example 10

Input:



```

1 \begin{smodule}{int}
2   \symdef{Integers}{\comp{\mathbb Z}}
3   \symdef{plus}[args=2,op=+]{#1 \comp+ #2}
4   \symdef{zero}{\comp0}
5   \symdef{uminus}[args=1,op=-]{\comp-#1}
6
7   \begin{interpretmodule}{group}{intisgroup}
8     \assign{universe}{\Integers}
9     \assign{operation}{\plus!}
10    \assign{unit}{\zero}
11    \assign{inverse}{\uminus!}
12  \end{interpretmodule}
13 \end{smodule}

```

Output:

### 3.6 Primitive Symbols (The sTeX Metatheory)

TODO: metatheory documentation

## Chapter 4

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

TODO: statements documentation  
TODO: sproofs documentation

## Chapter 5

# Additional Packages

TODO: tikzinput documentation

### 5.1 Modular Document Structuring

TODO: document-structure documentation

### 5.2 Slides and Course Notes

TODO: notesslides documentation

### 5.3 Homework, Problems and Exams

TODO: problem documentation

TODO: hwexam documentation

# Chapter 6

## Stuff

---

`\sTeX`  
`\stex`

---

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

### 6.0.1 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.

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

#### Example 11

Input:

```
1 \symdecl{mult}[args=2]
2 \notation{mult}{#1 #2}
3 $\mult{a}{b}$
```

Output:

$ab$

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

$$\text{\code{\symdef{mult}[args=2]{#1 #2}}}$$

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

### Example 12

Input:

```
1 \notation{mult}[cdot]{#1 \comp{\cdot} #2}
2 \notation{mult}[times]{#1 \comp{\times} #2}
3 $\mult[cdot]{a}{b}$ and $\mult[times]{a}{b}$
```

Output:

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

Not using an explicit option with a semantic macro yields the first declared notation, unless changed<sup>2</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 13

Input:

```
1 $\mult*{\arg{a}\comp{\ast}\arg{b}}$ is the
2 \mult{\comp{product of} \arg{$a$} \comp{and} \arg{$b$}}
```

Output:

$a * b$  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 14

Input:

```
1 \mult{\comp{Multiplying} \arg*{$\mult{a}{b}$} again by \arg{$b$}} yields...
```

Output:

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

<sup>2</sup>EdNOTE: TODO

### Example 15

Input:

```

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

```

Output:

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

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 16

Input:

```

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

```

Output:

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

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

### Example 17

Input:

```

1 \mult!{\comp{Multiplication}} (denoted by  $\text{\mult!}\cdot$ ) is defined by...

```

Output:

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

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 18

Input:

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

Output:

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

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 19

Input:

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

Output:

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

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

### 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:

$$\text{\notation{foo}}[\text{prec}=200;500\text{x}600]\{ \#1 \text{\comp{+}} \#2 \}$$

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:

### Example 20

Input:

```
1 \notation{plus}[prec=100]{#1 \comp{+} #2}
2 \notation{times}[prec=50]{#1 \comp{\cdot} #2}
3 $\plus{a}{\times{b}{c}}$ and $\times{a}{\plus{b}{c}}$
```

Output:

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

<sup>3</sup>EdNOTE: what about e.g.  $\int \int \int f \, dx \, dy \, dz$ ?

<sup>4</sup>EdNOTE: “decompose” a-type arguments into fixed-arity operators?



## 6.0.2 Archives and Imports

### Namespaces

Ideally,  $\text{\S\TeX}$  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,  $\text{\TeX}$  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  $\text{\S\TeX}$  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  $\text{\begin{module}\{Foo\}}$  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>2</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  $\text{\importmodule}$ :

- $\text{\importmodule}\{Foo\}$  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  $\text{\importmodule}\{some/path?Foo\}$  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,  $\text{\importmodule}[Some/Archive]\{some/path?Foo\}$  is resolved like the previous cases, but relative to the archive `Some/Archive` in the mathhub-directory.

---

<sup>2</sup>which is internally attached to the module name instead, but a user need not worry about that.

- 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.

## Part II

# Documentation

# Chapter 7

## sTeX-Basics

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

### 7.1 Macros and Environments

---

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

---

---

|                             |  |
|-----------------------------|--|
| <code>\stex_debug:nn</code> | <code>\stex_debug:nn {&lt;log-prefix&gt;} {&lt;message&gt;}</code> |
|-----------------------------|--|

---

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

#### 7.1.1 HTML Annotations

---

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

---

---

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

---

---

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

---

---

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

---

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

---

|   |   |
|---|---|
| <code>\stex_annotate:nnn</code>           | <code>\stex_annotate:nnn {⟨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> . |
|--------------------------------|---|

### 7.1.2 Babel Languages

---

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

---

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

### 7.1.3 Auxiliary Methods

---

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

---

## Chapter 8

# STEX-MathHub

This sub package provides code for handling ST<sub>E</sub>X archives, files, file paths and related methods.

### 8.1 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. |
|--------------------------------|---|

---

#### 8.1.1 Files, Paths, URIs

---

|  |  |
|--|--|
| <code>\stex_path_from_string:Nn</code> | <code>\stex_path_from_string:Nn</code> $\langle path-variable \rangle$ $\{\langle string \rangle\}$<br>turns the $\langle string \rangle$ into a path by splitting it at <code>/</code> -characters and stores the result in $\langle path-variable \rangle$ . Also applies <code>\stex_path_canonicalize:N</code> . |
|--|--|

---

---

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

---

---

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

---

---

|   |   |
|---|---|
| <code>\stex_path_if_absolute_p:N</code> $\star$<br><code>\stex_path_if_absolute:N<math>\underline{T</math></code> $\star$ | Checks whether the path provided is <i>absolute</i> , i.e. starts with an empty segment |
|---|---|

---

---

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

---

---

|                                      |  |
|--------------------------------------|--|
| <code>\g_stex_currentfile_seq</code> | The file being currently processed (respecting <code>\input</code> etc.) |
|--------------------------------------|--|

---



---

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

---

### 8.1.2 MathHub Archives

---

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

---

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

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

---

|  |
|--|
| <code>\l_stex_current_repository_prop</code> |
|--|

---

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

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

---

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

---

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

---

|   |  |
|---|--|
| <code>\stex_require_repository:n</code> | Calls <code>\__stex_mathhub_do_manifest:n</code> iff the corresponding archive property list does not already exist, and adds a corresponding definition to the <code>.sms</code> -file. |
|---|--|

---



---

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

---

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

### 8.1.3 Using Content in Archives

|   |  |
|---|--|
| <hr/> <hr/> <code>\mhp</code> <hr/>   | <code>\mhp{<i>archive-ID</i>}{<i>filename</i>}</code>  |
|   | Expands to the full path of file <i>filename</i> in repository <i>archive-ID</i> . Does not check whether the file or the repository exist.  |
| <hr/> <hr/> <code>\inputref</code><br><code>\mhinput</code> <hr/>                     | <code>\inputref[<i>archive-ID</i>]{<i>filename</i>}</code><br>Both <code>\input</code> the file <i>filename</i> in archive <i>archive-ID</i> (relative to the <code>source-</code> subdirectory). <code>\mhinput</code> does so directly. <code>\inputref</code> does so within an <code>\begingroup... \endgroup-</code> block, and skips it in <code>html-mode</code> , inserting a <i>reference</i> to the file instead.<br>Both also set <code>\ifinputref</code> to true.   |
| <hr/> <hr/> <code>\addmhbibresource</code> <hr/>                                      | <code>\inputref[<i>archive-ID</i>]{<i>filename</i>}</code><br>Adds a <code>.bib</code> -file <i>filename</i> in archive <i>archive-ID</i> (relative to the top-directory of the archive!).   |
| <hr/> <hr/> <code>\libinput</code> <hr/>  | <code>\libinput{<i>filename</i>}</code><br>Inputs <i>filename.tex</i> from the <code>lib</code> folders in the current archive and the <code>meta-inf-</code> archive of the current archive group(s) (if existent) in descending order. Throws an error if no file by that name exists in any of the relevant <code>lib</code> -folders.  |
| <hr/> <hr/> <code>\libusepackage</code> <hr/>   | <code>\libusepackage[<i>args</i>]{<i>filename</i>}</code><br>Like <code>\libinput</code> , but looks for <code>.sty</code> -files and calls <code>\usepackage[<i>meta</i>{<i>args</i>}]<i>Arg</i>{<i>filename</i>}</code> instead of <code>\input</code> .<br>Throws an error, if none or more than one suitable package file is found.  |
| <hr/> <hr/> <code>\mhgraphics</code><br><code>\cmhgraphics</code> <hr/>               | <i>If</i> the <code>graphicx</code> package is loaded, these macros are defined at <code>\begin{document}</code> .<br><code>\mhgraphics</code> takes the same arguments as <code>\includegraphics</code> , with the additional optional key <code>mhrepos</code> . It then resolves the file path in <code>\mhgraphics[mhrepos=Foo/Bar]{foo/bar.png}</code> relative to the <code>source-</code> folder of the <code>Foo/Bar</code> -archive.<br><code>\cmhgraphics</code> additional wraps the image in a <code>center</code> -environment. |
| <hr/> <hr/> <code>\lstinputmhlisting</code><br><code>\clstinputmhlisting</code> <hr/> | Like <code>\mhgraphics</code> , but only defined if the <code>listings</code> -package is loaded, and with <code>\lstinputlisting</code> instead of <code>\includegraphics</code> .  |



## Chapter 9

# STEX-References

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

### 9.1 Macros and Environments

---

**\STEXreftitle**

---

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

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

---

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

---

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

---

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

---

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

---

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

---

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

---

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

---

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

#### 9.1.1 Setting Reference Targets

---

**\stex\_ref\_new\_doc\_target:n**

---

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

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

---

**\stex\_ref\_new\_sym\_target:n**

---

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

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

### 9.1.2 Using References

---

`\sref`    `\sref[<opt-args>]{<id>}`

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

---

`\srefsym`    `\srefsym[<opt-args>]{<symbol>}`

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

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

---

`\srefsymuri`    `\srefsymuri{<URI>}{<text>}`

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

# Chapter 10

## STEX-Modules

This sub package contains code related to Modules

### 10.1 Macros and Environments

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

---

---

`\c_stex_module_<URI>_prop`

A property list with the following fields:

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

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

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

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

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

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

**meta** the metatheory of the module.

---

---

`\c_stex_module_<URI>_code`

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

---

---

`\c_stex_module_<URI>_constants`

The names of all constants declared in the module

---

---

`\c_stex_module_<URI>_constants`

The full URIs of all modules imported in this module

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

---

`\stex_modules_current_namespace:`

---

Computes the current namespace as follows:

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

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

### 10.1.1 The `smodule` environment

`module` `\begin{module}[\langle options \rangle]{\langle name \rangle}`  
 Opens a new module with name `\langle name \rangle`. Options are:

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

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

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

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

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

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

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

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

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

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

---

`\stex_module_setup:nn` `\stex_module_setup:nn{\langle params \rangle}{\langle name \rangle}`

---

Sets up a new module with name `\langle name \rangle` and optional parameters `\langle params \rangle`. In particular, sets `\l_stex_current_module_str` appropriately.

---

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

---

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

---

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

---

Attempts to find a module whose URI ends with `\langle fragment \rangle` in the current scope and passes the full URI on to `\stex_invoke_module:n`.

---

`\stex_invoke_module:n` Invoked by `\STEXModule`. Needs to be followed either by `!\macro` or `?{\langle symbolname \rangle}`. In the first case, it stores the full URI in `\macro`; in the second case, it invokes the symbol `\langle symbolname \rangle` in the selected module.

---

---

---

`\stex_activate_module:n`

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

# Chapter 11

## STEX-Module Inheritance

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

### 11.1 Macros and Environments

#### 11.1.1 SMS Mode

“SMS Mode” is used when loading modules from external tex files. It deactivates any output and ignores all T<sub>E</sub>X commands not explicitly allowed via the following lists – all of which either declare module content or are needed in order to declare module content:

---

`\g_stex_smsmode_allowedmacros_tl`

---

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

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

---

`\g_stex_smsmode_allowedmacros_escape_tl`

---

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

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

---

`\g_stex_smsmode_allowedenvs_seq`

---

The names of environments that should be allowed in SMS mode. The corresponding `\begin`-statements are treated like the macros in `\g_stex_smsmode_allowedmacros_escape_tl`, so `\stex_smsmode_do:` needs to be the last token in the `\begin`-code. Since `\end`-statements take no arguments anyway, those are called directly and sms mode continues afterwards.

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

---

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

---

Tests whether SMS mode is currently active.

---

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

---

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

---

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

---

### 11.1.2 Imports and Inheritance

---

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

---

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

---

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

---

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



---

**`\stex_import_module_uri:nn`**

---

**`\stex_import_module_uri:nn`**  $\{\langle archive-ID \rangle\}$   $\{\langle module-path \rangle\}$ 

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

If  $\langle archive-ID \rangle$  is empty, it is automatically set to the ID of the current archive (if one exists).

1. If  $\langle archive-ID \rangle$  is empty:

- (a) If  $\langle path \rangle$  is empty, then  $\langle name \rangle$  must have been declared earlier in the same file and retrievable from `\g_stex_modules_in_file_seq`, or a file with name  $\langle name \rangle . \langle lang \rangle . \text{tex}$  must exist in the same folder, containing a module  $\langle name \rangle$ .

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

- (b) If  $\langle path \rangle$  is not empty, it must point to the relative path of the containing file as well as the namespace.

2. Otherwise:

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

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

- (b) If  $\langle path \rangle$  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.

---

**`\l_stex_import_name_str`**  
**`\l_stex_import_archive_str`**  
**`\l_stex_import_path_str`**  
**`\l_stex_import_ns_str`**

---

stores the result in these four variables.

---

**`\stex_import_require_module:nnnn`**  $\{\langle ns \rangle\}$   $\{\langle archive-ID \rangle\}$   $\{\langle path \rangle\}$   $\{\langle name \rangle\}$ 

---

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

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

# Chapter 12

## STEX-Symbols

Code related to symbol declarations and notations

### 12.1 Macros and Environments

---

|                       |  |
|-----------------------|--|
| <code>\symdecl</code> | <code>\symdecl{<i>macroname</i>}[<i>args</i>]</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, representing a *type*. Not used by `STEX`, but passed on to MMT for semantic services.
- **def**: An (ideally semantic) term, representing a *definiens*. Not used by `STEX`, 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{plus}[args=ii]` 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{plus}[args=a]` allows for `\plus{2,2,2}`.
  - b** a *variable* argument. Is treated by `STEX` 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{forall}[args=bi]` allows for `\forall{x \in \mathbb{N}}{x \geq 0}`.

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

# Chapter 13

## STEX-Terms

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

### 13.1 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> .  |

|  |  |
|--|--|
| <hr/> <hr/>  |  |
| <code>\infprec</code><br><code>\neginfprec</code>  | Maximal and minimal notation precedences.  |
| <hr/> <hr/>  |  |
| <code>\dobrackets</code>   | <code>\dobrackets {⟨body⟩}</code>  |
|  | Puts $\langle body \rangle$ in parentheses; scaled if in display mode unscaled otherwise. Uses the current $\text{\SIX}$ brackets (by default ( and )), which can be changed temporarily using <code>\withbrackets</code> .  |
| <hr/> <hr/>  |  |
| <code>\withbrackets</code>   | <code>\withbrackets ⟨left⟩ ⟨right⟩ {⟨body⟩}</code>   |
|  | Temporarily (i.e. within $\langle body \rangle$ ) sets the brackets used by $\text{\SIX}$ for automated bracketing (by default ( and )) to $\langle left \rangle$ and $\langle right \rangle$ .<br>Note that $\langle left \rangle$ and $\langle right \rangle$ need to be allowed after <code>\left</code> and <code>\right</code> in display-mode.   |
| <hr/> <hr/>  |  |
| <code>\stex_term_custom:nn</code>  | <code>\stex_term_custom:nn{⟨URI⟩}{⟨args⟩}</code>   |
|  | Implements custom one-time notation. Invoked by <code>\stex_invoke_symbol:n</code> in text mode, or if followed by <code>*</code> in math mode, or whenever followed by <code>!</code> .   |
| <hr/> <hr/>  |  |
| <code>\stex_highlight_term:nn</code>   | <code>\stex_highlight_term:nn{⟨URI⟩}{⟨args⟩}</code>  |
|  | Establishes a context for <code>\comp</code> . Stores the URI in a variable so that <code>\comp</code> knows which symbol governs the current notation.  |
| <hr/> <hr/>  |  |
| <code>\comp</code><br><code>\compemph</code><br><code>\compemph@uri</code><br><code>\defemph</code><br><code>\defemph@uri</code><br><code>\symrefemph</code><br><code>\symrefemph@uri</code><br><code>\varemp</code><br><code>\varemp@uri</code> | <code>\comp{⟨args⟩}</code><br>Marks $\langle args \rangle$ as a notation component of the current symbol for highlighting, linking, etc.<br>The precise behavior is governed by <code>\@comp</code> , which takes as additional argument the URI of the current symbol. By default, <code>\@comp</code> adds the URI as a PDF tooltip and colors the highlighted part in blue.<br><code>\@defemph</code> behaves like <code>\@comp</code> , and can be similarly redefined, but marks an expression as <i>definiendum</i> (used by <code>\definiendum</code> ) |
| <hr/> <hr/>  |  |
| <code>\STEXinvisible</code>  | Exports its argument as OMDoc (invisible), but does not produce PDF output. Useful e.g. for semantic macros that take arguments that are not part of the symbolic notation.  |
| <hr/> <hr/>  |  |
| <code>\ellipses</code>   | TODO   |

## Chapter 14

# TeX-Structural Features

Code related to structural features

### 14.1 Macros and Environments

#### 14.1.1 Structures

`mathstructure` TODO

## Chapter 15

# sTeX-Statements

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

### 15.1 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 16

# **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

## 16.1 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]
  {We prove that  $\sum_{i=1}^n (2i-1) = n^2$  by induction over  $n$ }
  \begin{spfcases}{For the induction we have to consider the following cases:}
    \begin{spfcase}{ $n=1$ }
      \begin{spfstep}[type=inline] then we compute  $1=1^2$ \end{spfstep}
    \end{spfcase}
    \begin{spfcase}{ $n=2$ }
      \begin{sproofcomment}[type=inline]
        This case is not really necessary, but we do it for the
        fun of it (and to get more intuition).
      \end{sproofcomment}
      \begin{spfstep}[type=inline] 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}
  \end{spfcases}
  \begin{spfstep}[type=conclusion]
    We have considered all the cases, so we have proven the assertion.
  \end{spfstep}
\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>5</sup>

<sup>5</sup>EdNOTE: talk a bit more about proofs and their structure,... maybe copy from OMDoc spec.

## 16.2 The User Interface

### 16.2.1 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).

### 16.2.2 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.

### 16.2.3 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  $n$

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

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

16.2.4 Proof Structure

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

16.2.5 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 <code>sproof</code> package provides the <code>\sproofend</code> macro for this. If a different symbol for the proof end is to be used (e.g. <i>q.e.d</i> ), then this can be obtained by specifying it using the |
| \sProofEndSymbol   | <code>\sProofEndSymbol</code> configuration macro (e.g. by specifying <code>\sProofEndSymbol{q.e.d}</code> ).   |
| Some of the proof structuring macros above will insert proof end symbols for subproofs, 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 <code>proofend={}</code> in them or use use <code>\sProofEndSymbol{}</code> . |   |

16.2.6 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>6</sup>. The proof step labels can be customized via the `\pstlabelstyle` macro:

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

Figure 1: Configuration Hooks for Semantic Proof Markup

|                |   |
|----------------|---|
| \pstlabelstyle | <code>\pstlabelstyle{&lt;style&gt;}</code> sets the style; see Figure ?? for an overview of styles. Package writers can add additional styles by adding a macro <code>\pst@make@label@&lt;style&gt;</code> that takes |
|----------------|---|

<sup>6</sup>EdNOTE: we might want to develop an extension `sproof-babel` in the future.

two arguments: a comma-separated list of ordinals that make up the prefix and the current ordinal. Note that comma-separated lists can be conveniently iterated over by the  $\text{\LaTeX}$  `\@for...:=...\do{...}` macro; see Figure ?? for examples.

## 16.3 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  $\text{\TeX}$  issue tracker at [\[sTeX\]](#).

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  $\text{\LaTeX}$  `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 17

# 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.

### 17.1 Symbols

**Part III**  
**Extensions**



## Chapter 18

# Tikzinput

### 18.1 Macros and Environments

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

## Chapter 19

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

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

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

### 19.1 Introduction

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

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

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

source and the formatter does the copying during document formatting/presentation.<sup>7</sup>

## 19.2 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`.

### 19.2.1 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.

### 19.2.2 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>3</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>sfragment</code>     | element resulting from the L <sup>A</sup> T <sub>E</sub> XML transformation.  |
|                            | The structure of the document is given by the <code>omgroup</code> environment just like in OM-                               |
|                            | DOC. In the L <sup>A</sup> T <sub>E</sub> X route, the <code>omgroup</code> environment is flexibly mapped to sectioning com- |
|                            | mands, inducing the proper sectioning level from the nesting of <code>omgroup</code> environments.                            |
|                            | Correspondingly, the <code>omgroup</code> environment takes an optional key/value argument for                                |
|                            | metadata followed by a regular argument for the (section) title of the <code>omgroup</code> . The op-                         |
| <code>id</code>            | tional metadata argument has the keys <code>id</code> for an identifier, <code>creators</code> and <code>contributors</code>  |
| <code>creators</code>      | for the Dublin Core metadata [DCM03]; see [Koh20a] for details of the format. The   |
| <code>contributors</code>  | <code>short</code> allows to give a short title for the generated section. If the title contains semantic                     |
| <code>short</code>         | macros, they need to be protected by <code>\protect</code> , and we need to give the <code>loadmodules</code>                 |
| <code>loadmodules</code>   | key it needs no value. For instance we would have   |

```

\begin{smodule}{foo}
\symdef{bar}{B^a_r}
...
\begin{sfragment}[id=sec.barderviv,loadmodules]{Introducing $\protect\bar$ Derivation

```

<sup>7</sup>EdNOTE: integrate with latexml's XMRef in the Math mode.

<sup>3</sup>We cannot patch the document environment to accept an optional argument, since other packages we load already do; pity.

`blindfragment`

$\text{\TeX}$  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>4</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{blindfragment}
\begin{blindfragment}
\begin{frontmatter}
\maketitle\newpage
\begin{sfragment}[display=flow]{Preface}
... <<preface>> ...
\end{sfragment}
\clearpage\setcounter{tocdepth}{4}\tableofcontents\clearpage
\end{frontmatter}
\end{blindfragment}
... <<introductory remarks>> ...
\end{blindfragment}
\begin{sfragment}{Introduction}
... <<intro>> ...
\end{sfragment}
... <<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>4</sup>We shied away from redefining the `frontmatter` to induce a `blindomgroup`, but this may be the “right” way to go in the future.

### 19.2.3 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].

### 19.2.4 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` 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>8</sup>

`\STRsemantics`

### 19.2.5 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>8</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.

### 19.2.6 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.

## 19.3 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 20

# 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.

### 20.1 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`.

### 20.2 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.

#### 20.2.1 Package Options

The `notesslides` class takes a variety of class options:<sup>9</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 20.2.2).</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 20.2.4). 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>   |

## 20.2.2 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>5</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>9</sup>EDNOTE: leaving out `noproblems` for the moment until we decide what to do with it.

<sup>5</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.

`\inputref*`  
  
`nparagraph`  
  
`nfragment`  
`ndefinition`  
`nexample`  
`nsproof`  
`nassertion`

### 20.2.3 Header and Footer Lines of the Slides

The default logo provided by the `notesslides` package is the  $\TeX$  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.

`\setsource`  
  
`\setlicensing`

### 20.2.4 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  $\TeX$ 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>10</sup>

`\frameimage`  
  
`\mhframeimage`

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>10</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}
```

## 20.2.5 Colors and Highlighting

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

## 20.2.6 Front Matter, Titles, etc.

### 20.2.7 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{sfragment}[id=<id>]{Excursions}
  \inputref{<path>}
  \printexcursions
\end{sfragment}
\end{note}
```

### 20.2.8 Miscellaneous

## 20.3 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 21

## 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.

### 21.1 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>6</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.

### 21.2 The User Interface

#### 21.2.1 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>6</sup>for the moment multiple choice problems are not supported, but may well be in a future version

## 21.2.2 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.

### 21.2.3 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

### 21.2.4 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.

### 21.2.5 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.

## 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 `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 22

# **`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



## 22.1 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.

## 22.2 The User Interface

### 22.2.1 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].

### 22.2.2 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  
`title` referenced in the title of the assignment sheet), `type` (for the assignment type; e.g. “quiz”,  
`type` or “homework”), `given` (for the date the assignment was given), and `due` (for the date  
`given` the assignment is due).  
`due`

### 22.2.3 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  
`\testnewpage` space accordingly. `\testnewpage` makes a new page in `test` mode, and `\testemptypage`  
`\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 equiv-  
`min` alent in number of minutes, and `reqpts` the points that are required for a perfect grade.  
`reqpts`

### 22.2.4 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.

## 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 `STEX`GitHub repository [\[sTeX\]](#).

1. none reported yet.



Part IV

# Implementation

## Chapter 23

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

### 23.1 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}{2022/03/03}{3.1.0}{sTeX document class}
7 \LoadClass[border=1px,varwidth]{standalone}
8 \setlength\textwidth{15cm}
9
10 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
11 \ProcessOptions
12
13 \RequirePackage{stex}
14 </cls>
```

### 23.2 Preliminaries

```
15 <*package>
16
17 %%%%%%%%% basics.dtx %%%%%%%%%
18
19 \RequirePackage{expl3,l3keys2e,ltxcmds}
20 \ProvidesExplPackage{stex}{2022/03/03}{3.1.0}{sTeX package}
21
22 %\RequirePackage{morewrites}
23 %\RequirePackage{amsmath}
24
25 Package options:
26 \keys_define:nn { stex } {
```

```

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

```

**\stex** The  $\TeX$  logo:

**\sTeX**

```

34 \protected\def\stex{
35   \texorpdfstring{\raisebox{-.5ex}{S}\kern-.5ex\TeX}{sTeX}\xspace%
36 }
37 \let\sTeX\stex

```

(End definition for `\stex` and `\sTeX`. These functions are documented on page 30.)

## 23.3 Messages and logging

```

38 <@=stex_log>
    Warnings and error messages
39 \msg_new:nnn{stex}{error/unknownlanguage}{
40   Unknown~language:~#1
41 }
42 \msg_new:nnn{stex}{warning/nomathhub}{
43   MATHHUB~system~variable~not~found~and~no~
44   \detokenize{\mathhub}~value~set!
45 }
46 \msg_new:nnn{stex}{error/deactivated-macro}{
47   The~\detokenize{#1}~command~is~only~allowed~in~#2!
48 }

```

**\stex\_debug:nn** A simple macro issuing package messages with subpath.

```

49 \cs_new_protected:Nn \stex_debug:nn {
50   \clist_if_in:NnTF \c_stex_debug_clist { all } {
51     \msg_set:nnn{stex}{debug / #1}{
52       \\Debug~#1:~#2\\
53     }
54     \msg_none:nn{stex}{debug / #1}
55   }{
56     \clist_if_in:NnT \c_stex_debug_clist { #1 } {
57       \msg_set:nnn{stex}{debug / #1}{
58         \\Debug~#1:~#2\\
59       }
60       \msg_none:nn{stex}{debug / #1}
61     }
62   }
63 }

```

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

Redirecting messages:

```

64 \clist_if_in:NnTF \c_stex_debug_clist {all} {
65   \msg_redirect_module:nnn{ stex }{ none }{ term }

```

```

66 }{
67   \clist_map_inline:Nn \c_stex_debug_clist {
68     \msg_redirect_name:nnn{ stex }{ debug / ##1 }{ term }
69   }
70 }
71
72 \stex_debug:nn{log}{debug~mode~on}

```

## 23.4 HTML Annotations

```

73 <@@=stex_annotate>
74 \RequirePackage{rustex}

```

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

```

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

```

Conditionals for L<sup>A</sup>T<sub>E</sub>XML:

`\if@latexml`

```

76 \ifcsname if@latexml\endcsname\else
77   \expandafter\newif\csname if@latexml\endcsname\@latexmlfalse
78 \fi

```

(End definition for `\if@latexml`. This function is documented on page 30.)

`\latexml_if_p:`

`\latexml_if:TF`

```

79 \prg_new_conditional:Nnn \latexml_if: {p, T, F, TF} {
80   \if@latexml
81     \prg_return_true:
82   \else:
83     \prg_return_false:
84   \fi:
85 }

```

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

`\l__stex_annotate_arg_tl`  
`\c__stex_annotate_emptyarg_tl`

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

```

86 \tl_new:N \l__stex_annotate_arg_tl
87 \tl_const:Nx \c__stex_annotate_emptyarg_tl {
88   \rustex_if:TF {
89     \rustex_direct_HTML:n { \c_ampersand_str lrm; }
90   }{-}
91 }

```

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

`\__stex_annotate_checkempty:n`

```

92 \cs_new_protected:Nn \__stex_annotate_checkempty:n {
93   \tl_set:Nn \l__stex_annotate_arg_tl { #1 }
94   \tl_if_empty:NT \l__stex_annotate_arg_tl {
95     \tl_set_eq:NN \l__stex_annotate_arg_tl \c__stex_annotate_emptyarg_tl
96   }
97 }

```

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

```

\stex_if_do_html_p: Whether to (locally) produce HTML output
\stex_if_do_html:TF
  98 \bool_new:N \_stex_html_do_output_bool
  99 \bool_set_true:N \_stex_html_do_output_bool
 100
 101 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
 102   \bool_if:nTF \_stex_html_do_output_bool
 103     \prg_return_true: \prg_return_false:
 104 }

```

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

```

\stex_suppress_html:n Whether to (locally) produce HTML output
 105 \cs_new_protected:Nn \stex_suppress_html:n {
 106   \exp_args:Nne \use:nn {
 107     \bool_set_false:N \_stex_html_do_output_bool
 108     #1
 109   }{
 110     \stex_if_do_html:T {
 111       \bool_set_true:N \_stex_html_do_output_bool
 112     }
 113   }
 114 }

```

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

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

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

```

\stex_annotate_invisible:nnn
 115 \rustex_if:TF{
 116   \cs_new_protected:Nn \stex_annotate:nnn {
 117     \__stex_annotate_checkempty:n { #3 }
 118     \rustex_annotate_HTML:nn {
 119       property="stex:#1" ~
 120       resource="#2"
 121     } {
 122       \mode_if_vertical:TF{
 123         \tl_use:N \l__stex_annotate_arg_tl\par
 124       }{
 125         \tl_use:N \l__stex_annotate_arg_tl
 126       }
 127     }
 128   }
 129   \cs_new_protected:Nn \stex_annotate_invisible:n {
 130     \__stex_annotate_checkempty:n { #1 }
 131     \rustex_annotate_HTML:nn {
 132       stex:visible="false" ~
 133       style:display="none"
 134     } {
 135       \mode_if_vertical:TF{
 136         \tl_use:N \l__stex_annotate_arg_tl\par
 137       }{
 138         \tl_use:N \l__stex_annotate_arg_tl
 139       }
 140     }
 141   }

```



```

140     }
141   }
142   \cs_new_protected:Nn \stex_annotate_invisible:nnn {
143     \__stex_annotate_checkempty:n { #3 }
144     \rustex_annotate_HTML:nn {
145       property="stex:#1" ~
146       resource="#2" ~
147       stex:visible="false" ~
148       style:display="none"
149     } {
150       \mode_if_vertical:TF{
151         \tl_use:N \l__stex_annotate_arg_tl\par
152       }{
153         \tl_use:N \l__stex_annotate_arg_tl
154       }
155     }
156   }
157   \NewDocumentEnvironment{stex_annotate_env} { m m } {
158     \par
159     \rustex_annotate_HTML_begin:n {
160       property="stex:#1" ~
161       resource="#2"
162     }
163   }{
164     \par\rustex_annotate_HTML_end:
165   }
166 }{
167   \latexml_if:TF {
168     \cs_new_protected:Nn \stex_annotate:nnn {
169       \__stex_annotate_checkempty:n { #3 }
170       \mode_if_math:TF {
171         \cs:w latexml@annotate@math\cs_end:{#1}{#2}{
172           \tl_use:N \l__stex_annotate_arg_tl
173         }
174       }{
175         \cs:w latexml@annotate@text\cs_end:{#1}{#2}{
176           \tl_use:N \l__stex_annotate_arg_tl
177         }
178       }
179     }
180     \cs_new_protected:Nn \stex_annotate_invisible:n {
181       \__stex_annotate_checkempty:n { #1 }
182       \mode_if_math:TF {
183         \cs:w latexml@invisible@math\cs_end:{
184           \tl_use:N \l__stex_annotate_arg_tl
185         }
186       } {
187         \cs:w latexml@invisible@text\cs_end:{
188           \tl_use:N \l__stex_annotate_arg_tl
189         }
190       }
191     }
192     \cs_new_protected:Nn \stex_annotate_invisible:nnn {
193       \__stex_annotate_checkempty:n { #3 }

```

```

194     \cs:w latexml@annotate@invisible\cs_end:{#1}{#2}{
195       \tl_use:N \l__stex_annotate_arg_tl
196     }
197   }
198   \NewDocumentEnvironment{stex_annotate_env} { m m } {
199     \par\begin{latexml@annotateenv}{#1}{#2}
200   }{
201     \par\end{latexml@annotateenv}
202   }
203 }{
204   \cs_new_protected:Nn \stex_annotate:nnn {#3}
205   \cs_new_protected:Nn \stex_annotate_invisible:n {}
206   \cs_new_protected:Nn \stex_annotate_invisible:nnn {}
207   \NewDocumentEnvironment{stex_annotate_env} { m m } {}{}
208 }
209 }

```

(End definition for `\stex_annotate:nnn`, `\stex_annotate_invisible:n`, and `\stex_annotate_invisible:nnn`. These functions are documented on page 31.)

## 23.5 Babel Languages

```

210 <@@=stex_language>

```

`\c_stex_languages_prop`  
`\c_stex_language_abbrevs_prop`

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

```

211 \prop_const_from_keyval:Nn \c_stex_languages_prop {
212   en = english ,
213   de = ngerman ,
214   ar = arabic ,
215   bg = bulgarian ,
216   ru = russian ,
217   fi = finnish ,
218   ro = romanian ,
219   tr = turkish ,
220   fr = french
221 }
222
223 \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop {
224   english = en ,
225   ngerman = de ,
226   arabic = ar ,
227   bulgarian = bg ,
228   russian = ru ,
229   finnish = fi ,
230   romanian = ro ,
231   turkish = tr ,
232   french = fr
233 }
234 % todo: chinese simplified (zhs)
235 %       chinese traditional (zht)

```

(End definition for `\c_stex_languages_prop` and `\c_stex_language_abbrevs_prop`. These variables are documented on page 31.)

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

```

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

```

## 23.6 Auxiliary Methods

**\stex\_deactivate\_macro:Nn**

```

248 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
249   \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
250   \def#1{
251     \msg_error:nnnn{stex}{error/deactivated-macro}{#1}{#2}
252   }
253 }

```

(End definition for \stex\_deactivate\_macro:Nn. This function is documented on page 31.)

**\stex\_reactivate\_macro:N**

```

254 \cs_new_protected:Nn \stex_reactivate_macro:N {
255   \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname
256 }

```

(End definition for \stex\_reactivate\_macro:N. This function is documented on page 31.)

**\ignorespacesandpars**

```

257 \protected\def\ignorespacesandpars{
258   \begingroup\catcode13=10\relax
259   \@ifnextchar\par{
260     \endgroup\expandafter\ignorespacesandpars\@gobble
261   }{
262     \endgroup
263   }
264 }
265 \</package>

```

(End definition for \ignorespacesandpars. This function is documented on page 31.)

## Chapter 24

# STEX -MathHub Implementation

```
266 <*package>
267
268 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
269
270 <@@=stex_path>
271
272 Warnings and error messages
273 \msg_new:nnn{stex}{error/norepository}{
274   No~archive~#1~found~in~#2
275 }
276 \msg_new:nnn{stex}{error/notinarchive}{
277   Not~currently~in~an~archive,~but~\detokenize{#1}~
278   needs~one!
279 }
280 \msg_new:nnn{stex}{error/nofile}{
281   \detokenize{#1}~could~not~find~file~#2
282 }
283 \msg_new:nnn{stex}{error/twofiles}{
284   \detokenize{#1}~found~two~candidates~for~#2
285 }
```

### 24.1 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`

```
284 \cs_new_protected:Nn \stex_path_from_string:Nn {
285   \str_set:Nx \l_tmpa_str { #2 }
286   \str_if_empty:NTF \l_tmpa_str {
287     \seq_clear:N #1
288   }{
289     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
290     \sys_if_platform_windows:T{
291       \seq_clear:N \l_tmpa_tl
```

```

292 \seq_map_inline:Nn #1 {
293   \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
294   \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
295 }
296 \seq_set_eq:NN #1 \l_tmpa_tl
297 }
298 \stex_path_canonicalize:N #1
299 }
300 }
301

```

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

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

```

302 \cs_new_protected:Nn \stex_path_to_string:NN {
303   \exp_args:Nne \str_set:Nn #2 { \seq_use:Nn #1 / }
304 }
305
306 \cs_new:Nn \stex_path_to_string:N {
307   \seq_use:Nn #1 /
308 }

```

(End definition for `\stex_path_to_string:NN` and `\stex_path_to_string:N`. These functions are documented on page 32.)

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

```

309 \str_const:Nn \c__stex_path_dot_str {.}
310 \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.

```

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

```

```

333         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
334     }
335 }
336 }{
337     \str_if_empty:NF \l_tmpa_tl {
338         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
339     }
340 }
341 }
342 }
343 \seq_gset_eq:NN #1 \l_tmpa_seq
344 }
345 }

```

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

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

```

346 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
347     \seq_if_empty:NTF #1 {
348         \prg_return_false:
349     }{
350         \seq_get_left:NN #1 \l_tmpa_tl
351         \sys_if_platform_windows:TF{
352             \str_if_in:NnTF \l_tmpa_tl {:}{
353                 \prg_return_true:
354             }{
355                 \prg_return_false:
356             }
357         }{
358             \str_if_empty:NTF \l_tmpa_tl {
359                 \prg_return_true:
360             }{
361                 \prg_return_false:
362             }
363         }
364     }
365 }

```

(End definition for `\stex_path_if_absolute:N $\underline{TF}$` . This function is documented on page 32.)

## 24.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

366 \str_new:N\l_stex_kpsewhich_return_str
367 \cs_new_protected:Nn \stex_kpsewhich:n {
368     \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
369     \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
370     \tl_trim_spaces:N \l_stex_kpsewhich_return_str
371 }

```

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

We determine the PWD

`\c_stex_pwd_seq`  
`\c_stex_pwd_str`

```

372 \sys_if_platform_windows:TF{
373   \begingroup\escapechar=-1\catcode'\=12
374   \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
375   \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
376   \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_
377   }}{
378   \stex_kpsewhich:n{-var-value~PWD}
379   }
380
381 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
382 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
383 \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 32.)

## 24.3 File Hooks and Tracking

```

384 <@@=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

```

385 \seq_gclear_new:N\g__stex_files_stack

```

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`  
`\c_stex_mainfile_str`

```

386 \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
387 \stex_path_from_string:Nn \c_stex_mainfile_seq
388   \c_stex_mainfile_str

```

(End definition for `\c_stex_mainfile_seq` and `\c_stex_mainfile_str`. These variables are documented on page 32.)

`\g_stex_currentfile_seq`

```

389 \seq_gclear_new:N\g_stex_currentfile_seq

```

(End definition for `\g_stex_currentfile_seq`. This variable is documented on page 33.)

`\stex_filestack_push:n`

```

390 \cs_new_protected:Nn \stex_filestack_push:n {
391   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
392   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
393     \stex_path_from_string:Nn\g_stex_currentfile_seq{
394       \c_stex_pwd_str/#1
395     }
396   }
397   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
398   \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
399 }

```

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

`\stex_filestack_pop:`

```

400 \cs_new_protected:Nn \stex_filestack_pop: {
401   \seq_if_empty:NF\g__stex_files_stack{
402     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
403   }
404   \seq_if_empty:NTF\g__stex_files_stack{
405     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
406   }{
407     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
408     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
409   }
410 }
```

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

Hooks for the current file:

```

411 \AddToHook{file/before}{
412   \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
413 }
414 \AddToHook{file/after}{
415   \stex_filestack_pop:
416 }
```

## 24.4 MathHub Repositories

417 `<@=stex_mathhub>`

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

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

```

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



```

441 }
442 \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
443 \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
444 }

```

(End definition for `\mathhub`, `\c_stex_mathhub_seq`, and `\c_stex_mathhub_str`. These variables are documented on page 33.)

`\_stex_mathhub_do_manifest:n` Checks whether the manifest for archive #1 already exists, and if not, finds and parses the corresponding manifest file

```

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

```

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

`\l__stex_mathhub_manifest_file_seq`

```

461 \seq_new:N\l__stex_mathhub_manifest_file_seq

```

(End definition for `\l__stex_mathhub_manifest_file_seq`.)

`\_stex_mathhub_find_manifest:N` Attempts to find the MANIFEST.MF in some file path and stores its path in `\l__stex_mathhub_manifest_file_seq`:

```

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

```

```

480         \bool_set_false:N\l_tmpa_bool
481     }{
482         \file_if_exist:nTF{
483             \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
484         }{
485             \seq_put_right:Nn\l_tmpa_seq{meta-inf}
486             \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
487             \bool_set_false:N\l_tmpa_bool
488         }{
489             \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
490         }
491     }
492 }
493 }
494 }
495 \seq_set_eq:NN\l__stex_mathhub_manifest_file_seq\l_tmpa_seq
496 }

```

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

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

```

497 \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:

```

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

```

```

526     {ns} {
527         \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
528         { ns } \l_tmpb_tl
529     }
530     {dependencies} {
531         \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
532         { deps } \l_tmpb_tl
533     }
534     }{}{}
535 }{}
536 }
537 \ior_close:N \c__stex_mathhub_manifest_ior
538 }

```

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

`\stex_set_current_repository:n`

```

539 \cs_new_protected:Nn \stex_set_current_repository:n {
540     \stex_require_repository:n { #1 }
541     \prop_set_eq:Nc \l_stex_current_repository_prop {
542         c_stex_mathhub_#1_manifest_prop
543     }
544 }

```

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

`\stex_require_repository:n`

```

545 \cs_new_protected:Nn \stex_require_repository:n {
546     \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
547         \stex_debug:nn{mathhub}{Opening~archive:~#1}
548         \_stex_mathhub_do_manifest:n { #1 }
549     }
550 }

```

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

`\l_stex_current_repository_prop`

Current MathHub repository

```

551 %\prop_new:N \l_stex_current_repository_prop
552
553 \_stex_mathhub_find_manifest:N \c_stex_pwd_seq
554 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
555     \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
556 } {
557     \_stex_mathhub_parse_manifest:n { main }
558     \prop_get:Nn \c_stex_mathhub_main_manifest_prop {id}
559     \l_tmpa_str
560     \prop_set_eq:cN { c_stex_mathhub\_l_tmpa_str_manifest_prop }
561     \c_stex_mathhub_main_manifest_prop
562     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
563     \stex_debug:nn{mathhub}{Current~repository:~
564         \prop_item:Nn \l_stex_current_repository_prop {id}
565     }
566 }

```

(End definition for `\l_stex_current_repository_prop`. This variable is documented on page 33.)

`\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.

```

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

```

(End definition for `\stex_in_repository:nn`. This function is documented on page [33](#).)

## 24.5 Using Content in Archives

`\mhpath`

```

605 \def \mhpath #1 #2 {
606   \exp_args:Ne \tl_if_empty:nTF{#1}{
607     \c_stex_mathhub_str /
608     \prop_item:Nn \l_stex_current_repository_prop { id }
609     / source / #2
610 }{
611   \c_stex_mathhub_str / #1 / source / #2

```

```

612 }
613 }

```

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

`\inputref`  
`\mhinput`

```

614 \newif \ifinputref \inputreffalse
615
616 \cs_new_protected:Nn \__stex_mathhub_mhinput:nn {
617   \stex_in_repository:nn {#1} {
618     \ifinputref
619       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
620     \else
621       \inputreftrue
622       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
623     \inputreffalse
624   \fi
625 }
626 }
627 \NewDocumentCommand \mhinput { 0{} m}{
628   \stex_mhinput:nn{ #1 }{ #2 }
629 }
630
631 \cs_new_protected:Nn \__stex_mathhub_inputref:nn {
632   \stex_in_repository:nn {#1} {
633     \bool_lazy_any:nTF {
634       {\rustex_if_p:}
635       {\latexml_if_p:}
636     } {
637       \str_clear:N \l_tmpa_str
638       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
639         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
640       }
641       \stex_annotate_invisible:nnn{inputref}{
642         \l_tmpa_str / #2
643       }{}
644     }{
645       \begingroup
646         \inputreftrue
647         \input{ \c_stex_mathhub_str / ##1 / source / #2 }
648       \endgroup
649     }
650   }
651 }
652 \NewDocumentCommand \inputref { 0{} m}{
653   \__stex_mathhub_inputref:nn{ #1 }{ #2 }
654 }

```

(End definition for `\inputref` and `\mhinput`. These functions are documented on page 34.)

`\addmhbibresource`

```

655 \cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {
656   \stex_in_repository:nn {#1} {
657     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
658   }

```

```

659 }
660 \newcommand\addmhbibresource[2][]{
661   \_stex_mathhub_mhbibresource:nn{ #1 }{ #2 }
662 }

```

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

### \libinput

```

663 \cs_new_protected:Npn \libinput #1 {
664   \prop_if_exist:NF \l_stex_current_repository_prop {
665     \msg_error:nnn{stex}{error/notinarchive}\libinput
666   }
667   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
668     \msg_error:nnn{stex}{error/notinarchive}\libinput
669   }
670   \seq_clear:N \l__stex_mathhub_libinput_files_seq
671   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
672   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
673
674   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
675     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}
676     \IfFileExists{ \l_tmpa_str }{
677       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
678     }{}
679     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
680     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
681   }
682
683   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
684   \IfFileExists{ \l_tmpa_str }{
685     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
686   }{}
687
688   \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
689     \msg_error:nxxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
690   }{
691     \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
692       \input{ ##1 }
693     }
694   }
695 }

```

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

### \libusepackage

```

696 \NewDocumentCommand \libusepackage {0{} m} {
697   \prop_if_exist:NF \l_stex_current_repository_prop {
698     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
699   }
700   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
701     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
702   }
703   \seq_clear:N \l__stex_mathhub_libinput_files_seq
704   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
705   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str

```

```

706
707 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
708   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2}
709   \IfFileExists{ \l_tmpa_str.sty }{
710     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
711   }{}
712   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
713   \seq_put_right:No \l_tmpa_seq \l_tmpa_str
714 }
715
716 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2}
717 \IfFileExists{ \l_tmpa_str.sty }{
718   \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
719 }{}
720
721 \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
722   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
723 }{
724   \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
725     \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
726       \usepackage[#1]{ #1 }
727     }
728   }{
729     \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
730   }
731 }
732 }

```

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

`\mhgraphics`  
`\cmhgraphics`

```

733
734 \AddToHook{begindocument}{
735   \ltx@ifpackageloaded{graphicx}{
736     \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
737     \newcommand\mhgraphics[2][]{%
738       \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
739       \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}}
740     \newcommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
741   }{}

```

(End definition for `\mhgraphics` and `\cmhgraphics`. These functions are documented on page 34.)

`\lstinputmhlisting`  
`\clstinputmhlisting`

```

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

```

*(End definition for `\lstinputmhlisting` and `\clstinputmhlisting`. These functions are documented on page 34.)*



## Chapter 25

# STEX -References Implementation

```
752 <*package>
753
754 %%%%%%%%%% references.dtx %%%%%%%%%%
755
756 <@@=stex_refs>
    Warnings and error messages
757
```

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

```
758 %\iow_new:N \c__stex_refs_refs_iow
759 \AddToHook{begindocument}{
760 % \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
761 }
762 \AddToHook{enddocument}{
763 % \iow_close:N \c__stex_refs_refs_iow
764 }
```

`\STEXreftitle`

```
765 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
766
767 \NewDocumentCommand \STEXreftitle { m } {
768 \tl_gset:Nx \g__stex_refs_title_tl { #1 }
769 }
```

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

### 25.1 Document URIs and URLs

`\l_stex_current_docns_str`

```
770 \str_new:N \l_stex_current_docns_str
```

*(End definition for `\l_stex_current_docns_str`. This variable is documented on page 35.)*

`\stex_get_document_uri:`

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

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

`\l_stex_current_docurl_str`

```
811 \str_new:N \l_stex_current_docurl_str
```

(End definition for `\l_stex_current_docurl_str`. This variable is documented on page 35.)

`\stex_get_document_url:`

```
812 \cs_new_protected:Nn \stex_get_document_url: {  
813   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq  
814   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str  
815   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
```

```

816 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
817 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
818
819 \str_clear:N \l_tmpa_str
820 \prop_if_exist:NT \l_stex_current_repository_prop {
821   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
822     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
823       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
824     }
825   }
826 }
827
828 \str_if_empty:NTF \l_tmpa_str {
829   \str_set:Nx \l_stex_current_docurl_str {
830     file:/\stex_path_to_string:N \l_tmpa_seq
831   }
832 }{
833   \bool_set_true:N \l_tmpa_bool
834   \bool_while_do:Nn \l_tmpa_bool {
835     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
836     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
837       {source} { \bool_set_false:N \l_tmpa_bool }
838     }{}{
839       \seq_if_empty:NT \l_tmpa_seq {
840         \bool_set_false:N \l_tmpa_bool
841       }
842     }
843   }
844
845   \seq_if_empty:NTF \l_tmpa_seq {
846     \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
847   }{
848     \str_set:Nx \l_stex_current_docurl_str {
849       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
850     }
851   }
852 }
853 }

```

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

## 25.2 Setting Reference Targets

```

854 \str_const:Nn \c__stex_refs_url_str{URL}
855 \str_const:Nn \c__stex_refs_ref_str{REF}
856 \str_new:N \l__stex_refs_curr_label_str
857 % @currentlabel -> number
858 % @currentlabelname -> title
859 % @currentHref -> name.number <- id of some kind
860 % \theH# -> \arabic{section}
861 % \the# -> number
862 % \hyper@makecurrent{#}
863 \int_new:N \l__stex_refs_unnamed_counter_int

```

`\stex_ref_new_doc_target:n`

```

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

```

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

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

```

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

```

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

```

902 \AtEndDocument{
903   \def\stexauxadddocref#1 #2 {}{}
904 }

```

`\stex_ref_new_sym_target:n`

```

905 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
906   \stex_if_smsmode:TF {
907     \str_if_exist:cF{sref_sym_#1_type}{
908       \stex_get_document_url:
909       \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str

```

```

910     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
911   }
912 }{
913   \str_if_empty:NF \l__stex_refs_curr_label_str {
914     \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
915     \immediate\write\@auxout{
916       \exp_not:N\expandafter\def\exp_not:N\csname sref_sym_#1_label_str\exp_not:N\endcsname
917         \l__stex_refs_curr_label_str
918     }
919   }
920 }
921 }
922 }

```

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

## 25.3 Using References

```

923 \str_new:N \l__stex_refs_indocument_str

```

**\sref** Optional arguments:

```

924
925 \keys_define:nn { stex / sref } {
926   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
927   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
928   pre           .tl_set:N = \l__stex_refs_pre_tl ,
929   post          .tl_set:N = \l__stex_refs_post_tl ,
930 }
931 \cs_new_protected:Nn \__stex_refs_args:n {
932   \tl_clear:N \l__stex_refs_linktext_tl
933   \tl_clear:N \l__stex_refs_fallback_tl
934   \tl_clear:N \l__stex_refs_pre_tl
935   \tl_clear:N \l__stex_refs_post_tl
936   \str_clear:N \l__stex_refs_repo_str
937   \keys_set:nn { stex / sref } { #1 }
938 }

```

The actual macro:

```

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

```

```

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

```

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

### \srefsym

```

1001 \NewDocumentCommand \srefsym { 0{} m}{
1002   \stex_get_symbol:n { #2 }
1003   \__stex_refs_sym_aux:nn{##1}{\l_stex_get_symbol_uri_str}
1004 }

```

```

1005
1006 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1007   \str_if_exist:cTF {sref_sym_#2 _label_str }{
1008     \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1009   }{
1010     \__stex_refs_args:n { #1 }
1011     \str_if_empty:NTF \l__stex_refs_indocument_str {
1012       \tl_if_exist:cTF{sref_sym_#2 _type}{
1013         % doc uri in \l_tmpb_str
1014         \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1015         \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1016           % reference
1017           \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1018             \cs_if_exist:cTF{autoref}{
1019               \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1020             }{
1021               \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1022             }
1023           }{
1024             \ltx@ifpackageloaded{hyperref}{
1025               \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1026             }{
1027               \l__stex_refs_linktext_tl
1028             }
1029           }
1030         }{
1031           % URL
1032           \ltx@ifpackageloaded{hyperref}{
1033             \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl \
1034           }{
1035             \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_re
1036           }
1037         }
1038       }{
1039         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_l
1040       }
1041     }{
1042       % TODO
1043     }
1044   }
1045 }

```

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

**\srefsymuri**

```

1046 \cs_new_protected:Npn \srefsymuri #1 #2 {
1047   \__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1048 }

```

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

```

1049 </package>

```

## Chapter 26

# STEX -Modules Implementation

```
1050 <*package>
1051
1052 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1053
1054 <@@=stex_modules>
1055
1056   Warnings and error messages
1057   \msg_new:nnn{stex}{error/unknownmodule}{
1058     No~module~#1~found
1059   }
1060   \msg_new:nnn{stex}{error/syntax}{
1061     Syntax~error:~#1
1062   }
1063   \msg_new:nnn{stex}{error/siglanguage}{
1064     Module~#1~declares~signature~#2,~but~does~not~
1065     declare~its~language
1066   }
1067   \msg_new:nnn{stex}{warning/deprecated}{
1068     #1~is~deprecated;~please~use~#2~instead!
1069   }
1070   \msg_new:nnn{stex}{error/conflictingmodules}{
1071     Conflicting~imports~for~module~#1
1072   }
1073
1074 \l_stex_current_module_str The current module:
1075 \str_new:N \l_stex_current_module_str
1076
1077 (End definition for \l_stex_current_module_str. This variable is documented on page 38.)
1078
1079 \l_stex_all_modules_seq Stores all available modules
1080 \seq_new:N \l_stex_all_modules_seq
1081
1082 (End definition for \l_stex_all_modules_seq. This variable is documented on page 38.)
```



```

\stex_if_in_module_p:
\stex_if_in_module:TF
1074 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1075   \str_if_empty:NTF \l_stex_current_module_str
1076   \prg_return_false: \prg_return_true:
1077 }

```

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

```

\stex_if_module_exists_p:n
\stex_if_module_exists:nTF
1078 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1079   \prop_if_exist:cTF { c_stex_module_#1_prop }
1080   \prg_return_true: \prg_return_false:
1081 }

```

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

`\stex_add_to_current_module:n` Only allowed within modules:

```

\STEXexport
1082 \cs_new_protected:Nn \stex_add_to_current_module:n {
1083   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1084 }
1085 \cs_new_protected:Npn \STEXexport {
1086   \begingroup
1087   \newlinechar=-1\relax
1088   \endlinechar=-1\relax
1089   %\catcode'\ = 9\relax
1090   \expandafter\endgroup\__stex_modules_export:n
1091 }
1092 \cs_new_protected:Nn \__stex_modules_export:n {
1093   \ignorespaces #1
1094   \stex_add_to_current_module:n { \ignorespaces #1 }
1095   \stex_smsmode_do:
1096 }
1097 \stex_deactivate_macro:Nn \STEXexport {module~environments}

```

(End definition for `\stex_add_to_current_module:n` and `\STEXexport`. These functions are documented on page 38.)

```

\stex_add_constant_to_current_module:n
1098 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1099   \str_set:Nx \l_tmpa_str { #1 }
1100   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1101 }

```

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

```

\stex_add_import_to_current_module:n
1102 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1103   \str_set:Nx \l_tmpa_str { #1 }
1104   \exp_args:Nno
1105   \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1106     \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1107   }
1108 }

```

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

`\stex_collect_imports:n`

```

1109 \cs_new_protected:Nn \stex_collect_imports:n {
1110   \seq_clear:N \l_stex_collect_imports_seq
1111   \__stex_modules_collect_imports:n {#1}
1112 }
1113 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1114   \seq_map_inline:cn {c_stex_module_#1_imports} {
1115     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1116       \__stex_modules_collect_imports:n { ##1 }
1117     }
1118   }
1119   \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1120     \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1121   }
1122 }

```

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

`\stex_do_up_to_module:n`

```

1123 \int_new:N \l__stex_modules_group_depth_int
1124 \tl_new:N \l__stex_modules_aftergroup_tl
1125 \cs_new_protected:Nn \stex_do_up_to_module:n {
1126   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1127     #1
1128   }{
1129     #1
1130     \expandafter \tl_gset:Nn \expandafter \l__stex_modules_aftergroup_tl \expandafter { \l__
1131       \aftergroup\__stex_modules_aftergroup_do:
1132     }
1133   }
1134   \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1135     \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1136       \l__stex_modules_aftergroup_tl
1137       \tl_clear:N \l__stex_modules_aftergroup_tl
1138     }{
1139       \l__stex_modules_aftergroup_tl
1140       \aftergroup\__stex_modules_aftergroup_do:
1141     }
1142   }

```

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

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

1143

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

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

```

1144 \str_new:N \l_stex_modules_ns_str
1145 \str_new:N \l_stex_modules_subpath_str

```

```

1146 \cs_new_protected:Nn \__stex_modules_compute_namespace:nN {
1147   \str_set:Nx \l_tmpa_str { #1 }
1148   \seq_set_eq:NN \l_tmpa_seq #2
1149   % split off file extension
1150   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1151   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1152   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1153   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1154
1155   \bool_set_true:N \l_tmpa_bool
1156   \bool_while_do:Nn \l_tmpa_bool {
1157     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1158     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1159       {source} { \bool_set_false:N \l_tmpa_bool }
1160     }{}{
1161       \seq_if_empty:NT \l_tmpa_seq {
1162         \bool_set_false:N \l_tmpa_bool
1163       }
1164     }
1165   }
1166
1167   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
1168   \str_if_empty:NTF \l_stex_modules_subpath_str {
1169     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
1170   }{
1171     \str_set:Nx \l_stex_modules_ns_str {
1172       \l_tmpa_str/\l_stex_modules_subpath_str
1173     }
1174   }
1175 }
1176
1177 \cs_new_protected:Nn \stex_modules_current_namespace: {
1178   \str_clear:N \l_stex_modules_subpath_str
1179   \prop_if_exist:NTF \l_stex_current_repository_prop {
1180     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1181     \__stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1182   }{
1183     % split off file extension
1184     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1185     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1186     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1187     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1188     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1189     \str_set:Nx \l_stex_modules_ns_str {
1190       file:/\stex_path_to_string:N \l_tmpa_seq
1191     }
1192   }
1193 }

```

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

## 26.1 The smodule environment

smodule arguments:

```

1194 \keys_define:nn { stex / module } {
1195   title      .tl_set:N      = \smodulename ,
1196   type       .str_set_x:N   = \smoduletype ,
1197   id         .str_set_x:N   = \smoduleid ,
1198   deprecate  .str_set_x:N   = \l_stex_module_deprecate_str ,
1199   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1200   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1201   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1202   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1203   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1204   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1205   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1206 }
1207
1208 \cs_new_protected:Nn \__stex_modules_args:n {
1209   \str_clear:N \smodulename
1210   \str_clear:N \smoduletype
1211   \str_clear:N \smoduleid
1212   \str_clear:N \l_stex_module_ns_str
1213   \str_clear:N \l_stex_module_deprecate_str
1214   \str_clear:N \l_stex_module_lang_str
1215   \str_clear:N \l_stex_module_sig_str
1216   \str_clear:N \l_stex_module_creators_str
1217   \str_clear:N \l_stex_module_contributors_str
1218   \str_clear:N \l_stex_module_meta_str
1219   \str_clear:N \l_stex_module_srccite_str
1220   \keys_set:nn { stex / module } { #1 }
1221 }
1222
1223 % module parameters here? In the body?
1224
```

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

```

1225 \cs_new_protected:Nn \stex_module_setup:nn {
1226   \str_set:Nx \l_stex_module_name_str { #2 }
1227   \__stex_modules_args:n { #1 }
1228
1229   First, we set up the name and namespace of the module.
1230   Are we in a nested module?
1231
1232   \stex_if_in_module:TF {
1233     % Nested module
1234     \prop_get:cnN {c_stex_module_\l_stex_current_module_str _prop}
1235       { ns } \l_stex_module_ns_str
1236     \str_set:Nx \l_stex_module_name_str {
1237       \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
1238       { name } / \l_stex_module_name_str
1239     }
1240   }{
1241     % not nested:
1242     \str_if_empty:NT \l_stex_module_ns_str {
1243       \stex_modules_current_namespace:
1244     }
1245   }
1246 }

```

```

1240 \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1241 \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1242 / {\l_stex_module_ns_str}
1243 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1244 \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1245 \str_set:Nx \l_stex_module_ns_str {
1246 \stex_path_to_string:N \l_tmpa_seq
1247 }
1248 }
1249 }
1250 }

```

Next, we determine the language of the module:

```

1251 \str_if_empty:NT \l_stex_module_lang_str {
1252 \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1253 \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1254 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1255 \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1256 \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1257 \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1258 inferred~from~file~name}
1259 \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1260 }
1261 }
1262
1263 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
1264 \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1265 \l_tmpa_str {
1266 \ltx@ifpackageloaded{babel}{
1267 \exp_args:Nx \selectlanguage { \l_tmpa_str }
1268 }{}
1269 } {
1270 \msg_error:nxx{stex}{error/unknownlanguage}{\l_tmpa_str}
1271 }
1272 }}

```

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

```

1273 \str_if_empty:NTF \l_stex_module_sig_str {
1274 \exp_args:Nnx \prop_gset_from_keyval:cn {
1275 c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1276 } {
1277 name = \l_stex_module_name_str ,
1278 ns = \l_stex_module_ns_str ,
1279 file = \exp_not:o { \g_stex_currentfile_seq } ,
1280 lang = \l_stex_module_lang_str ,
1281 sig = \l_stex_module_sig_str ,
1282 deprecate = \l_stex_module_deprecate_str ,
1283 meta = \l_stex_module_meta_str
1284 }
1285 \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1286 \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1287 \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1288 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1289 \str_if_empty:NT \l_stex_module_meta_str {
1290   \str_set:Nx \l_stex_module_meta_str {
1291     \c_stex_metatheory_ns_str ? Metatheory
1292   }
1293 }
1294 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1295   \bool_set_true:N \l_stex_in_meta_bool
1296   \exp_args:Nx \stex_add_to_current_module:n {
1297     \bool_set_true:N \l_stex_in_meta_bool
1298     \stex_activate_module:n {\l_stex_module_meta_str}
1299     \bool_set_false:N \l_stex_in_meta_bool
1300   }
1301   \stex_activate_module:n {\l_stex_module_meta_str}
1302   \bool_set_false:N \l_stex_in_meta_bool
1303 }
1304 }{
1305   \str_if_empty:NT \l_stex_module_lang_str {
1306     \msg_error:nnxx{stex}{error/siglanguage}{
1307       \l_stex_module_ns_str?\l_stex_module_name_str
1308     }{\l_stex_module_sig_str}
1309   }
1310
1311   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1312   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1313   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1314   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1315   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1316   \str_set:Nx \l_tmpa_str {
1317     \stex_path_to_string:N \l_tmpa_seq /
1318     \l_tmpa_str . \l_stex_module_sig_str .tex
1319   }
1320   \IfFileExists \l_tmpa_str {
1321     \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1322       \str_clear:N \l_stex_current_module_str
1323       \seq_clear:N \l_stex_all_modules_seq
1324       \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1325     }
1326   }{
1327     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1328   }
1329   \stex_if_smsmode:F {
1330     \stex_activate_module:n {
1331       \l_stex_module_ns_str ? \l_stex_module_name_str
1332     }
1333   }
1334   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1335 }
1336 \str_if_empty:NF \l_stex_module_deprecate_str {
1337   \msg_warning:nnxx{stex}{warning/deprecated}{
1338     Module~\l_stex_current_module_str
1339   }{
1340     \l_stex_module_deprecate_str
1341   }

```

```

1342 }
1343 \seq_put_right:Nx \l_stex_all_modules_seq {
1344   \l_stex_module_ns_str ? \l_stex_module_name_str
1345 }
1346 }

```

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

**smodule** The module environment.

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

```

1347 \cs_new_protected:Nn \_stex_modules_begin_module: {
1348   \stex_reactivate_macro:N \STEXexport
1349   \stex_reactivate_macro:N \importmodule
1350   \stex_reactivate_macro:N \symdecl
1351   \stex_reactivate_macro:N \notation
1352   \stex_reactivate_macro:N \symdef
1353
1354   \stex_debug:nn{modules}{
1355     New~module:\\
1356     Namespace:~\l_stex_module_ns_str\\
1357     Name:~\l_stex_module_name_str\\
1358     Language:~\l_stex_module_lang_str\\
1359     Signature:~\l_stex_module_sig_str\\
1360     Metatheory:~\l_stex_module_meta_str\\
1361     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1362   }
1363
1364   \stex_if_smsmode:F{
1365     \begin{stex_annotate_env} {theory} {
1366       \l_stex_module_ns_str ? \l_stex_module_name_str
1367     }
1368
1369     \stex_annotate_invisible:nnn{header}{} {
1370       \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1371       \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1372       \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1373         \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1374       }
1375       \str_if_empty:NF \smoduletype {
1376         \stex_annotate:nnn{type}{\smoduletype}{}
1377       }
1378     }
1379   }
1380   \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1381   % TODO: Inherit metatheory for nested modules?
1382 }
1383 \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}`

```

1384 \cs_new_protected:Nn \_stex_modules_end_module: {
1385   \stex_debug:nn{modules}{Closing~module~\prop_item:cn {c_stex_module\_l_stex_current_module}
1386 }

```

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

The core environment

```

1387 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1388 \NewDocumentEnvironment { smodule } { 0{} m } {
1389   \stex_module_setup:nn{#1}{#2}
1390   \par
1391   \stex_if_smsmode:F{
1392     \tl_clear:N \l_tmpa_tl
1393     \clist_map_inline:Nn \smodulotype {
1394       \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1395         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1396       }
1397     }
1398     \tl_if_empty:NTF \l_tmpa_tl {
1399       \__stex_modules_smodule_start:
1400     }{
1401       \l_tmpa_tl
1402     }
1403   }
1404   \__stex_modules_begin_module:
1405   \str_if_empty:NF \smoduleid {
1406     \stex_ref_new_doc_target:n \smoduleid
1407   }
1408   \stex_smsmode_do:
1409 } {
1410   \__stex_modules_end_module:
1411   \stex_if_smsmode:F {
1412     \end{stex_annotate_env}
1413     \clist_set:Nn \l_tmpa_clist \smodulotype
1414     \tl_clear:N \l_tmpa_tl
1415     \clist_map_inline:Nn \l_tmpa_clist {
1416       \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1417         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1418       }
1419     }
1420     \tl_if_empty:NTF \l_tmpa_tl {
1421       \__stex_modules_smodule_end:
1422     }{
1423       \l_tmpa_tl
1424     }
1425   }
1426 }

```

**\stexpatchmodule**

```

1427 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1428 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1429
1430 \newcommand\stexpatchmodule[3] [] {
1431   \str_set:Nx \l_tmpa_str{ #1 }
1432   \str_if_empty:NTF \l_tmpa_str {
1433     \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1434     \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1435   }{

```



```

1436     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1437     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1438   }
1439 }

```

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

## 26.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n
1440 \NewDocumentCommand \STEXModule { m } {
1441   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1442   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1443   \tl_set:Nn \l_tmpa_tl {
1444     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1445   }
1446   \seq_map_inline:Nn \l_stex_all_modules_seq {
1447     \str_set:Nn \l_tmpb_str { ##1 }
1448     \str_if_eq:eeT { \l_tmpa_str } {
1449       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1450     } {
1451       \seq_map_break:n {
1452         \tl_set:Nn \l_tmpa_tl {
1453           \stex_invoke_module:n { ##1 }
1454         }
1455       }
1456     }
1457   }
1458   \l_tmpa_tl
1459 }
1460
1461 \cs_new_protected:Nn \stex_invoke_module:n {
1462   \stex_debug:nn{modules}{Invoking~module~#1}
1463   \peek_charcode_remove:NTF ! {
1464     \__stex_modules_invoke_uri:nN { #1 }
1465   } {
1466     \peek_charcode_remove:NTF ? {
1467       \__stex_modules_invoke_symbol:nn { #1 }
1468     } {
1469       \msg_error:nnx{stex}{error/syntax}{
1470         ?~or~!~expected~after~
1471         \c_backslash_str STEXModule{#1}
1472       }
1473     }
1474   }
1475 }
1476
1477 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1478   \str_set:Nn #2 { #1 }
1479 }
1480
1481 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1482   \stex_invoke_symbol:n{#1?#2}

```

```
1483 }
```

(End definition for `\STEXModule` and `\stex_invoke_module:n`. These functions are documented on page 39.)

`\stex_activate_module:n`

```
1484 \bool_new:N \l_stex_in_meta_bool
1485 \bool_set_false:N \l_stex_in_meta_bool
1486 \cs_new_protected:Nn \stex_activate_module:n {
1487   \stex_debug:nn{modules}{Activating~module~#1}
1488   \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1489     \msg_error:nnn{stex}{error/conflictingmodules}{ #1 }
1490   }
1491   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1492     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1493     \use:c{ c_stex_module_#1_code }
1494   }
1495 }
```

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

```
1496 \</package>
```

## Chapter 27

# STEX -Module Inheritance Implementation

```
1497 <*package>
1498
1499 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1500
```

### 27.1 SMS Mode

```
1501 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1502 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1503 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1504 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1505
1506 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1507   \makeatletter
1508   \makeatother
1509   \ExplSyntaxOn
1510   \ExplSyntaxOff
1511   \rustexBREAK
1512 }
1513
1514 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1515   \symdef
1516   \importmodule
1517   \notation
1518   \symdecl
1519   \STEXexport
1520   \inlineass
1521   \inlinedef
1522   \inlineex
1523   \endinput
1524   \setnotation
```

```

1525 \copynotation
1526 }
1527
1528 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1529   \tl_to_str:n {
1530     smodule,
1531     copymodule,
1532     interpretmodule,
1533     sdefinition,
1534     sexample,
1535     sassertion,
1536     sparagraph
1537   }
1538 }

```

(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 41.)

`\stex_if_smsmode_p:`  
`\stex_if_smsmode:TF`

```

1539 \bool_new:N \g__stex_smsmode_bool
1540 \bool_set_false:N \g__stex_smsmode_bool
1541 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1542   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1543 }

```

(End definition for `\stex_if_smsmode:TF`. This function is documented on page 41.)

`\_stex_smsmode_in_smsmode:nn`

```

1544 \cs_new_protected:Nn \_stex_smsmode_in_smsmode:nn {
1545   \vbox_set:Nn \l_tmpa_box {
1546     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1547     \bool_gset_true:N \g__stex_smsmode_bool
1548     #2
1549     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1550   }
1551   \box_clear:N \l_tmpa_box
1552 }

```

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

`\stex_file_in_smsmode:nn`

```

1553 \quark_new:N \q__stex_smsmode_break
1554
1555 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1556   \stex_filestack_push:n{#1}
1557   \_stex_smsmode_in_smsmode:nn{#1} {
1558     #2
1559     \everyeof{\q__stex_smsmode_break\noexpand}
1560     \expandafter\expandafter\expandafter
1561     \stex_smsmode_do:
1562     \csname @ @ input\endcsname "#1"\relax
1563   }
1564   \stex_filestack_pop:
1565 }

```

(End definition for `\stex_file_in_smsmode:nn`. This function is documented on page 42.)

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

```

1566 \cs_new_protected:Npn \stex_smsmode_do: {
1567   \stex_if_smsmode:T {
1568     \__stex_smsmode_do:w
1569   }
1570 }
1571 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1572   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
1573     \expandafter\if\expandafter\relax\noexpand#1
1574     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1575   } \else\expandafter\__stex_smsmode_do:w\fi
1576 }{
1577   \__stex_smsmode_do:w % #1
1578 }
1579 }
1580 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1581   \cs_if_eq:NNTF #1 \q__stex_smsmode_break {
1582     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1583       #1\__stex_smsmode_do:w
1584     }{
1585       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1586         #1
1587       }{
1588         \cs_if_eq:NNTF \begin #1 {
1589           \__stex_smsmode_check_begin:n
1590         }{
1591           \cs_if_eq:NNTF \end #1 {
1592             \__stex_smsmode_check_end:n
1593           }{
1594             \__stex_smsmode_do:w
1595           }
1596         }
1597       }
1598     }
1599   }
1600 }
1601
1602 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1603   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1604     \begin{#1}
1605   }{
1606     \__stex_smsmode_do:w
1607   }
1608 }
1609 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1610   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1611     \end{#1}\__stex_smsmode_do:w
1612   }{
1613     \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1614   }
1615 }
```

(End definition for `\stex_smsmode_do:.` This function is documented on page 42.)

## 27.2 Inheritance

```

1616 <@@=stex_importmodule>

\stex_import_module_uri:nn

1617 \cs_new_protected:Nn \stex_import_module_uri:nn {
1618   \str_set:Nx \l_stex_import_archive_str { #1 }
1619   \str_set:Nn \l_stex_import_path_str { #2 }
1620
1621   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1622   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1623   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1624
1625   \stex_modules_current_namespace:
1626   \bool_lazy_all:nTF {
1627     {\str_if_empty_p:N \l_stex_import_archive_str}
1628     {\str_if_empty_p:N \l_stex_import_path_str}
1629     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1630   }{
1631     \str_set_eq:NN \l_stex_import_path_str \l_stex_modules_subpath_str
1632     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1633   }{
1634     \str_if_empty:NT \l_stex_import_archive_str {
1635       \prop_if_exist:NT \l_stex_current_repository_prop {
1636         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1637       }
1638     }
1639     \str_if_empty:NTF \l_stex_import_archive_str {
1640       \str_if_empty:NF \l_stex_import_path_str {
1641         \str_set:Nx \l_stex_import_ns_str {
1642           \l_stex_module_ns_str / \l_stex_import_path_str
1643         }
1644       }
1645     }{
1646       \stex_require_repository:n \l_stex_import_archive_str
1647       \prop_get:cnN { c_stex_mathhub\_l_stex_import_archive_str_manifest_prop } { ns }
1648       \l_stex_import_ns_str
1649       \str_if_empty:NF \l_stex_import_path_str {
1650         \str_set:Nx \l_stex_import_ns_str {
1651           \l_stex_import_ns_str / \l_stex_import_path_str
1652         }
1653       }
1654     }
1655   }
1656 }
```

(End definition for `\stex_import_module_uri:nn`. This function is documented on page 43.)

|   |  |
|---|--|
| <code>\l_stex_import_name_str</code>    | Store the return values of <code>\stex_import_module_uri:nn</code> . |
| <code>\l_stex_import_archive_str</code> | 1657 <code>\str_new:N \l_stex_import_name_str</code>                 |
| <code>\l_stex_import_path_str</code>    | 1658 <code>\str_new:N \l_stex_import_archive_str</code>              |
| <code>\l_stex_import_ns_str</code>      | 1659 <code>\str_new:N \l_stex_import_path_str</code>                 |

```
1660 \str_new:N \l_stex_import_ns_str
```

(End definition for `\l_stex_import_name_str` and others. These variables are documented on page 43.)

```
\stex_import_require_module:nnnnn {{ns}} {{archive-ID}} {{path}} {{name}}

1661 \cs_new_protected:Nn \stex_import_require_module:nnnnn {
1662   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1663
1664     % archive
1665     \str_set:Nx \l_tmpa_str { #2 }
1666     \str_if_empty:NTF \l_tmpa_str {
1667       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1668     } {
1669       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1670       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1671       \seq_put_right:Nn \l_tmpa_seq { source }
1672     }
1673
1674     % path
1675     \str_set:Nx \l_tmpb_str { #3 }
1676     \str_if_empty:NTF \l_tmpb_str {
1677       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1678
1679       \ltx@ifpackageloaded{babel} {
1680         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1681           { \language } \l_tmpb_str {
1682           \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1683         }
1684       } {
1685         \str_clear:N \l_tmpb_str
1686       }
1687
1688       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1689       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1690         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1691       }{
1692         \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1693         \IfFileExists{ \l_tmpa_str.tex }{
1694           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1695         }{
1696           % try english as default
1697           \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1698           \IfFileExists{ \l_tmpa_str.en.tex }{
1699             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1700           }{
1701             \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1702           }
1703         }
1704       }
1705
1706     } {
1707       \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1708       \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1709
```

```

1710 \ltx@ifpackageloaded{babel} {
1711   \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1712     { \language } \l_tmpb_str {
1713       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1714     }
1715   } {
1716     \str_clear:N \l_tmpb_str
1717   }
1718
1719   \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1720
1721   \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1722   \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1723     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1724   }{
1725     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1726     \IfFileExists{ \l_tmpa_str/#4.tex }{
1727       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1728     }{
1729       % try english as default
1730       \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1731       \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1732         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1733       }{
1734         \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1735         \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1736           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1737         }{
1738           \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1739           \IfFileExists{ \l_tmpa_str.tex }{
1740             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1741           }{
1742             % try english as default
1743             \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1744             \IfFileExists{ \l_tmpa_str.en.tex }{
1745               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1746             }{
1747               \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1748             }
1749           }
1750         }
1751       }
1752     }
1753   }
1754 }
1755
1756 \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
1757   \seq_clear:N \l_stex_all_modules_seq
1758   \str_clear:N \l_stex_current_module_str
1759   \str_set:Nx \l_tmpb_str { #2 }
1760   \str_if_empty:NF \l_tmpb_str {
1761     \stex_set_current_repository:n { #2 }
1762   }
1763   \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}

```



```

1764     }
1765
1766     \stex_if_module_exists:nF { #1 ? #4 } {
1767         \msg_error:nnx{stex}{error/unknownmodule}{
1768             #1?#4~(in~file~\g__stex_importmodule_file_str)
1769         }
1770     }
1771 }
1772 \stex_activate_module:n { #1 ? #4 }
1773 }

```

(End definition for `\stex_import_require_module:nnnn`. This function is documented on page 43.)

### `\importmodule`

```

1774 \NewDocumentCommand \importmodule { 0{} m } {
1775     \stex_import_module_uri:nn { #1 } { #2 }
1776     \stex_debug:nn{modules}{Importing~module:~
1777         \l_stex_import_ns_str ? \l_stex_import_name_str
1778     }
1779     \stex_if_smsmode:F {
1780         \stex_import_require_module:nnnn
1781         { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1782         { \l_stex_import_path_str } { \l_stex_import_name_str }
1783         \stex_annotate_invisible:nnn
1784         {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1785     }
1786     \exp_args:Nx \stex_add_to_current_module:n {
1787         \stex_import_require_module:nnnn
1788         { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1789         { \l_stex_import_path_str } { \l_stex_import_name_str }
1790     }
1791     \exp_args:Nx \stex_add_import_to_current_module:n {
1792         \l_stex_import_ns_str ? \l_stex_import_name_str
1793     }
1794     \stex_smsmode_do:
1795     \ignorespacesandpars
1796 }
1797 \stex_deactivate_macro:Nn \importmodule {module-environments}

```

(End definition for `\importmodule`. This function is documented on page 42.)

### `\usemodule`

```

1798 \NewDocumentCommand \usemodule { 0{} m } {
1799     \stex_if_smsmode:F {
1800         \stex_import_module_uri:nn { #1 } { #2 }
1801         \stex_import_require_module:nnnn
1802         { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1803         { \l_stex_import_path_str } { \l_stex_import_name_str }
1804         \stex_annotate_invisible:nnn
1805         {usemodule} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1806     }
1807     \stex_smsmode_do:
1808     \ignorespacesandpars
1809 }

```

*(End definition for \usemodule. This function is documented on page 42.)*

1810 `\endpackage`

## Chapter 28

# STEX -Symbols Implementation

```
1811 <*package>
1812
1813 %%%%%%%%%% symbols.dtx %%%%%%%%%%
1814
1815 Warnings and error messages
1816 \msg_new:nnn{stex}{error/wrongargs}{
1817   args~value~in~symbol~declaration~for~#1~
1818   needs~to~be~i,~a,~b~or~B,~but~#2~given
1819 }
1820 \msg_new:nnn{stex}{error/unknownsymbol}{
1821   No~symbol~#1~found!
1822 }
1823 \msg_new:nnn{stex}{error/seqlength}{
1824   Expected~#1~arguments;~got~#2!
1825 }
```

### 28.1 Symbol Declarations

```
1825 <@@=stex_symdecl>
\stex_all_symbols:n Map over all available symbols
1826 \cs_new_protected:Nn \stex_all_symbols:n {
1827   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
1828   \seq_map_inline:Nn \l_stex_all_modules_seq {
1829     \seq_map_inline:cn{c_stex_module_##1_constants}{
1830       \__stex_symdecl_all_symbols_cs{##1?####1}
1831     }
1832   }
1833 }
(End definition for \stex_all_symbols:n. This function is documented on page 45.)
\STEXsymbol
1834 \NewDocumentCommand \STEXsymbol { m } {
1835   \stex_get_symbol:n { #1 }
```

```

1836 \exp_args:No
1837 \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1838 }

```

(End definition for `\STEXsymbol`. This function is documented on page 46.)

`symdecl` arguments:

```

1839 \keys_define:nn { stex / symdecl } {
1840   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1841   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1842   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1843   type      .tl_set:N = \l_stex_symdecl_type_tl ,
1844   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,
1845   align     .str_set:N = \l_stex_symdecl_align_str , % TODO(?)
1846   gfc       .str_set:N = \l_stex_symdecl_gfc_str , % TODO(?)
1847   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
1848   def       .tl_set:N = \l_stex_symdecl_definiens_tl ,
1849   assoc     .choices:nn =
1850     {bin,binl,binr,pre,conj,pwconj}
1851     {\str_set:Nx \l_stex_symdecl_astype_str {\l_keys_choice_tl}}
1852 }
1853
1854 \bool_new:N \l_stex_symdecl_make_macro_bool
1855
1856 \cs_new_protected:Nn \__stex_symdecl_args:n {
1857   \str_clear:N \l_stex_symdecl_name_str
1858   \str_clear:N \l_stex_symdecl_args_str
1859   \str_clear:N \l_stex_symdecl_deprecate_str
1860   \str_clear:N \l_stex_symdecl_astype_str
1861   \bool_set_false:N \l_stex_symdecl_local_bool
1862   \tl_clear:N \l_stex_symdecl_type_tl
1863   \tl_clear:N \l_stex_symdecl_definiens_tl
1864
1865   \keys_set:nn { stex / symdecl } { #1 }
1866 }

```

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

```

1867
1868 \NewDocumentCommand \symdecl { s m O{} } {
1869   \__stex_symdecl_args:n { #3 }
1870   \IfBooleanTF #1 {
1871     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1872   } {
1873     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1874   }
1875   \stex_symdecl_do:n { #2 }
1876   \stex_smsmode_do:
1877 }
1878
1879 \cs_new_protected:Nn \stex_symdecl_do:nn {
1880   \__stex_symdecl_args:n{#1}
1881   \bool_set_false:N \l_stex_symdecl_make_macro_bool
1882   \stex_symdecl_do:n{#2}
1883 }

```

```

1884
1885 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

(End definition for \symdecl. This function is documented on page 44.)

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

```

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

```

```

1934     \bool_set_false:N \l_tmpa_bool
1935     \int_incr:N \l_tmpb_int
1936   }
1937 }{
1938   \msg_error:nnxx{stex}{error/wrongargs}{
1939     \l_stex_current_module_str ?
1940     \l_stex_symdecl_name_str
1941   }{##1}
1942 }
1943 }
1944 \bool_if:NTF \l_tmpa_bool {
1945   % possibly numeric
1946   \str_if_empty:NTF \l_stex_symdecl_args_str {
1947     \prop_put:Nnn \l_tmpa_prop { args } {}
1948     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1949   }{
1950     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1951     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1952     \str_clear:N \l_tmpa_str
1953     \int_step_inline:nn \l_tmpa_int {
1954       \str_put_right:Nn \l_tmpa_str i
1955     }
1956     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1957   }
1958 } {
1959   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1960   \prop_put:Nnx \l_tmpa_prop { arity }
1961     { \str_count:N \l_stex_symdecl_args_str }
1962 }
1963 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
1964
1965 % semantic macro
1966
1967 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1968   \exp_args:Nx \stex_do_up_to_module:n {
1969     \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1970       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1971     }}
1972   }
1973 }
1974
1975 \bool_if:NF \l_stex_symdecl_local_bool {
1976   \exp_args:Nx \stex_add_to_current_module:n {
1977     \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1978       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1979     } }
1980   }
1981 }
1982 }
1983
1984 \stex_debug:nn{symbols}{New~symbol:~
1985   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
1986   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
1987   Args:~\prop_item:Nn \l_tmpa_prop { args }^^J

```

```

1988     Definiens:~\exp_not:o {\l_stex_symdecl_definiens_tl}
1989 }
1990
1991 % circular dependencies require this:
1992
1993 \prop_if_exist:cF {
1994     \l_stex_symdecl_
1995     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1996     _prop
1997 } {
1998     \exp_args:Nx \stex_do_up_to_module:n {
1999         \prop_set_from_keyval:cn {
2000             \l_stex_symdecl_
2001             \l_stex_current_module_str ? \l_stex_symdecl_name_str
2002             _prop
2003         } {\prop_to_keyval:N \l_tmpa_prop}
2004         \seq_clear:c {
2005             \l_stex_symdecl_
2006             \l_stex_current_module_str ? \l_stex_symdecl_name_str
2007             _notations
2008         }
2009     }
2010 }
2011
2012
2013
2014 \bool_if:NF \l_stex_symdecl_local_bool {
2015     \exp_args:Nx
2016     \stex_add_to_current_module:n {
2017         \seq_clear:c {
2018             \l_stex_symdecl_
2019             \l_stex_current_module_str ? \l_stex_symdecl_name_str
2020             _notations
2021         }
2022         \prop_set_from_keyval:cn {
2023             \l_stex_symdecl_
2024             \l_stex_current_module_str ? \l_stex_symdecl_name_str
2025             _prop
2026         } {
2027             name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2028             module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2029             type       = \prop_item:Nn \l_tmpa_prop { type }     ,
2030             args       = \prop_item:Nn \l_tmpa_prop { args }     ,
2031             arity      = \prop_item:Nn \l_tmpa_prop { arity }    ,
2032             assocs     = \prop_item:Nn \l_tmpa_prop { assocs }   ,
2033         }
2034     }
2035 }
2036
2037 \stex_if_smsmode:F {
2038 %     \exp_args:Nx \stex_do_up_to_module:n {
2039 %         \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2040 %             \l_stex_current_module_str ? \l_stex_symdecl_name_str
2041 %         }

```

```

2042 %    }
2043 \stex_if_do_html:T {
2044   \stex_annotate_invisible:nnn {symdecl} {
2045     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2046   } {
2047     \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}{}}{\l_st
2048     \stex_annotate_invisible:nnn{args}{}}{
2049       \prop_item:Nn \l_tmpa_prop { args }
2050     }
2051     \stex_annotate_invisible:nnn{macroname}{#1}{}
2052     \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2053       \stex_annotate_invisible:nnn{definiens}{}
2054       {\l_stex_symdecl_definiens_tl$}
2055     }
2056     \str_if_empty:NF \l_stex_symdecl_assoc_type_str {
2057       \stex_annotate_invisible:nnn{assoc_type}{\l_stex_symdecl_assoc_type_str}{}
2058     }
2059   }
2060 }
2061 }
2062 }

```

(End definition for `\stex_symdecl_do:n`. This function is documented on page 45.)

`\stex_get_symbol:n`

```

2063 \str_new:N \l_stex_get_symbol_uri_str
2064
2065 \cs_new_protected:Nn \stex_get_symbol:n {
2066   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2067     \tl_set:Nn \l_tmpa_tl { #1 }
2068     \__stex_symdecl_get_symbol_from_cs:
2069   }{
2070     % argument is a string
2071     % is it a command name?
2072     \cs_if_exist:cTF { #1 }{
2073       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2074       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2075       \str_if_empty:NTF \l_tmpa_str {
2076         \exp_args:Nx \cs_if_eq:NNTF {
2077           \tl_head:N \l_tmpa_tl
2078         } \stex_invoke_symbol:n {
2079           \__stex_symdecl_get_symbol_from_cs:
2080         }{
2081           \__stex_symdecl_get_symbol_from_string:n { #1 }
2082         }
2083       } {
2084         \__stex_symdecl_get_symbol_from_string:n { #1 }
2085       }
2086     }{
2087       % argument is not a command name
2088       \__stex_symdecl_get_symbol_from_string:n { #1 }
2089       % \l_stex_all_symbols_seq
2090     }
2091   }

```



```

2092 \str_if_eq:eeF {
2093   \prop_item:cn {
2094     l_stex_symdecl\l_stex_get_symbol_uri_str _prop
2095   }{ deprecate }
2096 }{}{
2097   \msg_warning:nnxx{stex}{warning/deprecated}{
2098     Symbol~\l_stex_get_symbol_uri_str
2099   }{
2100     \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{ deprecate }
2101   }
2102 }
2103 }
2104
2105 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2106   \tl_set:Nn \l_tmpa_tl {
2107     \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2108   }
2109   \str_set:Nn \l_tmpa_str { #1 }
2110   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2111
2112   \stex_all_symbols:n {
2113     \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2114       \seq_map_break:n{\seq_map_break:n{
2115         \tl_set:Nn \l_tmpa_tl {
2116           \str_set:Nn \l_stex_get_symbol_uri_str { ##1 }
2117         }
2118       }}
2119     }
2120   }
2121
2122   \l_tmpa_tl
2123 }
2124
2125 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
2126   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2127   { \tl_tail:N \l_tmpa_tl }
2128   \tl_if_single:NTF \l_tmpa_tl {
2129     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2130       \exp_after:wN \str_set:Nn \exp_after:wN
2131       \l_stex_get_symbol_uri_str \l_tmpa_tl
2132     }{
2133       % TODO
2134       % tail is not a single group
2135     }
2136   }{
2137     % TODO
2138     % tail is not a single group
2139   }
2140 }

```

(End definition for `\stex_get_symbol:n`. This function is documented on page 45.)

## 28.2 Notations

```

2141 <@@=stex_notation>
      notation arguments:
2142 \keys_define:nn { stex / notation } {
2143   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2144   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2145   prec .str_set_x:N = \l__stex_notation_prec_str ,
2146   op .tl_set:N = \l__stex_notation_op_tl ,
2147   primary .bool_set:N = \l__stex_notation_primary_bool ,
2148   primary .default:n = {true} ,
2149   unknown .code:n = \str_set:Nx
2150     \l__stex_notation_variant_str \l_keys_key_str
2151 }
2152
2153 \cs_new_protected:Nn \stex_notation_args:n {
2154   \str_clear:N \l__stex_notation_lang_str
2155   \str_clear:N \l__stex_notation_variant_str
2156   \str_clear:N \l__stex_notation_prec_str
2157   \tl_clear:N \l__stex_notation_op_tl
2158   \bool_set_false:N \l__stex_notation_primary_bool
2159
2160   \keys_set:nn { stex / notation } { #1 }
2161 }

```

## **\notation**

```

2162 \NewDocumentCommand \notation { s m O{}} {
2163   \stex_notation_args:n { #3 }
2164   \tl_clear:N \l_stex_symdecl_definiens_tl
2165   \stex_get_symbol:n { #2 }
2166   \tl_set:Nn \l_stex_notation_after_do_tl {
2167     \__stex_notation_final:
2168     \IfBooleanTF#1{
2169       \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2170     }{}
2171     \stex_smsmode_do:\ignorespacesandpars
2172   }
2173   \stex_notation_do:nnnnn
2174   { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } }
2175   { \prop_item:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } }
2176   { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2177   { \l__stex_notation_prec_str }
2178 }
2179 \stex_deactivate_macro:Nn \notation {module~environments}

```

(End definition for \notation. This function is documented on page 45.)

## **\stex\_notation\_do:nnnnn**

```

2180 \seq_new:N \l__stex_notation_precedences_seq
2181 \tl_new:N \l__stex_notation_opprec_tl
2182 \int_new:N \l__stex_notation_currarg_int
2183 \tl_new:N \stex_symbol_after_invokation_tl
2184
2185 \cs_new_protected:Nn \stex_notation_do:nnnnn {
2186   \let\l_stex_current_symbol_str\relax
2187   \seq_clear:N \l__stex_notation_precedences_seq

```

```

2188 \tl_clear:N \l__stex_notation_opprec_tl
2189 \str_set:Nx \l__stex_notation_args_str { #1 }
2190 \str_set:Nx \l__stex_notation_arity_str { #2 }
2191 \str_set:Nx \l__stex_notation_suffix_str { #3 }
2192 \str_set:Nx \l__stex_notation_prec_str { #4 }
2193
2194 % precedences
2195 \str_if_empty:NTF \l__stex_notation_prec_str {
2196   \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2197     \tl_set:No \l__stex_notation_opprec_tl { \neginfpref }
2198   }{
2199     \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2200   }
2201 } {
2202   \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2203     \tl_set:No \l__stex_notation_opprec_tl { \neginfpref }
2204     \int_step_inline:nn { \l__stex_notation_arity_str } {
2205       \exp_args:NNo
2206       \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2207     }
2208   }{
2209     \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2210     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2211       \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2212       \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2213         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2214         \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2215         \seq_map_inline:Nn \l_tmpa_seq {
2216           \seq_put_right:Nn \l_tmpb_seq { ##1 }
2217         }
2218       }
2219     }{
2220       \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2221         \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2222       }{
2223         \tl_set:No \l__stex_notation_opprec_tl { 0 }
2224       }
2225     }
2226   }
2227 }
2228
2229 \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2230 \int_step_inline:nn { \l__stex_notation_arity_str } {
2231   \seq_pop_left:NNTF \l_tmpa_seq \l_tmpb_str {
2232     \exp_args:NNo
2233     \seq_put_right:No \l__stex_notation_precedences_seq {
2234       \l__stex_notation_opprec_tl
2235     }
2236   }
2237 }
2238 \tl_clear:N \l_stex_notation_dummyargs_tl
2239
2240 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2241   \exp_args:NNe

```

```

2242 \cs_set:Npn \l_stex_notation_macrocode_cs {
2243   \_stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2244   { \l__stex_notation_suffix_str }
2245   { \l__stex_notation_opprec_tl }
2246   { \exp_not:n { #5 } }
2247 }
2248 \l_stex_notation_after_do_tl
2249 }{
2250 \str_if_in:NnTF \l_stex_notation_args_str b {
2251   \exp_args:Nne \use:nn
2252   {
2253     \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2254     \cs_set:Npn \l__stex_notation_arity_str } { {
2255       \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2256       { \l__stex_notation_suffix_str }
2257       { \l__stex_notation_opprec_tl }
2258       { \exp_not:n { #5 } }
2259     }}
2260 }{
2261 \str_if_in:NnTF \l__stex_notation_args_str B {
2262   \exp_args:Nne \use:nn
2263   {
2264     \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2265     \cs_set:Npn \l__stex_notation_arity_str } { {
2266       \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2267       { \l__stex_notation_suffix_str }
2268       { \l__stex_notation_opprec_tl }
2269       { \exp_not:n { #5 } }
2270     } }
2271 }{
2272   \exp_args:Nne \use:nn
2273   {
2274     \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2275     \cs_set:Npn \l__stex_notation_arity_str } { {
2276       \_stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2277       { \l__stex_notation_suffix_str }
2278       { \l__stex_notation_opprec_tl }
2279       { \exp_not:n { #5 } }
2280     } }
2281   }
2282 }
2283
2284 \str_set_eq:NN \l__stex_notation_remaining_args_str \l_stex_notation_args_str
2285 \int_zero:N \l__stex_notation_currarg_int
2286 \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2287 \l__stex_notation_arguments:
2288 }
2289 }

```

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

`\__stex_notation_arguments:` Takes care of annotating the arguments in a notation macro

```

2290 \cs_new_protected:Nn \__stex_notation_arguments: {
2291   \int_incr:N \l__stex_notation_currarg_int

```

```

2292 \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2293   \l_stex_notation_after_do_tl
2294 }{
2295   \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2296   \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2297   \str_if_eq:VnTF \l_tmpa_str a {
2298     \__stex_notation_argument_assoc:n
2299   }{
2300     \str_if_eq:VnTF \l_tmpa_str B {
2301       \__stex_notation_argument_assoc:n
2302     }{
2303       \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2304       \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2305         { \stex_term_math_arg:nnn
2306           { \int_use:N \l__stex_notation_currarg_int }
2307           { \l_tmpa_str }
2308           { ####\int_use:N \l__stex_notation_currarg_int }
2309         }
2310       }
2311       \__stex_notation_arguments:
2312     }
2313   }
2314 }
2315 }

```

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

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

```

2316 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
2317
2318   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2319     {\l__stex_notation_arity_str}{
2320       #1
2321     }
2322   \int_zero:N \l_tmpa_int
2323   \tl_clear:N \l_tmpa_tl
2324   \str_map_inline:Nn \l__stex_notation_args_str {
2325     \int_incr:N \l_tmpa_int
2326     \tl_put_right:Nx \l_tmpa_tl {
2327       \str_if_eq:nnTF {##1}{a}{ {} }{
2328         \str_if_eq:nnTF {##1}{B}{ {} }{
2329           {\stex_term_arg:nn{\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa_in
2330         }
2331       }
2332     }
2333   }
2334   \exp_after:wN\exp_after:wN\exp_after:wN \def
2335   \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2336   \exp_after:wN\exp_after:wN\exp_after:wN ##
2337   \exp_after:wN\exp_after:wN\exp_after:wN 1
2338   \exp_after:wN\exp_after:wN\exp_after:wN ##
2339   \exp_after:wN\exp_after:wN\exp_after:wN 2
2340   \exp_after:wN\exp_after:wN\exp_after:wN {
2341     \exp_after:wN \exp_after:wN \exp_after:wN

```

```

2342 \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2343 \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2344 }
2345 }
2346
2347 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2348 \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
2349 \stex_term_math_assoc_arg:nnnn
2350 { \int_use:N \l__stex_notation_currarg_int }
2351 { \l_tmpa_str }
2352 { ####\int_use:N \l__stex_notation_currarg_int }
2353 { \l_tmpa_cs {####1} {####2} }
2354 } }
2355 \__stex_notation_arguments:
2356 }

```

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

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

```

2357 \cs_new_protected:Nn \__stex_notation_final: {
2358 \exp_args:Nne \use:nn
2359 {
2360 \cs_generate_from_arg_count:cNnn {
2361 stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2362 \l__stex_notation_suffix_str
2363 _cs
2364 }
2365 \cs_set:Npn \l__stex_notation_arity_str } { {
2366 \exp_after:wN \exp_after:wN \exp_after:wN
2367 \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2368 { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2369 } }
2370
2371 \tl_if_empty:NF \l__stex_notation_op_tl {
2372 \cs_set:cpx {
2373 stex_op_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2374 \l__stex_notation_suffix_str
2375 _cs
2376 } { \exp_not:N \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2377 }
2378
2379 \exp_args:Ne
2380 \stex_add_to_current_module:n {
2381 \cs_generate_from_arg_count:cNnn {
2382 stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2383 \l__stex_notation_suffix_str
2384 _cs
2385 } \cs_set:Npn {\l__stex_notation_arity_str} {
2386 \exp_after:wN \exp_after:wN \exp_after:wN
2387 \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2388 { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2389 }
2390 \tl_if_empty:NF \l__stex_notation_op_tl {
2391 \cs_set:cpn {

```

```

2392         stex_op_notation_\l_stex_get_symbol_uri_str \c_hash_str
2393         \l__stex_notation_suffix_str
2394         _cs
2395     } { \exp_not:N \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2396 }
2397 }
2398 %\exp_args:Nx
2399 % \stex_do_up_to_module:n {
2400     \seq_put_right:cx {
2401         l_stex_symdecl_ \l_stex_get_symbol_uri_str
2402         _notations
2403     } {
2404         \l__stex_notation_suffix_str
2405     }
2406 % }
2407
2408 \stex_debug:nn{symbols}{
2409     Notation~\l__stex_notation_suffix_str
2410     ~for~\l_stex_get_symbol_uri_str^^J
2411     Operator~precedence:~\l__stex_notation_opprec_tl^^J
2412     Argument~precedences:~
2413     \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2414     Notation: \cs_meaning:c {
2415         stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2416         \l__stex_notation_suffix_str
2417         _cs
2418     }
2419 }
2420
2421 \exp_args:Ne
2422 \stex_add_to_current_module:n {
2423     \seq_put_right:cn {
2424         l_stex_symdecl_\l_stex_get_symbol_uri_str
2425         _notations
2426     } { \l__stex_notation_suffix_str }
2427 }
2428
2429 \stex_if_smsmode:F {
2430
2431     % HTML annotations
2432     \stex_if_do_html:T {
2433         \stex_annotate_invisible:nnn { notation }
2434         { \l_stex_get_symbol_uri_str } {
2435             \stex_annotate_invisible:nnn { notationfragment }
2436             { \l__stex_notation_suffix_str }{}
2437             \stex_annotate_invisible:nnn { precedence }
2438             { \l__stex_notation_prec_str }{}
2439
2440             \int_zero:N \l_tmpa_int
2441             \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2442             \tl_clear:N \l_tmpa_tl
2443             \int_step_inline:nn { \l__stex_notation_arity_str }{
2444                 \int_incr:N \l_tmpa_int
2445                 \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }

```

```

2446 \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_r
2447 \str_if_eq:VnTF \l_tmpb_str a {
2448 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2449 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2450 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2451 } }
2452 }{
2453 \str_if_eq:VnTF \l_tmpb_str B {
2454 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2455 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2456 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2457 } }
2458 }{
2459 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2460 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2461 } }
2462 }
2463 }
2464 }
2465 \stex_annotate_invisible:nnn { notationcomp }{}{
2466 \str_set:Nx \l_stex_current_symbol_str { \l_stex_get_symbol_uri_str }
2467 $ \exp_args:Nno \use:nn { \use:c {
2468 stex_notation_ \l_stex_current_symbol_str
2469 \c_hash_str \l__stex_notation_suffix_str _cs
2470 } } { \l_tmpa_tl } $
2471 }
2472 }
2473 }
2474 }
2475 }

```

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

\setnotation

```

2476 \keys_define:nn { stex / setnotation } {
2477 lang .tl_set_x:N = \l__stex_notation_lang_str ,
2478 variant .tl_set_x:N = \l__stex_notation_variant_str ,
2479 unknown .code:n = \str_set:Nx
2480 \l__stex_notation_variant_str \l_keys_key_str
2481 }
2482
2483 \cs_new_protected:Nn \stex_setnotation_args:n {
2484 \str_clear:N \l__stex_notation_lang_str
2485 \str_clear:N \l__stex_notation_variant_str
2486 \keys_set:nn { stex / setnotation } { #1 }
2487 }
2488
2489 \cs_new_protected:Nn \stex_setnotation:n {
2490 \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1 _notations }
2491 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2492 \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2493 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2494 \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2495 { \c_hash_str }

```



```

2496 \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_#1 _notations }
2497 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2498 \exp_args:Nx \stex_add_to_current_module:n {
2499   \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2500   { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2501   \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_#1 _notations }
2502   { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2503   \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2504   { \c_hash_str }
2505 }
2506 \stex_debug:nn {notations}{
2507   Setting~default~notation~
2508   {\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str}~for~
2509   #1 \
2510   \expandafter\meaning\csname
2511   l_stex_symdecl_#1 _notations\endcsname
2512 }
2513 }{
2514   % todo throw error
2515 }
2516 }
2517
2518 \NewDocumentCommand \setnotation {m m} {
2519   \stex_get_symbol:n { #1 }
2520   \_stex_setnotation_args:n { #2 }
2521   \stex_setnotation:n{\l_stex_get_symbol_uri_str}
2522   \stex_smsmode_do:\ignorespacesandpars
2523 }
2524
2525 \cs_new_protected:Nn \stex_copy_notations:nn {
2526   \stex_debug:nn {notations}{
2527     Copying~notations~from~#2~to~#1\
2528     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
2529   }
2530   \tl_clear:N \l_tmpa_tl
2531   \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2532     \tl_put_right:Nn \l_tmpa_tl { {## ##1} }
2533   }
2534   \seq_map_inline:cn {l_stex_symdecl_#2_notations}{
2535     \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2536     \edef \l_tmpa_tl {
2537       \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2538       \exp_after:wN\exp_after:wN\exp_after:wN {
2539         \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2540       }
2541     }
2542     \exp_args:Nx
2543     \stex_do_up_to_module:n {
2544       \seq_put_right:cn{l_stex_symdecl_#1_notations}{##1}
2545       \cs_generate_from_arg_count:cNnn {
2546         stex_notation_ #1 \c_hash_str ##1 _cs
2547       } \cs_set:Npn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }{
2548         \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl}
2549       }

```

```

2550     }
2551   }
2552 }
2553
2554 \NewDocumentCommand \copynotation {m m} {
2555   \stex_get_symbol:n { #1 }
2556   \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2557   \stex_get_symbol:n { #2 }
2558   \exp_args:Noo
2559   \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2560   \exp_args:Nx \stex_add_import_to_current_module:n{
2561     \stex_copy_notations:nn {\l_tmpa_str} {\l_stex_get_symbol_uri_str}
2562   }
2563   \stex_smsmode_do:\ignorespacesandpars
2564 }
2565

```

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

### \symdef

```

2566 \keys_define:nn { stex / symdef } {
2567   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2568   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2569   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2570   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
2571   def       .tl_set:N    = \l_stex_symdecl_definiens_tl ,
2572   op        .tl_set:N    = \l__stex_notation_op_tl ,
2573   lang      .str_set_x:N = \l__stex_notation_lang_str ,
2574   variant   .str_set_x:N = \l__stex_notation_variant_str ,
2575   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2576   assoc     .choices:nn =
2577     {bin,binl,binr,pre,conj,pwconj}
2578     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
2579   unknown   .code:n      = \str_set:Nx
2580     \l__stex_notation_variant_str \l_keys_key_str
2581 }
2582
2583 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2584   \str_clear:N \l_stex_symdecl_name_str
2585   \str_clear:N \l_stex_symdecl_args_str
2586   \str_clear:N \l_stex_symdecl_assoctype_str
2587   \bool_set_false:N \l_stex_symdecl_local_bool
2588   \tl_clear:N \l_stex_symdecl_type_tl
2589   \tl_clear:N \l_stex_symdecl_definiens_tl
2590   \str_clear:N \l__stex_notation_lang_str
2591   \str_clear:N \l__stex_notation_variant_str
2592   \str_clear:N \l__stex_notation_prec_str
2593   \tl_clear:N \l__stex_notation_op_tl
2594
2595   \keys_set:nn { stex / symdef } { #1 }
2596 }
2597
2598 \NewDocumentCommand \symdef { m O{} } {
2599   \__stex_notation_symdef_args:n { #2 }

```

```

2600 \bool_set_true:N \l_stex_symdecl_make_macro_bool
2601 \stex_symdecl_do:n { #1 }
2602 \tl_set:Nn \l_stex_notation_after_do_tl {
2603   \__stex_notation_final:
2604   \stex_smsmode_do:\ignorespacesandpars
2605 }
2606 \str_set:Nx \l_stex_get_symbol_uri_str {
2607   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2608 }
2609 \exp_args:Nx \stex_notation_do:nnnnn
2610 { \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str _prop } { args } }
2611 { \prop_item:cn { l_stex_symdecl\l_stex_get_symbol_uri_str _prop } { arity } }
2612 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2613 { \l__stex_notation_prec_str}
2614 }
2615 \stex_deactivate_macro:Nn \symdef {module~environments}

```

(End definition for \symdef. This function is documented on page 45.)

## 28.3 Variables

```

2616 <@@=stex_variables>
2617
2618 \keys_define:nn { stex / vardef } {
2619   name      .str_set_x:N = \l__stex_variables_name_str ,
2620   args      .str_set_x:N = \l__stex_variables_args_str ,
2621   type      .tl_set:N    = \l__stex_variables_type_tl ,
2622   def       .tl_set:N    = \l__stex_variables_def_tl ,
2623   op        .tl_set:N    = \l__stex_variables_op_tl ,
2624   prec      .str_set_x:N = \l__stex_variables_prec_str ,
2625   assoc     .choices:nn =
2626     {bin,binl,binr,pre,conj,pwconj}
2627     {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
2628   bind      .choices:nn =
2629     {forall,exists}
2630     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2631 }
2632
2633 \cs_new_protected:Nn \__stex_variables_args:n {
2634   \str_clear:N \l__stex_variables_name_str
2635   \str_clear:N \l__stex_variables_args_str
2636   \str_clear:N \l__stex_variables_prec_str
2637   \str_clear:N \l__stex_variables_assoctype_str
2638   \str_clear:N \l__stex_variables_bind_str
2639   \tl_clear:N \l__stex_variables_type_tl
2640   \tl_clear:N \l__stex_variables_def_tl
2641   \tl_clear:N \l__stex_variables_op_tl
2642
2643   \keys_set:nn { stex / vardef } { #1 }
2644 }
2645
2646 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{} } {
2647   \__stex_variables_args:n {#2}
2648   \str_if_empty:NT \l__stex_variables_name_str {

```

```

2649     \str_set:Nx \l__stex_variables_name_str { #1 }
2650 }
2651 \prop_clear:N \l_tmpa_prop
2652 \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
2653
2654 \int_zero:N \l_tmpb_int
2655 \bool_set_true:N \l_tmpa_bool
2656 \str_map_inline:Nn \l__stex_variables_args_str {
2657     \token_case_meaning:NnF ##1 {
2658         0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2659         {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2660         {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2661         {\tl_to_str:n a} {
2662             \bool_set_false:N \l_tmpa_bool
2663             \int_incr:N \l_tmpb_int
2664         }
2665         {\tl_to_str:n B} {
2666             \bool_set_false:N \l_tmpa_bool
2667             \int_incr:N \l_tmpb_int
2668         }
2669     }{
2670         \msg_error:nnxx{stex}{error/wrongargs}{
2671             variable~\l__stex_variables_name_str
2672         }{##1}
2673     }
2674 }
2675 \bool_if:NTF \l_tmpa_bool {
2676     % possibly numeric
2677     \str_if_empty:NTF \l__stex_variables_args_str {
2678         \prop_put:Nnn \l_tmpa_prop { args } {}
2679         \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2680     }{
2681         \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
2682         \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2683         \str_clear:N \l_tmpa_str
2684         \int_step_inline:nn \l_tmpa_int {
2685             \str_put_right:Nn \l_tmpa_str i
2686         }
2687         \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
2688         \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2689     }
2690 } {
2691     \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2692     \prop_put:Nnx \l_tmpa_prop { arity }
2693     { \str_count:N \l__stex_variables_args_str }
2694 }
2695 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
2696 \tl_set:cx { #1 }{ \stex_invoke_variable:n { \l__stex_variables_name_str } }
2697
2698 \prop_set_eq:cN { l_stex_variable_\l__stex_variables_name_str_prop } \l_tmpa_prop
2699
2700 \tl_if_empty:NF \l__stex_variables_op_tl {
2701     \cs_set:cpx {
2702         stex_var_op_notation_\l__stex_variables_name_str_cs

```

```

2703     } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
2704 }
2705
2706 \tl_set:Nn \l_stex_notation_after_do_tl {
2707   \exp_args:Nne \use:nn {
2708     \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str _cs }
2709     \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
2710   } {{
2711     \exp_after:wN \exp_after:wN \exp_after:wN
2712     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2713     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_symbol
2714     }}
2715   \stex_if_do_html:T {
2716     \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
2717       \stex_annotate_invisible:nnn { precedence }
2718       { \l__stex_variables_prec_str }{}
2719       \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{}{${\l
2720       \stex_annotate_invisible:nnn{args}{}{ \l__stex_variables_args_str }
2721       \stex_annotate_invisible:nnn{macroname}{#1}{}
2722       \tl_if_empty:NF \l__stex_variables_def_tl {
2723         \stex_annotate_invisible:nnn{definiens}{}
2724         {${\l__stex_variables_def_tl$}
2725       }
2726       \str_if_empty:NF \l__stex_variables_assoctype_str {
2727         \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{}
2728       }
2729       \int_zero:N \l_tmpa_int
2730       \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
2731       \tl_clear:N \l_tmpa_tl
2732       \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{{
2733         \int_incr:N \l_tmpa_int
2734         \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
2735         \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables
2736         \str_if_eq:VnTF \l_tmpb_str a {
2737           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2738             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2739             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2740           } }
2741         }{
2742           \str_if_eq:VnTF \l_tmpb_str B {
2743             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2744               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2745               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2746             } }
2747           }{
2748             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2749               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2750             } }
2751         }
2752       }
2753     }
2754     \stex_annotate_invisible:nnn { notationcomp }{}{
2755       \str_set:Nx \l_stex_current_symbol_str {var://\l__stex_variables_name_str }
2756       $ \exp_args:Nno \use:nn { \use:c {

```

```

2757         stex_var_notation_\l__stex_variables_name_str _cs
2758     } } { \l_tmpa_tl } $
2759     }
2760 }
2761 }\ignorespacesandpars
2762 }
2763
2764 \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
2765 }
2766
2767 \cs_new:Nn \_stex_reset:N {
2768     \tl_if_exist:NTF #1 {
2769         \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
2770     }{
2771         \let \exp_not:N #1 \exp_not:N \undefined
2772     }
2773 }
2774
2775 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
2776     \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
2777     \exp_args:Nnx \use:nn {
2778         % TODO
2779         \stex_annotate_invisible:nnn {vardecls}{\clist_use:Nn\l__stex_variables_names,}{
2780             #2
2781         }
2782     }{
2783         \_stex_reset:N \varnot
2784         \_stex_reset:N \vartype
2785         \_stex_reset:N \vardefi
2786     }
2787 }
2788
2789 \NewDocumentCommand \vardef { s } {
2790     \IfBooleanTF#1 {
2791         \__stex_variables_do_complex:nn
2792     }{
2793         \__stex_variables_do_simple:nnn
2794     }
2795 }
2796
2797 \NewDocumentCommand \svar { 0{} m }{
2798     \tl_if_empty:nTF {#1}{
2799         \str_set:Nn \l_tmpa_str { #2 }
2800     }{
2801         \str_set:Nn \l_tmpa_str { #1 }
2802     }
2803     \_stex_term_omv:nn {
2804         var://\l_tmpa_str
2805     }{
2806         \exp_args:Nnx \use:nn {
2807             \def\comp{\_varcomp}
2808             \str_set:Nx \l_stex_current_symbol_str { var://\l_tmpa_str }
2809             \comp{ #2 }
2810         }{

```

```

2811 \stex_reset:N \comp
2812 \stex_reset:N \l_stex_current_symbol_str
2813 }
2814 }
2815 }
2816
2817
2818
2819 \keys_define:nn { stex / varseq } {
2820   name      .str_set_x:N = \l__stex_variables_name_str ,
2821   args      .int_set:N   = \l__stex_variables_args_int ,
2822   type      .tl_set:N    = \l__stex_variables_type_tl ,
2823   mid       .tl_set:N    = \l__stex_variables_mid_tl ,
2824   bind      .choices:nn =
2825     {forall,exists}
2826     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2827 }
2828
2829 \cs_new_protected:Nn \__stex_variables_seq_args:n {
2830   \str_clear:N \l__stex_variables_name_str
2831   \int_set:Nn \l__stex_variables_args_int 1
2832   \tl_clear:N \l__stex_variables_type_tl
2833   \str_clear:N \l__stex_variables_bind_str
2834
2835   \keys_set:nn { stex / varseq } { #1 }
2836 }
2837
2838 \NewDocumentCommand \varseq {m O{} m m m}{
2839   \__stex_variables_seq_args:n { #2 }
2840   \str_if_empty:NT \l__stex_variables_name_str {
2841     \str_set:Nx \l__stex_variables_name_str { #1 }
2842   }
2843   \prop_clear:N \l_tmpa_prop
2844   \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
2845
2846   \seq_set_from_clist:Nn \l_tmpa_seq {#3}
2847   \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
2848     \msg_error:nnxx{stex}{error/seqlength}
2849     {\int_use:N \l__stex_variables_args_int}
2850     {\seq_count:N \l_tmpa_seq}
2851   }
2852   \seq_set_from_clist:Nn \l_tmpb_seq {#4}
2853   \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
2854     \msg_error:nnxx{stex}{error/seqlength}
2855     {\int_use:N \l__stex_variables_args_int}
2856     {\seq_count:N \l_tmpb_seq}
2857   }
2858   \prop_put:Nnn \l_tmpa_prop {starts} {#3}
2859   \prop_put:Nnn \l_tmpa_prop {ends} {#4}
2860
2861   \cs_generate_from_arg_count:cNnn {stex_varseq\l__stex_variables_name_str _cs}
2862   \cs_set:Npn {\int_use:N \l__stex_variables_args_int} { #5 }
2863
2864   \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}

```

```

2865 \int_step_inline:nn \l__stex_variables_args_int {
2866   \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
2867 }
2868 \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
2869 \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
2870 \tl_if_empty:NF \l__stex_variables_mid_tl {
2871   \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
2872   \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
2873 }
2874 \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}
2875 \int_step_inline:nn \l__stex_variables_args_int {
2876   \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
2877 }
2878 \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
2879 \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
2880
2881
2882 \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl
2883
2884 \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}
2885
2886 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}
2887
2888 \int_step_inline:nn \l__stex_variables_args_int {
2889   \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
2890     \stex_term_math_arg:nnn{##1}{0}{\exp_not:n{####}##1}
2891   }}
2892 }
2893
2894 \tl_set:Nx \l_tmpa_tl {
2895   \stex_term_math_oma:nnnn { varseq://\l__stex_variables_name_str}{0}{
2896     \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
2897   }
2898 }
2899
2900 \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \stex_symbol_after_invokation_tl} }
2901
2902 \exp_args:Nno \use:nn {
2903   \cs_generate_from_arg_count:cNnn {stex_varseq\l__stex_variables_name_str _cs}
2904   \cs_set:Npn {\int_use:N \l__stex_variables_args_int}{\l_tmpa_tl}
2905
2906   \stex_debug:nn{sequences}{New~Sequence:~
2907     \expandafter\meaning\csname stex_varseq\l__stex_variables_name_str _cs\endcsname\\~\\
2908     \prop_to_keyval:N \l_tmpa_prop
2909   }
2910
2911   \prop_set_eq:cN {stex_varseq\l__stex_variables_name_str _prop}\l_tmpa_prop
2912   \ignorespacesandpars
2913 }
2914
2915 </package>

```



## Chapter 29

# STEX -Terms Implementation

```
2916 <*package>
2917
2918 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2919
2920 <@@=stex_terms>
2921
2922 Warnings and error messages
2923 \msg_new:nnn{stex}{error/nonotation}{
2924   Symbol~#1~invoked,~but~has~no~notation#2!
2925 }
2926 \msg_new:nnn{stex}{error/notationarg}{
2927   Error~in~parsing~notation~#1
2928 }
2929 \msg_new:nnn{stex}{error/noop}{
2930   Symbol~#1~has~no~operator~notation~for~notation~#2
2931 }
2932 \msg_new:nnn{stex}{error/notallowed}{
2933   Symbol~invocation~#1~not~allowed~in~notation~component~of~#2
2934 }
2935
```

### 29.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
2934
2935
2936 \bool_new:N \l_stex_allow_semantic_bool
2937 \bool_set_true:N \l_stex_allow_semantic_bool
2938
2939 \cs_new_protected:Nn \stex_invoke_symbol:n {
2940   \bool_if:NTF \l_stex_allow_semantic_bool {
2941     \str_if_eq:eeF {
2942       \prop_item:cn {
2943         l_stex_symdecl_#1_prop
2944       }{ deprecate }
2945     }
2946   }
2947 }
```

```

2945   }{}{
2946     \msg_warning:nxxx{stex}{warning/deprecated}{
2947       Symbol~#1
2948     }{
2949       \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
2950     }
2951   }
2952   \if_mode_math:
2953     \exp_after:wN \__stex_terms_invoke_math:n
2954   \else:
2955     \exp_after:wN \__stex_terms_invoke_text:n
2956   \fi: { #1 }
2957 }{
2958   \msg_error:nxxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
2959 }
2960 }
2961
2962 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
2963   \peek_charcode_remove:NTF ! {
2964     \__stex_terms_invoke_op_custom:nn {#1}
2965   }{
2966     \__stex_terms_invoke_custom:nn {#1}
2967   }
2968 }
2969
2970 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2971   \peek_charcode_remove:NTF ! {
2972     % operator
2973     \peek_charcode_remove:NTF * {
2974       % custom op
2975       \__stex_terms_invoke_op_custom:nn {#1}
2976     }{
2977       % op notation
2978       \peek_charcode:NTF [ {
2979         \__stex_terms_invoke_op_notation:nw {#1}
2980       }{
2981         \__stex_terms_invoke_op_notation:nw {#1}[]
2982       }
2983     }
2984   }{
2985     \peek_charcode_remove:NTF * {
2986       \__stex_terms_invoke_custom:nn {#1}
2987       % custom
2988     }{
2989       % normal
2990       \peek_charcode:NTF [ {
2991         \__stex_terms_invoke_notation:nw {#1}
2992       }{
2993         \__stex_terms_invoke_notation:nw {#1}[]
2994       }
2995     }
2996   }
2997 }
2998

```

```

2999
3000 \cs_new_protected:Nn \__stex_terms_invoke_op_custom:nn {
3001   \exp_args:Nnx \use:nn {
3002     \def\comp{\_comp}
3003     \str_set:Nn \l_stex_current_symbol_str { #1 }
3004     \bool_set_false:N \l_stex_allow_semantic_bool
3005     \stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
3006       \comp{ #2 }
3007     }
3008   }{
3009     \stex_reset:N \comp
3010     \stex_reset:N \l_stex_current_symbol_str
3011     \bool_set_true:N \l_stex_allow_semantic_bool
3012   }
3013 }
3014
3015 \keys_define:nn { stex / terms } {
3016   lang .tl_set_x:N = \l_stex_notation_lang_str ,
3017   variant .tl_set_x:N = \l_stex_notation_variant_str ,
3018   unknown .code:n = \str_set:Nx
3019     \l_stex_notation_variant_str \l_keys_key_str
3020 }
3021
3022 \cs_new_protected:Nn \__stex_terms_args:n {
3023   \str_clear:N \l_stex_notation_lang_str
3024   \str_clear:N \l_stex_notation_variant_str
3025
3026   \keys_set:nn { stex / terms } { #1 }
3027 }
3028
3029 \cs_new_protected:Nn \stex_find_notation:nn {
3030   \__stex_terms_args:n { #2 }
3031   \seq_if_empty:cTF {
3032     l_stex_symdecl_ #1 _notations
3033   } {
3034     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
3035   } {
3036     \bool_lazy_all:nTF {
3037       {\str_if_empty_p:N \l_stex_notation_variant_str}
3038       {\str_if_empty_p:N \l_stex_notation_lang_str}
3039     }{
3040       \seq_get_left:cN {l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
3041     }{
3042       \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
3043         \l_stex_notation_variant_str \c_hash_str \l_stex_notation_lang_str
3044       }{
3045         \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3046       }{
3047         \msg_error:nnxx{stex}{error/nonotation}{#1}{
3048           ~\l_stex_notation_variant_str \c_hash_str \l_stex_notation_lang_str
3049         }
3050       }
3051     }
3052   }

```

```

3053 }
3054
3055 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3056   \exp_args:Nnx \use:nn {
3057     \def\comp{\_comp}
3058     \str_set:Nn \l_stex_current_symbol_str { #1 }
3059     \stex_find_notation:nn { #1 }{ #2 }
3060     \bool_set_false:N \l_stex_allow_semantic_bool
3061     \cs_if_exist:cTF {
3062       stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3063     }{
3064       \_stex_term_oms:nnn {
3065         #1 \c_hash_str \l_stex_notation_variant_str
3066       }{ #1 }{
3067         \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3068       }
3069     }{
3070       \int_compare:nNnTF {\prop_item:cn {\l_stex_symdecl_#1_prop}{arity}} = 0{
3071         \cs_if_exist:cTF {
3072           stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3073         }{
3074           \tl_set:Nx \stex_symbol_after_invokation_tl {
3075             \_stex_reset:N \comp
3076             \_stex_reset:N \stex_symbol_after_invokation_tl
3077             \_stex_reset:N \l_stex_current_symbol_str
3078             \bool_set_true:N \l_stex_allow_semantic_bool
3079           }
3080           \def\comp{\_comp}
3081           \str_set:Nn \l_stex_current_symbol_str { #1 }
3082           \bool_set_false:N \l_stex_allow_semantic_bool
3083           \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3084         }{
3085           \msg_error:nnxx{stex}{error/nonotation}{#1}{
3086             ~\l_stex_notation_variant_str
3087           }
3088         }
3089       }{
3090         \msg_error:nnxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3091       }
3092     }
3093   }{
3094     \_stex_reset:N \comp
3095     \_stex_reset:N \l_stex_current_symbol_str
3096     \bool_set_true:N \l_stex_allow_semantic_bool
3097   }
3098 }
3099
3100 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {
3101   \stex_find_notation:nn { #1 }{ #2 }
3102   \cs_if_exist:cTF {
3103     stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3104   }{
3105     \tl_set:Nx \stex_symbol_after_invokation_tl {
3106       \_stex_reset:N \comp

```

```

3107     \stex_reset:N \stex_symbol_after_invokation_tl
3108     \stex_reset:N \l_stex_current_symbol_str
3109     \bool_set_true:N \l_stex_allow_semantic_bool
3110   }
3111   \def\comp{\_comp}
3112   \str_set:Nn \l_stex_current_symbol_str { #1 }
3113   \bool_set_false:N \l_stex_allow_semantic_bool
3114   \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3115 }{
3116   \msg_error:nnxx{stex}{error/nonotation}{#1}{
3117     ~\l_stex_notation_variant_str
3118   }
3119 }
3120 }
3121
3122 \prop_new:N \l__stex_terms_custom_args_prop
3123
3124 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3125   \exp_args:Nnx \use:nn {
3126     \bool_set_false:N \l_stex_allow_semantic_bool
3127     \def\comp{\_comp}
3128     \str_set:Nn \l_stex_current_symbol_str { #1 }
3129     \prop_clear:N \l__stex_terms_custom_args_prop
3130     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3131     \prop_get:cnN {
3132       l_stex_symdecl_#1 _prop
3133     }{ args } \l_tmpa_str
3134     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3135     \tl_set:Nn \arg { \__stex_terms_arg: }
3136     \str_if_empty:NTF \l_tmpa_str {
3137       \stex_term_oms:nnn {#1}{#1}{#2}
3138     }{
3139       \str_if_in:NnTF \l_tmpa_str b {
3140         \stex_term_ombind:nnn {#1}{#1}{#2}
3141       }{
3142         \str_if_in:NnTF \l_tmpa_str B {
3143           \stex_term_ombind:nnn {#1}{#1}{#2}
3144         }{
3145           \stex_term_oma:nnn {#1}{#1}{#2}
3146         }
3147       }
3148     }
3149     % TODO check that all arguments exist
3150   }{
3151     \stex_reset:N \l_stex_current_symbol_str
3152     \stex_reset:N \arg
3153     \stex_reset:N \comp
3154     \stex_reset:N \l__stex_terms_custom_args_prop
3155     \bool_set_true:N \l_stex_allow_semantic_bool
3156   }
3157 }
3158
3159 \NewDocumentCommand \__stex_terms_arg: { s O{} m}{
3160   \tl_if_empty:nTF {#2}{

```

```

3161 \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
3162 \bool_set_true:N \l_tmpa_bool
3163 \bool_do_while:Nn \l_tmpa_bool {
3164   \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int}
3165   \int_incr:N \l_tmpa_int
3166   }{
3167   \bool_set_false:N \l_tmpa_bool
3168   }
3169 }
3170 }{
3171   \int_set:Nn \l_tmpa_int { #2 }
3172   \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
3173   % TODO throw error
3174   }
3175 }
3176 \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
3177 \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3178   % TODO throw error
3179 }
3180 \bool_set_true:N \l_stex_allow_semantic_bool
3181 \IfBooleanTF#1{
3182   \stex_annotate_invisible:n {
3183     \exp_args:No \_stex_term_arg:nn {\l_stex_current_symbol_str}{#3}
3184   }
3185 }{
3186   \exp_args:No \_stex_term_arg:nn {\l_stex_current_symbol_str}{#3}
3187 }
3188 \bool_set_false:N \l_stex_allow_semantic_bool
3189 }
3190
3191
3192 \cs_new_protected:Nn \_stex_term_arg:nn {
3193   \bool_set_true:N \l_stex_allow_semantic_bool
3194   \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3195   \bool_set_false:N \l_stex_allow_semantic_bool
3196 }
3197
3198 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
3199   \exp_args:Nnx \use:nn
3200   { \int_set:Nn \l__stex_terms_downprec { #2 }
3201     \_stex_term_arg:nn { #1 }{ #3 }
3202   }
3203   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3204   }

```

(End definition for `\stex_invoke_symbol:n`. This function is documented on page 46.)

`\_stex_term_math_assoc_arg:nnnn`

```

3205 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
3206   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3207   \tl_set:Nn \l_tmpb_tl {\_stex_term_math_arg:nnn{#1}{#2}}
3208   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 }}{
3209     \expandafter\if\expandafter\relax\noexpand#3
3210     \expandafter\__stex_terms_math_assoc_arg_maybe_sequence:N\expandafter#3

```

```

3211 \else\expandafter\__stex_terms_math_assoc_arg_simple:n\expandafter#3\fi
3212 }{
3213 \__stex_terms_math_assoc_arg_simple:n{#3}
3214 }
3215 }
3216
3217 \cs_new_protected:Nn \__stex_terms_math_assoc_arg_maybe_sequence:N {
3218 \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
3219 \str_if_empty:NTF \l_tmpa_str {
3220 \exp_args:Nx \cs_if_eq:NNTF {
3221 \tl_head:N #1
3222 } \stex_invoke_sequence:n {
3223 \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
3224 \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
3225 \tl_set:Nx \l_tmpa_tl {\prop_item:cn {stex_varseq \l_tmpa_str _prop}{notation}}
3226 \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
3227 \tl_set:Nx \l_tmpa_tl {\exp_not:N \exp_not:n{
3228 \exp_not:n{\exp_args:Nnx \use:nn} {
3229 \exp_not:n {
3230 \def\comp{\_varcomp}
3231 \str_set:Nn \l_stex_current_symbol_str
3232 } {varseq://\l_tmpa_str}
3233 \exp_not:n{ ##1 }
3234 }{
3235 \exp_not:n {
3236 \_stex_reset:N \comp
3237 \_stex_reset:N \l_stex_current_symbol_str
3238 }
3239 }
3240 }}}
3241 \exp_args:Nno \use:nn {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
3242 \seq_reverse:N \l_tmpa_seq
3243 \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
3244 \seq_map_inline:Nn \l_tmpa_seq {
3245 \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3246 \exp_args:Nno
3247 \l_tmpa_cs { ##1 } \l_tmpa_tl
3248 }
3249 }
3250 \tl_set:Nx \l_tmpa_tl {
3251 \_stex_term_omv:nn {varseq://\l_tmpa_str}{
3252 \exp_args:No \exp_not:n \l_tmpa_tl
3253 }
3254 }
3255 \exp_args:No\l_tmpb_tl\l_tmpa_tl
3256 }{
3257 \__stex_terms_math_assoc_arg_simple:n { #1 }
3258 }
3259 } {
3260 \__stex_terms_math_assoc_arg_simple:n { #1 }
3261 }
3262
3263 }
3264

```

```

3265 \cs_new_protected:Nn \__stex_terms_math_assoc_arg_simple:n {
3266   \clist_set:Nn \l_tmpa_clist{ #1 }
3267   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3268     \tl_set:Nn \l_tmpa_tl { #1 }
3269   }{
3270     \clist_reverse:N \l_tmpa_clist
3271     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3272
3273     \clist_map_inline:Nn \l_tmpa_clist {
3274       \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3275         \exp_args:Nno
3276         \l_tmpa_cs { ##1 } \l_tmpa_tl
3277       }
3278     }
3279   }
3280   \exp_args:No\l_tmpb_tl\l_tmpa_tl
3281 }

```

(End definition for `\stex_term_math_assoc_arg:nnnn`. This function is documented on page 46.)

## 29.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
3282 \tl_const:Nx \infprec {\int_use:N \c_max_int}
3283 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
3284 \int_new:N \l__stex_terms_downprec
3285 \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 47.)

Bracketing:

```

\l__stex_terms_left_bracket_str
\l__stex_terms_right_bracket_str
3286 \tl_set:Nn \l__stex_terms_left_bracket_str (
3287 \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

```

3288 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
3289   \bool_if:NTF \l__stex_terms_brackets_done_bool {
3290     \bool_set_false:N \l__stex_terms_brackets_done_bool
3291     #2
3292   } {
3293     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
3294       \bool_if:NTF \l_stex_inarray_bool { #2 }{
3295         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
3296         \dobrackets { #2 }
3297       }
3298     }{ #2 }
3299   }
3300 }

```



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

### `\dobrackets`

```

3301 \bool_new:N \l__stex_terms_brackets_done_bool
3302 %\RequirePackage{scalerel}
3303 \cs_new_protected:Npn \dobrackets #1 {
3304   %\ThisStyle{\if D\m@switch
3305   %   \exp_args:Nnx \use:nn
3306   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
3307   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
3308   % \else
3309   \exp_args:Nnx \use:nn
3310   {
3311     \bool_set_true:N \l__stex_terms_brackets_done_bool
3312     \int_set:Nn \l__stex_terms_downprec \infpref
3313     \l__stex_terms_left_bracket_str
3314     #1
3315   }
3316   {
3317     \bool_set_false:N \l__stex_terms_brackets_done_bool
3318     \l__stex_terms_right_bracket_str
3319     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3320   }
3321   %\fi}
3322 }
```

(End definition for `\dobrackets`. This function is documented on page 47.)

### `\withbrackets`

```

3323 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
3324   \exp_args:Nnx \use:nn
3325   {
3326     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
3327     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
3328     #3
3329   }
3330   {
3331     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
3332     {\l__stex_terms_left_bracket_str}
3333     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
3334     {\l__stex_terms_right_bracket_str}
3335   }
3336 }
```

(End definition for `\withbrackets`. This function is documented on page 47.)

### `\STEXinvisible`

```

3337 \cs_new_protected:Npn \STEXinvisible #1 {
3338   \stex_annotate_invisible:n { #1 }
3339 }
```

(End definition for `\STEXinvisible`. This function is documented on page 47.)

OMDoc terms:

`\_stex_term_math_oms:nnnn`

```
3340 \cs_new_protected:Nn \_stex_term_oms:nnn {
3341   \stex_annotate:nnn{ OMID }{ #2 }{
3342     \stex_highlight_term:nn { #1 } { #3 }
3343   }
3344 }
3345
3346 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
3347   \__stex_terms_maybe_brackets:nn { #3 }{
3348     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3349   }
3350 }
```

*(End definition for \\_stex\_term\_math\_oms:nnnn. This function is documented on page 46.)*

`\_stex_term_math_omv:nn`

```
3351 \cs_new_protected:Nn \_stex_term_omv:nn {
3352   \stex_annotate:nnn{ OMV }{ #1 }{
3353     \stex_highlight_term:nn { #1 } { #2 }
3354   }
3355 }
```

*(End definition for \\_stex\_term\_math\_omv:nn. This function is documented on page ??.)*

`\_stex_term_math_oma:nnnn`

```
3356 \cs_new_protected:Nn \_stex_term_oma:nnn {
3357   \stex_annotate:nnn{ OMA }{ #2 }{
3358     \stex_highlight_term:nn { #1 } { #3 }
3359   }
3360 }
3361
3362 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
3363   \__stex_terms_maybe_brackets:nn { #3 }{
3364     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3365   }
3366 }
```

*(End definition for \\_stex\_term\_math\_oma:nnnn. This function is documented on page 46.)*

`\_stex_term_math_omb:nnnn`

```
3367 \cs_new_protected:Nn \_stex_term_ombind:nnn {
3368   \stex_annotate:nnn{ OMBIND }{ #2 }{
3369     \stex_highlight_term:nn { #1 } { #3 }
3370   }
3371 }
3372
3373 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
3374   \__stex_terms_maybe_brackets:nn { #3 }{
3375     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3376   }
3377 }
```

*(End definition for \\_stex\_term\_math\_omb:nnnn. This function is documented on page 46.)*

```

\symref
\symname
3378 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3379
3380 \keys_define:nn { stex / symname } {
3381   pre      .tl_set_x:N = \l__stex_terms_pre_tl ,
3382   post     .tl_set_x:N = \l__stex_terms_post_tl ,
3383   root     .tl_set_x:N = \l__stex_terms_root_tl
3384 }
3385
3386 \cs_new_protected:Nn \stex_symname_args:n {
3387   \tl_clear:N \l__stex_terms_post_tl
3388   \tl_clear:N \l__stex_terms_pre_tl
3389   \tl_clear:N \l__stex_terms_root_str
3390   \keys_set:nn { stex / symname } { #1 }
3391 }
3392
3393 \NewDocumentCommand \symref { m m }{
3394   \let\compemph_uri_prev:\compemph@uri
3395   \let\compemph@uri\symrefemph@uri
3396   \STEXsymbol{#1}!\{ #2 }
3397   \let\compemph@uri\compemph_uri_prev:
3398 }
3399
3400 \NewDocumentCommand \synonym { 0{} m m }{
3401   \stex_symname_args:n { #1 }
3402   \let\compemph_uri_prev:\compemph@uri
3403   \let\compemph@uri\symrefemph@uri
3404   % TODO
3405   \STEXsymbol{#2}!\{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
3406   \let\compemph@uri\compemph_uri_prev:
3407 }
3408
3409 \NewDocumentCommand \symname { 0{} m }{
3410   \stex_symname_args:n { #1 }
3411   \stex_get_symbol:n { #2 }
3412   \str_set:Nx \l_tmpa_str {
3413     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3414   }
3415   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3416
3417   \let\compemph_uri_prev:\compemph@uri
3418   \let\compemph@uri\symrefemph@uri
3419   \exp_args:NNx \use:nn
3420   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\{
3421     \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
3422   } }
3423   \let\compemph@uri\compemph_uri_prev:
3424 }
3425
3426 \NewDocumentCommand \Symname { 0{} m }{
3427   \stex_symname_args:n { #1 }
3428   \stex_get_symbol:n { #2 }
3429   \str_set:Nx \l_tmpa_str {
3430     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }

```

```

3431 }
3432 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {-}
3433 \let\compemph_uri_prev:\compemph@uri
3434 \let\compemph@uri\symrefemph@uri
3435 \exp_args:NNx \use:nn
3436 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!{
3437   \exp_after:wN \stex_capitalize:n \l_tmpa_str
3438   \l__stex_terms_post_tl
3439 } }
3440 \let\compemph@uri\compemph_uri_prev:
3441 }

```

(End definition for `\symref` and `\symname`. These functions are documented on page 46.)

## 29.3 Notation Components

```

3442 <@@=stex_notationcomps>

```

`\stex_highlight_term:nn`

```

3443 \cs_new_protected:Nn \stex_highlight_term:nn {
3444   #2
3445 }
3446
3447 \cs_new_protected:Nn \stex_unhighlight_term:n {
3448   % \latexml_if:TF {
3449   %   #1
3450   % } {
3451   %   \rustex_if:TF {
3452   %     #1
3453   %   } {
3454   %     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
3455   %   }
3456   % }
3457 }

```

(End definition for `\stex_highlight_term:nn`. This function is documented on page 47.)

```

\comp
\compemph@uri
\compemph
\defemph
\defemph@uri
\symrefemph
\symrefemph@uri
\varemp
\varemp@uri
3458 \cs_new_protected:Npn \_comp #1 {
3459   \str_if_empty:NF \l_stex_current_symbol_str {
3460     \rustex_if:TF {
3461       \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
3462     }{
3463       \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3464     }
3465   }
3466 }
3467
3468 \cs_new_protected:Npn \_varcomp #1 {
3469   \str_if_empty:NF \l_stex_current_symbol_str {
3470     \rustex_if:TF {
3471       \stex_annotate:nnn { varcomp }{ \l_stex_current_symbol_str }{ #1 }
3472     }{
3473       \exp_args:Nnx \varemp@uri { #1 } { \l_stex_current_symbol_str }

```

```

3474     }
3475   }
3476 }
3477
3478 \def\comp{\_comp}
3479
3480 \cs_new_protected:Npn \compemph@uri #1 #2 {
3481   \compemph{ #1 }
3482 }
3483
3484
3485 \cs_new_protected:Npn \compemph #1 {
3486   #1
3487 }
3488
3489 \cs_new_protected:Npn \defemph@uri #1 #2 {
3490   \defemph{#1}
3491 }
3492
3493 \cs_new_protected:Npn \defemph #1 {
3494   \textbf{#1}
3495 }
3496
3497 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3498   \symrefemph{#1}
3499 }
3500
3501 \cs_new_protected:Npn \symrefemph #1 {
3502   \textbf{#1}
3503 }
3504
3505 \cs_new_protected:Npn \varemp@uri #1 #2 {
3506   \varemp{#1}
3507 }
3508
3509 \cs_new_protected:Npn \varemp #1 {
3510   #1
3511 }

```

(End definition for `\comp` and others. These functions are documented on page 47.)

## **\ellipses**

```

3512 \NewDocumentCommand \ellipses {} { \ldots }

```

(End definition for `\ellipses`. This function is documented on page 47.)

```

\parray
\prmatrix 3513 \bool_new:N \l_stex_inarray_bool
\parrayline 3514 \bool_set_false:N \l_stex_inarray_bool
\parraylineh 3515 \NewDocumentCommand \parray { m m } {
\parraycell 3516   \begingroup
3517   \bool_set_true:N \l_stex_inarray_bool
3518   \begin{array}{#1}
3519     #2
3520   \end{array}

```

```

3521 \endgroup
3522 }
3523
3524 \NewDocumentCommand \prmatrix { m } {
3525   \begingroup
3526   \bool_set_true:N \l_stex_inarray_bool
3527   \begin{matrix}
3528     #1
3529   \end{matrix}
3530 \endgroup
3531 }
3532
3533 \def \maybepline {
3534   \bool_if:NT \l_stex_inarray_bool {\hline}
3535 }
3536
3537 \def \parrayline #1 #2 {
3538   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3539 }
3540
3541 \def \pmrow #1 { \parrayline{}{ #1 } }
3542
3543 \def \parraylineh #1 #2 {
3544   #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3545 }
3546
3547 \def \parraycell #1 {
3548   #1 \bool_if:NT \l_stex_inarray_bool {&}
3549 }

```

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

## 29.4 Variables

```

3550 <@@=stex_variables>

```

\stex\_invoke\_variable:n Invokes a variable

```

3551 \cs_new_protected:Nn \stex_invoke_variable:n {
3552   \if_mode_math:
3553     \exp_after:wN \__stex_variables_invoke_math:n
3554   \else:
3555     \exp_after:wN \__stex_variables_invoke_text:n
3556   \fi: {#1}
3557 }
3558
3559 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
3560   %TODO
3561 }
3562
3563
3564 \cs_new_protected:Nn \__stex_variables_invoke_math:n {
3565   \peek_charcode_remove:NTF ! {
3566     \peek_charcode_remove:NTF ! {
3567       \peek_charcode:NTF [ {

```

```

3568     \__stex_variables_invoke_op_custom:nw
3569   }{
3570     % TODO throw error
3571   }
3572 }{
3573   \__stex_variables_invoke_op:n { #1 }
3574 }
3575 }{
3576   \peek_charcode_remove:NTF * {
3577     \__stex_variables_invoke_text:n { #1 }
3578   }{
3579     \__stex_variables_invoke_math_ii:n { #1 }
3580   }
3581 }
3582 }
3583
3584 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
3585   \cs_if_exist:cTF {
3586     stex_var_op_notation_ #1 _cs
3587   }{
3588     \exp_args:Nnx \use:nn {
3589       \def\comp{\_varcomp}
3590       \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3591       \_stex_term_omv:nn { var://#1 }{
3592         \use:c{stex_var_op_notation_ #1 _cs }
3593       }
3594     }{
3595       \_stex_reset:N \comp
3596       \_stex_reset:N \l_stex_current_symbol_str
3597     }
3598   }{
3599     \int_compare:nNnTF {\prop_item:cn {\l_stex_variable_#1_prop}{arity}} = 0{
3600       \__stex_variables_invoke_math_ii:n {#1}
3601     }{
3602       \msg_error:nxxx{stex}{error/noop}{variable~#1}{-}
3603     }
3604   }
3605 }
3606
3607 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n #1 {
3608   \cs_if_exist:cTF {
3609     stex_var_notation_#1_cs
3610   }{
3611     \tl_set:Nx \stex_symbol_after_invokation_tl {
3612       \_stex_reset:N \comp
3613       \_stex_reset:N \stex_symbol_after_invokation_tl
3614       \_stex_reset:N \l_stex_current_symbol_str
3615       \bool_set_true:N \l_stex_allow_semantic_bool
3616     }
3617     \def\comp{\_varcomp}
3618     \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3619     \bool_set_false:N \l_stex_allow_semantic_bool
3620     \use:c{stex_var_notation_#1_cs}
3621   }{

```

```

3622 \msg_error:nxx{stex}{error/nonotation}{variable~#1}{s}
3623 }
3624 }

```

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

## 29.5 Sequences

```

3625 <@@=stex_sequences>
3626
3627 \cs_new_protected:Nn \stex_invoke_sequence:n {
3628   \peek_charcode_remove:NTF ! {
3629     \stex_term_omv:nn {varseq://#1}{
3630       \exp_args:Nnx \use:nn {
3631         \def\comp{\_varcomp}
3632         \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3633         \prop_item:cn{stex_varseq_#1_prop}{notation}
3634       }{
3635         \stex_reset:N \comp
3636         \stex_reset:N \l_stex_current_symbol_str
3637       }
3638     }
3639   }{
3640     \bool_set_false:N \l_stex_allow_semantic_bool
3641     \def\comp{\_varcomp}
3642     \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3643     \tl_set:Nx \stex_symbol_after_invokation_tl {
3644       \stex_reset:N \comp
3645       \stex_reset:N \stex_symbol_after_invokation_tl
3646       \stex_reset:N \l_stex_current_symbol_str
3647       \bool_set_true:N \l_stex_allow_semantic_bool
3648     }
3649     \use:c { stex_varseq_#1_cs }
3650   }
3651 }
3652 </package>

```



## Chapter 30

# STEX -Structural Features Implementation

```
3653 <*package>
3654
3655 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3656
3657
3658 Warnings and error messages
3659 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3660   Symbol~#1~can~not~be~assigned~in~copymodule~#2
3661 }
3662 \msg_new:nnn{stex}{error/interpretmodule/noddefinens}{
3663   Symbol~#1~not~assigned~in~interpretmodule~#2
3664 }
3665 \msg_new:nnn{stex}{error/unknownstructure}{
3666   No~structure~#1~found!
3667 }
3668 \msg_new:nnn{stex}{error/unknownfield}{
3669   No~field~#1~in~instance~#2~found!
3670 }
3671 \msg_new:nnn{stex}{error/keyval}{
3672   Invalid~key=value~pair~#1
3673 }
3674 \msg_new:nnn{stex}{error/instantiate/missing}{
3675   Assignments~missing~in~instantiate:~#1
3676 }
3677 \msg_new:nnn{stex}{error/incompatible}{
3678   Incompatible~signature:~#1~(#2)~and~#3~(#4)
3679 }
3680
3681
```

## 30.1 Imports with modification

```

3682 <@@=stex_copymodule>
3683 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
3684   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3685     \tl_set:Nn \l_tmpa_tl { #1 }
3686     \__stex_copymodule_get_symbol_from_cs:
3687   }{
3688     % argument is a string
3689     % is it a command name?
3690     \cs_if_exist:cTF { #1 }{
3691       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3692       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3693       \str_if_empty:NNTF \l_tmpa_str {
3694         \exp_args:Nx \cs_if_eq:NNTF {
3695           \tl_head:N \l_tmpa_tl
3696         } \stex_invoke_symbol:n {
3697           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
3698         }{
3699           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3700         }
3701       } {
3702         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3703       }
3704     }{
3705       % argument is not a command name
3706       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3707       % \l_stex_all_symbols_seq
3708     }
3709   }
3710 }
3711
3712 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
3713   \str_set:Nn \l_tmpa_str { #1 }
3714   \bool_set_false:N \l_tmpa_bool
3715   \bool_if:NF \l_tmpa_bool {
3716     \tl_set:Nn \l_tmpa_tl {
3717       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
3718     }
3719     \str_set:Nn \l_tmpa_str { #1 }
3720     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3721     \seq_map_inline:Nn #2 {
3722       \str_set:Nn \l_tmpb_str { ##1 }
3723       \str_if_eq:eeT { \l_tmpa_str } {
3724         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3725       } {
3726         \seq_map_break:n {
3727           \tl_set:Nn \l_tmpa_tl {
3728             \str_set:Nn \l_stex_get_symbol_uri_str {
3729               ##1
3730             }
3731           }
3732         }
3733       }

```

```

3734     }
3735     \l_tmpa_tl
3736   }
3737 }
3738
3739 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
3740   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3741     { \tl_tail:N \l_tmpa_tl }
3742   \tl_if_single:NTF \l_tmpa_tl {
3743     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3744       \exp_after:wN \str_set:Nn \exp_after:wN
3745         \l_stex_get_symbol_uri_str \l_tmpa_tl
3746       \__stex_copymodule_get_symbol_check:n { #1 }
3747     }{
3748       % TODO
3749       % tail is not a single group
3750     }
3751   }{
3752     % TODO
3753     % tail is not a single group
3754   }
3755 }
3756
3757 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
3758   \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
3759     \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3760       :~\seq_use:Nn #1 {,~}
3761     }
3762   }
3763 }
3764
3765 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
3766   \stex_import_module_uri:nn { #1 } { #2 }
3767   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3768   \stex_import_require_module:nnnn
3769     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3770     { \l_stex_import_path_str } { \l_stex_import_name_str }
3771   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3772   \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
3773   \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
3774   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
3775     \seq_map_inline:cn {c_stex_module_###_constants}{
3776       \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
3777         ##1 ? ####1
3778       }
3779     }
3780   }
3781   \seq_clear:N \l_tmpa_seq
3782   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3783     name      = \l_stex_current_copymodule_name_str ,
3784     module    = \l_stex_current_module_str ,
3785     from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3786     includes  = \l_tmpa_seq ,
3787     fields    = \l_tmpa_seq

```

```

3788 }
3789 \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3790   as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3791   \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_seq {,
3792 \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
3793 \stex_if_smsmode:F {
3794   \begin{stex_annotate_env} {#4} {
3795     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3796   }
3797   \stex_annotate_invisible:nnn{from}{\l_stex_import_ns_str ?\l_stex_import_name_str}{}
3798 }
3799 \bool_set_eq:NN \l__stex_copymodule_oldhtml_bool \_stex_html_do_output_bool
3800 \bool_set_false:N \_stex_html_do_output_bool
3801 }
3802 \cs_new_protected:Nn \stex_copymodule_end:n {
3803   \def \l_tmpa_cs ##1 ##2 {#1}
3804   \bool_set_eq:NN \_stex_html_do_output_bool \l__stex_copymodule_oldhtml_bool
3805   \tl_clear:N \l_tmpa_tl
3806   \tl_clear:N \l_tmpb_tl
3807   \prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3808   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
3809     \seq_map_inline:cn {c_stex_module_##1_constants}{
3810       \tl_clear:N \l_tmpc_tl
3811       \l_tmpa_cs{##1}{####1}
3812       \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
3813         \tl_put_right:Nx \l_tmpa_tl {
3814           \prop_set_from_keyval:cn {
3815             l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_
3816           }{
3817             \exp_after:wN \prop_to_keyval:N \csname
3818               l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_
3819             \endcsname
3820           }
3821           \seq_clear:c {
3822             l_stex_symdecl_
3823             \l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_name
3824             _notations
3825           }
3826         }
3827         \tl_put_right:Nx \l_tmpc_tl {
3828           \stex_copy_notations:nn {\l_stex_current_module_str ? \use:c{l__stex_copymodule_co
3829           \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?####1
3830         }
3831         \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{l__stex_copymodul
3832         \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
3833           \tl_put_right:Nx \l_tmpc_tl {
3834             \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1
3835           }
3836           \tl_put_right:Nx \l_tmpa_tl {
3837             \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
3838               \stex_invoke_symbol:n {
3839                 \l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_
3840             }
3841           }

```

```

3842     }
3843   }
3844 }{
3845   \tl_put_right:Nx \l_tmpc_tl {
3846     \stex_copy_notations:nn {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str}
3847   }
3848   \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
3849   \prop_put:Nnx \l_tmpa_prop { name }{ \l_stex_current_copymodule_name_str / ####1 }
3850   \prop_put:Nnx \l_tmpa_prop { module }{ \l_stex_current_module_str }
3851   \tl_put_right:Nx \l_tmpa_tl {
3852     \prop_set_from_keyval:cn {
3853       l_stex_symdecl_ \l_stex_current_module_str ? \l_stex_current_copymodule_name_str
3854     }{
3855       \prop_to_keyval:N \l_tmpa_prop
3856     }
3857     \seq_clear:c {
3858       l_stex_symdecl_
3859       \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3860       _notations
3861     }
3862   }
3863   \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str}
3864   \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
3865     \tl_put_right:Nx \l_tmpc_tl {
3866       \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}
3867     }
3868     \tl_put_right:Nx \l_tmpa_tl {
3869       \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
3870         \stex_invoke_symbol:n {
3871           \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3872         }
3873       }
3874     }
3875   }
3876 }
3877 \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
3878   \tl_put_right:Nx \l_tmpc_tl {
3879     \stex_annotate_invisible:nnn{definiens}{\use:c{l__stex_copymodule_copymodule_##1?####1_def_tl}}
3880   }
3881 }
3882 \tl_put_right:Nx \l_tmpb_tl {
3883   \stex_annotate:nnn{assignment} {##1?####1} { \l_tmpc_tl }
3884 }
3885 }
3886 }
3887 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3888 \tl_put_left:Nx \l_tmpa_tl {
3889   \prop_set_from_keyval:cn {
3890     l_stex_copymodule_ \l_stex_current_module_str? \l_stex_current_copymodule_name_str _prop
3891   }{
3892     \prop_to_keyval:N \l_stex_current_copymodule_prop
3893   }
3894 }
3895 \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl

```

```

3896 \stex_debug:nn{copymodule}{result:\meaning \l_tmpa_tl}
3897 \exp_args:Nx \stex_do_up_to_module:n {
3898   \exp_args:No \exp_not:n \l_tmpa_tl
3899 }
3900 \l_tmpb_tl
3901 \stex_if_smsmode:F {
3902   \end{stex_annotate_env}
3903 }
3904 }
3905
3906 \NewDocumentEnvironment {copymodule} { 0{} m m}{
3907   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ structure }
3908   \stex_deactivate_macro:Nn \symdecl {module~environments}
3909   \stex_deactivate_macro:Nn \symdef {module~environments}
3910   \stex_deactivate_macro:Nn \notation {module~environments}
3911   \stex_reactivate_macro:N \assign
3912   \stex_reactivate_macro:N \renamedec1
3913   \stex_reactivate_macro:N \donotcopy
3914   \stex_smsmode_do:
3915 }{
3916   \stex_copymodule_end:n {}
3917 }
3918
3919 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
3920   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ realization }
3921   \stex_deactivate_macro:Nn \symdecl {module~environments}
3922   \stex_deactivate_macro:Nn \symdef {module~environments}
3923   \stex_deactivate_macro:Nn \notation {module~environments}
3924   \stex_reactivate_macro:N \assign
3925   \stex_reactivate_macro:N \renamedec1
3926   \stex_reactivate_macro:N \donotcopy
3927   \stex_smsmode_do:
3928 }{
3929   \stex_copymodule_end:n {
3930     \tl_if_exist:cF {
3931       l__stex_copymodule_copymodule_##1?##2_def_tl
3932     }{
3933       \msg_error:nnxx{stex}{error/interpretmodule/nodedefiniens}{
3934         ##1?##2
3935       }{\l_stex_current_copymodule_name_str}
3936     }
3937   }
3938 }
3939
3940 \NewDocumentCommand \donotcopy { 0{} m}{
3941   \stex_import_module_uri:nn { #1 } { #2 }
3942   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3943   \seq_map_inline:Nn \l_stex_collect_imports_seq {
3944     \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
3945     \seq_map_inline:cn {c_stex_module_##1_constants}{
3946       \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? #####1 }
3947       \bool_lazy_any_p:nT {
3948         { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_name_str}}
3949         { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_macroname_str}}

```

```

3950         { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_def_tl}}
3951     }{
3952         % TODO throw error
3953     }
3954 }
3955 }
3956
3957 \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
3958 \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_ns_str ?\l_stex_import_name_str }
3959 \prop_put:Nnx \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
3960 }
3961
3962 \NewDocumentCommand \assign { m m }{
3963     \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
3964     \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
3965     \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _def_tl}{#2}
3966 }
3967
3968 \keys_define:nn { stex / renamedec1 } {
3969     name .str_set_x:N = \l_stex_renamedec1_name_str
3970 }
3971 \cs_new_protected:Nn \__stex_copymodule_renamedec1_args:n {
3972     \str_clear:N \l_stex_renamedec1_name_str
3973     \keys_set:nn { stex / renamedec1 } { #1 }
3974 }
3975
3976 \NewDocumentCommand \renamedec1 { O{} m m }{
3977     \__stex_copymodule_renamedec1_args:n { #1 }
3978     \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
3979     \stex_debug:nn{renamedec1}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
3980     \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
3981     \str_if_empty:NTF \l_stex_renamedec1_name_str {
3982         \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3983             \l_stex_get_symbol_uri_str
3984         } }
3985     } {
3986         \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex
3987         \stex_debug:nn{renamedec1}{@~\l_stex_current_module_str ? \l_stex_renamedec1_name_str}
3988         \prop_set_eq:cc {l_stex_symdecl_
3989             \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3990             _prop
3991         }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
3992         \seq_set_eq:cc {l_stex_symdecl_
3993             \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3994             _notations
3995         }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
3996         \prop_put:cnx {l_stex_symdecl_
3997             \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3998             _prop
3999         }{ name }{ \l_stex_renamedec1_name_str }
4000         \prop_put:cnx {l_stex_symdecl_
4001             \l_stex_current_module_str ? \l_stex_renamedec1_name_str
4002             _prop
4003         }{ module }{ \l_stex_current_module_str }

```

```

4004 \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
4005 \l_stex_current_module_str ? \l_stex_renameddecl_name_str
4006 }
4007 \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4008 \l_stex_current_module_str ? \l_stex_renameddecl_name_str
4009 } }
4010 }
4011 }
4012
4013 \stex_deactivate_macro:Nn \assign {copymodules}
4014 \stex_deactivate_macro:Nn \renameddecl {copymodules}
4015 \stex_deactivate_macro:Nn \donotcopy {copymodules}
4016
4017
4018 \seq_new:N \l_stex_implicit_morphisms_seq
4019 \NewDocumentCommand \implicitmorphism { 0{} m m}{
4020 \stex_import_module_uri:nn { #1 } { #2 }
4021 \stex_debug:nn{implicits}{
4022 Implicit~morphism:~
4023 \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4024 }
4025 \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
4026 \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4027 }{
4028 \msg_error:nnn{stex}{error/conflictingmodules}{
4029 \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4030 }
4031 }
4032
4033 % TODO
4034
4035
4036
4037 \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
4038 \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4039 }
4040 }
4041

```

## 30.2 The feature environment

structural@feature

```

4042 <@@=stex_features>
4043
4044 \NewDocumentEnvironment{structural_feature_module}{ m m m }{
4045 \stex_if_in_module:F {
4046 \msg_set:nnn{stex}{error/nomodule}{
4047 Structural~Feature~has~to~occur~in~a~module:\\
4048 Feature~#2~of~type~#1\\
4049 In~File::~\stex_path_to_string:N \g_stex_currentfile_seq
4050 }
4051 \msg_error:nn{stex}{error/nomodule}
4052 }

```



```

4053
4054 \stex_module_setup:nn{meta=NONE}{#2 - #1}
4055
4056 \stex_if_smsmode:F {
4057   \begin{stex_annotate_env}{ feature:#1 }{}
4058   \stex_annotate_invisible:nnn{header}{}{ #3 }
4059 }
4060 }{
4061   \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
4062   \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
4063   \stex_debug:nn{features}{
4064     Feature: \l_stex_last_feature_str
4065   }
4066   \stex_if_smsmode:F {
4067     \end{stex_annotate_env}
4068   }
4069 }

```

### 30.3 Structure

structure

```

4070 <@@=stex_structures>
4071 \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
4072   \prop_if_exist:cF {c_stex_module_ \l_stex_current_module_str _structures}{
4073     \prop_new:c {c_stex_module_ \l_stex_current_module_str _structures}
4074   }
4075   \prop_gput:cxx{c_stex_module_ \l_stex_current_module_str _structures}
4076     {#1}{#2}
4077 }
4078
4079 \keys_define:nn { stex / features / structure } {
4080   name .str_set_x:N = \l__stex_structures_name_str ,
4081 }
4082
4083 \cs_new_protected:Nn \__stex_structures_structure_args:n {
4084   \str_clear:N \l__stex_structures_name_str
4085   \keys_set:nn { stex / features / structure } { #1 }
4086 }
4087
4088 \NewDocumentEnvironment{mathstructure}{m O{}}{
4089   \__stex_structures_structure_args:n { #2 }
4090   \str_if_empty:NT \l__stex_structures_name_str {
4091     \str_set:Nx \l__stex_structures_name_str { #1 }
4092   }
4093   \exp_args:Nnnx
4094   \begin{structural_feature_module}{ structure }
4095     { \l__stex_structures_name_str }{}
4096   \stex_smsmode_do:
4097 }{
4098   \end{structural_feature_module}
4099   \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
4100   \seq_clear:N \l_tmpa_seq
4101   \seq_map_inline:Nn \l_stex_collect_imports_seq {

```

```

4102     \seq_map_inline:cn{c_stex_module_##1_constants}{
4103       \seq_put_right:Nn \l_tmpa_seq { ##1 ? ####1 }
4104     }
4105   }
4106   \exp_args:Nnno
4107   \prop_gput:cnn {c_stex_module_ \l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
4108   \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
4109   \stex_add_structure_to_current_module:nn
4110     \l__stex_structures_name_str
4111     \l_stex_last_feature_str
4112   \exp_args:Nx \stex_symdecl_do:nn {
4113     name = \l__stex_structures_name_str ,
4114     type = \metacollection ,
4115     def = {\STEXsymbol{module-type}{
4116       \stex_term_math_oms:nnnn { \l_stex_last_feature_str }{}{0}{}}
4117     }}
4118   }{ #1 }
4119   \exp_args:Nx
4120   \stex_add_to_current_module:n {
4121     \tl_set:cn { #1 }{
4122       \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures
4123     }
4124   }
4125   \exp_args:Nx
4126   \stex_do_up_to_module:n {
4127     \tl_set:cn { #1 }{
4128       \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures
4129     }
4130   }
4131 }
4132 \seq_put_right:Nx \g_stex_smsmode_allowedenvs_seq { \tl_to_str:n {mathstructure}}
4133
4134 \cs_new:Nn \stex_invoke_structure:nn {
4135   \stex_invoke_symbol:n { #1?#2 }
4136 }
4137
4138 \cs_new_protected:Nn \stex_get_structure:n {
4139   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4140     \tl_set:Nn \l_tmpa_tl { #1 }
4141     \__stex_structures_get_from_cs:
4142   }{
4143     \cs_if_exist:cTF { #1 }{
4144       \cs_set_eq:Nc \l_tmpa_cs { #1 }
4145       \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
4146       \str_if_empty:NNTF \l_tmpa_str {
4147         \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs } \stex_invoke_structure:nn {
4148           \__stex_structures_get_from_cs:
4149         }{
4150           \__stex_structures_get_from_string:n { #1 }
4151         }
4152       }{
4153         \__stex_structures_get_from_string:n { #1 }
4154       }
4155     }{

```

```

4156     \__stex_structures_get_from_string:n { #1 }
4157   }
4158 }
4159 }
4160
4161 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
4162   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4163     { \tl_tail:N \l_tmpa_tl }
4164   \str_set:Nx \l_tmpa_str {
4165     \exp_after:wN \use_i:nn \l_tmpa_tl
4166   }
4167   \str_set:Nx \l_tmpb_str {
4168     \exp_after:wN \use_ii:nn \l_tmpa_tl
4169   }
4170   \str_set:Nx \l_stex_get_structure_str {
4171     \l_tmpa_str ? \l_tmpb_str
4172   }
4173   \str_set:Nx \l_stex_get_structure_module_str {
4174     \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
4175   }
4176 }
4177
4178 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
4179   \tl_set:Nn \l_tmpa_tl {
4180     \msg_error:nnn{stex}{error/unknownstructure}{#1}
4181   }
4182   \str_set:Nn \l_tmpa_str { #1 }
4183   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4184
4185   \seq_map_inline:Nn \l_stex_all_modules_seq {
4186     \prop_if_exist:cT {c_stex_module_##1_structures} {
4187       \prop_map_inline:cn {c_stex_module_##1_structures} {
4188         \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?###1}{-\l_tmpa_int}{-1}}{
4189           \prop_map_break:n{\seq_map_break:n{
4190             \tl_set:Nn \l_tmpa_tl {
4191               \str_set:Nn \l_stex_get_structure_str {##1?###1}
4192               \str_set:Nn \l_stex_get_structure_module_str {####2}
4193             }
4194           }}
4195         }
4196       }
4197     }
4198   }
4199   \l_tmpa_tl
4200 }

```

\instantiate

```

4201
4202 \keys_define:nn { stex / instantiate } {
4203   name .str_set_x:N = \l__stex_structures_name_str
4204 }
4205 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
4206   \str_clear:N \l__stex_structures_name_str
4207   \keys_set:nn { stex / instantiate } { #1 }

```

```

4208 }
4209
4210 \NewDocumentCommand \instantiate {m O{} m m m}{
4211   \beginingroup
4212     \stex_get_structure:n {#4}
4213     \__stex_structures_instantiate_args:n { #2 }
4214     \str_if_empty:NT \l__stex_structures_name_str {
4215       \str_set:Nn \l__stex_structures_name_str { #1 }
4216     }
4217     \seq_clear:N \l__stex_structures_fields_seq
4218     \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4219     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4220       \seq_map_inline:cn {c_stex_module_##1_constants}{
4221         \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? #####1 }
4222       }
4223     }
4224     \seq_set_split:Nnn \l_tmpa_seq , {#3}
4225     \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4226     \prop_clear:N \l_tmpa_prop
4227     \seq_map_inline:Nn \l_tmpa_seq {
4228       \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4229       \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4230         \msg_error:nnn{stex}{error/keyval}{##1}
4231       }
4232       \exp_args:Nx \stex_get_symbol_in_seq:nn { \seq_item:Nn \l_tmpb_seq 1} \l__stex_structur
4233       \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4234       \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_uri
4235       \exp_args:Nx \stex_get_symbol:n { \seq_item:Nn \l_tmpb_seq 2}
4236       \exp_args:Nxx \str_if_eq:nnF
4237         { \prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4238         { \prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}{
4239         \msg_error:nnxxx{stex}{error/incompatible}
4240         { \l__stex_structures_dom_str }
4241         { \prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4242         { \l_stex_get_symbol_uri_str }
4243         { \prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}
4244       }
4245       \prop_put:Nxx \l_tmpa_prop { \seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
4246     }
4247     \seq_if_empty:NF \l__stex_structures_fields_seq {
4248       \msg_error:nnx{stex}{error/instantiate/missing}{ \seq_use:Nn \l__stex_structures_fields_
4249     }
4250     \exp_args:Nx
4251     \stex_add_to_current_module:n {
4252       \prop_set_from_keyval:cn {l_stex_instance\l_stex_current_module_str?\l__stex_structur
4253       domain = \l_stex_get_structure_module_str ,
4254       \prop_to_keyval:N \l_tmpa_prop
4255     }
4256     \tl_set:cn{ #1 }{ \stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
4257   }
4258   \exp_args:Nx
4259   \stex_do_up_to_module:n {
4260     \prop_set_from_keyval:cn {l_stex_instance\l_stex_current_module_str?\l__stex_structur
4261     domain = \l_stex_get_structure_module_str ,

```

```

4262     \prop_to_keyval:N \l_tmpa_prop
4263   }
4264   \tl_set:cn{ #1 }{\stex_invoke_instance:n{\l_stex_current_module_str?\l__stex_structures_name_str}{\exp_not:n{\comp{#5}}}}
4265   \notation{\l__stex_structures_name_str}{\exp_not:n{\comp{#5}}}}
4266 }
4267 \exp_args:Nxx \stex_symdecl_do:nn {
4268   type={\STEXsymbol{module-type}}{
4269     \stex_term_math_oms:nnnn {
4270       \l_stex_get_structure_module_str
4271     }{}{0}{}
4272   }}
4273   }{\l__stex_structures_name_str}
4274 \endgroup
4275 \stex_smsmode_do:\ignorespacesandpars
4276 }
4277 \tl_put_right:Nx \g_stex_smsmode_allowedmacros_escape_tl {\instantiate}
4278
4279 \cs_new_protected:Nn \stex_symbol_or_var:n {
4280   \cs_if_exist:cTF{#1}{
4281     \cs_set_eq:Nc \l_tmpa_tl { #1 }
4282     \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4283     \str_if_empty:NTF \l_tmpa_str {
4284       \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
4285       \stex_invoke_variable:n {
4286         \bool_set_true:N \l_stex_symbol_or_var_bool
4287         \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4288         \str_set:Nx \l_stex_get_symbol_uri_str {
4289           \exp_after:wN \use:n \l_tmpa_tl
4290         }
4291       }{
4292         \bool_set_false:N \l_stex_symbol_or_var_bool
4293         \stex_get_symbol:n{#1}
4294       }
4295     }{
4296       \__stex_structures_symbolorvar_from_string:n{ #1 }
4297     }
4298   }{
4299     \__stex_structures_symbolorvar_from_string:n{ #1 }
4300   }
4301 }
4302
4303 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
4304   \prop_if_exist:cTF {l_stex_variable_#1 _prop}{
4305     \bool_set_true:N \l_stex_symbol_or_var_bool
4306     \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
4307   }{
4308     \bool_set_false:N \l_stex_symbol_or_var_bool
4309     \stex_get_symbol:n{#1}
4310   }
4311 }
4312
4313
4314 \NewDocumentCommand \varinstantiate {m O{} m m m}{
4315   \begingroup

```

```

4316 \stex_get_structure:n {#4}
4317 \__stex_structures_instantiate_args:n { #2 }
4318 \str_if_empty:NT \l__stex_structures_name_str {
4319   \str_set:Nn \l__stex_structures_name_str { #1 }
4320 }
4321 \seq_clear:N \l__stex_structures_fields_seq
4322 \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4323 \seq_map_inline:Nn \l_stex_collect_imports_seq {
4324   \seq_map_inline:cn {c_stex_module_##1_constants}{
4325     \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? #####1 }
4326   }
4327 }
4328 \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4329 \prop_clear:N \l_tmpa_prop
4330 \tl_if_empty:nF {#3} {
4331   \seq_set_split:Nnn \l_tmpa_seq , {#3}
4332   \seq_map_inline:Nn \l_tmpa_seq {
4333     \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4334     \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4335       \msg_error:nnn{stex}{error/keyval}{##1}
4336     }
4337     \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
4338     \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4339     \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
4340     \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
4341     \bool_if:NTF \l_stex_symbol_or_var_bool {
4342       \exp_args:Nxx \str_if_eq:nnF
4343         {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4344         {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}{
4345       \msg_error:nnxxx{stex}{error/incompatible}
4346       {\l__stex_structures_dom_str}
4347       {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4348       {\l_stex_get_symbol_uri_str}
4349       {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}
4350     }
4351     \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n {
4352   }{
4353     \exp_args:Nxx \str_if_eq:nnF
4354       {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4355       {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
4356     \msg_error:nnxxx{stex}{error/incompatible}
4357     {\l__stex_structures_dom_str}
4358     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4359     {\l_stex_get_symbol_uri_str}
4360     {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
4361   }
4362   \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n {\l
4363 }
4364 }
4365 }
4366 \tl_gclear:N \g__stex_structures_aftergroup_tl
4367 \seq_map_inline:Nn \l__stex_structures_fields_seq {
4368   \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdecl_
4369   \stex_find_notation:nn{##1}{}}

```

```

4370 \cs_gset_eq:cc{g__stex_structures_tmpa_\l_tmpa_str_cs}
4371 {stex_notation_##1\c_hash_str \l_stex_notation_variant_str_cs}
4372 \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str_cs}{
4373 \cs_gset_eq:cc {g__stex_structures_tmpa_op_\l_tmpa_str_cs}
4374 {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str_cs}
4375 }
4376
4377 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4378 \prop_set_from_keyval:cn { l_stex_variable_ \l_tmpa_str_prop}{
4379 name = \l_tmpa_str ,
4380 args = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4381 arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4382 assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4383 }
4384 \cs_set_eq:cc {stex_var_notation_\l_tmpa_str_cs}
4385 {g__stex_structures_tmpa_\l_tmpa_str_cs}
4386 \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str_cs}
4387 {g__stex_structures_tmpa_op_\l_tmpa_str_cs}
4388 }
4389 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_invoke
4390 }
4391 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4392 \prop_set_from_keyval:cn {l_stex_varinstance_\l__stex_structures_name_str_prop}{
4393 domain = \l_stex_get_structure_module_str ,
4394 \prop_to_keyval:N \l_tmpa_prop
4395 }
4396 \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
4397 \tl_set:cn {l_stex_varinstance_\l__stex_structures_name_str_op_tl}{
4398 \exp_args:Nnx \exp_not:N \use:nn {
4399 \str_set:Nn \exp_not:N \l_stex_current_symbol_str {var://\l__stex_structures_name_
4400 \stex_term_omv:nn {var://\l__stex_structures_name_str}{
4401 \exp_not:n{
4402 \varcomp{#5}
4403 }
4404 }
4405 }{
4406 \exp_not:n{\stex_reset:N \l_stex_current_symbol_str}
4407 }
4408 }
4409 }
4410 \aftergroup\g__stex_structures_aftergroup_tl
4411 \endgroup
4412 \stex_smsmode_do:\ignorespacesandpars
4413 }
4414
4415 \cs_new_protected:Nn \stex_invoke_instance:n {
4416 \peek_charcode_remove:NTF ! {
4417 \STEXsymbol{?#1}
4418 }{
4419 \stex_invoke_instance:nn {#1}
4420 }
4421 }
4422
4423

```

```

4424 \cs_new_protected:Nn \stex_invoke_varinstance:n {
4425   \peek_charcode_remove:NTF ! {
4426     \use:c{l_stex_varinstance_#1_op_tl}
4427   }{
4428     \stex_invoke_varinstance:nn {#1}
4429   }
4430 }
4431
4432 \cs_new_protected:Nn \stex_invoke_instance:nn {
4433   \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
4434     \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
4435   }{
4436     \msg_error:nnnn{stex}{error/unknownfield}{#2}{#1}
4437   }
4438 }
4439
4440 \cs_new_protected:Nn \stex_invoke_varinstance:nn {
4441   \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
4442     \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
4443     \l_tmpa_tl
4444   }{
4445     \msg_error:nnnn{stex}{error/unknownfield}{#2}{#1}
4446   }
4447 }

```

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

\stex\_invoke\_structure:nnn

```

4448 % #1: URI of the instance
4449 % #2: URI of the instantiated module
4450 \cs_new_protected:Nn \stex_invoke_structure:nnn {
4451   \tl_if_empty:nTF{ #3 }{
4452     \prop_set_eq:Nc \l__stex_structures_structure_prop {
4453       c_stex_feature_ #2 _prop
4454     }
4455     \tl_clear:N \l_tmpa_tl
4456     \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
4457     \seq_map_inline:Nn \l_tmpa_seq {
4458       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
4459       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
4460       \cs_if_exist:cT {
4461         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
4462       }{
4463         \tl_if_empty:NF \l_tmpa_tl {
4464           \tl_put_right:Nn \l_tmpa_tl {,}
4465         }
4466         \tl_put_right:Nx \l_tmpa_tl {
4467           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
4468         }
4469       }
4470     }
4471     \exp_args:No \mathstrut \l_tmpa_tl
4472   }{
4473     \stex_invoke_symbol:n{#1/#3}

```



```

4474 }
4475 }

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

4476 </package>

```

## Chapter 31

# STEX -Statements Implementation

```
4477 <*package>
4478
4479 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4480
4481 <@@=stex_statements>
    Warnings and error messages
4482
\titleemph
4483 \def\titleemph#1{\textbf{#1}}
    (End definition for \titleemph. This function is documented on page ??.)
```

### 31.1 Definitions

#### definiendum

```
4484 \keys_define:nn {stex / definiendum }{
4485   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
4486   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
4487   root     .str_set_x:N    = \l__stex_statements_definiendum_root_str,
4488   gfa      .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
4489 }
4490 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
4491   \str_clear:N \l__stex_statements_definiendum_root_str
4492   \tl_clear:N \l__stex_statements_definiendum_post_tl
4493   \str_clear:N \l__stex_statements_definiendum_gfa_str
4494   \keys_set:nn { stex / definiendum }{ #1 }
4495 }
4496 \NewDocumentCommand \definiendum { O{} m m } {
4497   \__stex_statements_definiendum_args:n { #1 }
4498   \stex_get_symbol:n { #2 }
4499   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4500   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
4501     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```

4502     \tl_set:Nn \l_tmpa_tl { #3 }
4503   } {
4504     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
4505     \tl_set:Nn \l_tmpa_tl {
4506       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
4507     }
4508   }
4509 } {
4510   \tl_set:Nn \l_tmpa_tl { #3 }
4511 }
4512
4513 % TODO root
4514 \rustex_if:TF {
4515   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
4516 } {
4517   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
4518 }
4519 }
4520 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

(End definition for definiendum. This function is documented on page 6.)

## definame

```

4521
4522 \NewDocumentCommand \definame { 0{ } m } {
4523   \__stex_statements_definiendum_args:n { #1 }
4524   % TODO: root
4525   \stex_get_symbol:n { #2 }
4526   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4527   \str_set:Nx \l_tmpa_str {
4528     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4529   }
4530   \str_replace_all:Nnn \l_tmpa_str {-} {~}
4531   \rustex_if:TF {
4532     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4533       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4534     }
4535   } {
4536     \exp_args:Nnx \defemph@uri {
4537       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4538     } { \l_stex_get_symbol_uri_str }
4539   }
4540 }
4541 \stex_deactivate_macro:Nn \definame {definition~environments}
4542
4543 \NewDocumentCommand \Definame { 0{ } m } {
4544   \__stex_statements_definiendum_args:n { #1 }
4545   \stex_get_symbol:n { #2 }
4546   \str_set:Nx \l_tmpa_str {
4547     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4548   }
4549   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4550   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4551   \rustex_if:TF {

```

```

4552 \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4553 \l_tmpa_str\l__stex_statements_definiendum_post_tl
4554 }
4555 } {
4556 \exp_args:Nnx \defemph@uri {
4557 \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
4558 } { \l_stex_get_symbol_uri_str }
4559 }
4560 }
4561 \stex_deactivate_macro:Nn \Definame {definition~environments}
4562
4563 \NewDocumentCommand \premise { m }{
4564 \stex_annotate:nnn{ premise }{}{ #1 }
4565 }
4566 \NewDocumentCommand \conclusion { m }{
4567 \stex_annotate:nnn{ conclusion }{}{ #1 }
4568 }
4569 \NewDocumentCommand \definiens { m }{
4570 \stex_annotate:nnn{ definiens }{}{ #1 }
4571 }
4572
4573 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
4574 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
4575 \stex_deactivate_macro:Nn \definiens {definition~environments}
4576

```

(End definition for `definame`. This function is documented on page 6.)

## sdefinition

```

4577
4578 \keys_define:nn {stex / sdefinition }{
4579 type .str_set_x:N = \sdefinitiontype,
4580 id .str_set_x:N = \sdefinitionid,
4581 name .str_set_x:N = \sdefinitionname,
4582 for .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4583 title .tl_set:N = \sdefinitiontitle
4584 }
4585 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
4586 \str_clear:N \sdefinitiontype
4587 \str_clear:N \sdefinitionid
4588 \str_clear:N \sdefinitionname
4589 \clist_clear:N \l__stex_statements_sdefinition_for_clist
4590 \tl_clear:N \sdefinitiontitle
4591 \keys_set:nn { stex / sdefinition }{}{ #1 }
4592 }
4593
4594 \NewDocumentEnvironment{sdefinition}{0{}}{
4595 \__stex_statements_sdefinition_args:n{ #1 }
4596 \stex_reactivate_macro:N \definiendum
4597 \stex_reactivate_macro:N \definame
4598 \stex_reactivate_macro:N \Definame
4599 \stex_reactivate_macro:N \premise
4600 \stex_reactivate_macro:N \definiens
4601 \stex_if_smsmode:F{

```

```

4602 \seq_clear:N \l_tmpa_seq
4603 \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4604   \tl_if_empty:nF{ ##1 }{
4605     \stex_get_symbol:n { ##1 }
4606     \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4607       \l_stex_get_symbol_uri_str
4608     }
4609   }
4610 }
4611 \exp_args:Nnnx
4612 \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {,}}
4613 \str_if_empty:NF \sdefinitiontype {
4614   \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{}
4615 }
4616 \clist_set:No \l_tmpa_clist \sdefinitiontype
4617 \tl_clear:N \l_tmpa_tl
4618 \clist_map_inline:Nn \l_tmpa_clist {
4619   \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
4620     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
4621   }
4622 }
4623 \tl_if_empty:NTF \l_tmpa_tl {
4624   \__stex_statements_sdefinition_start:
4625 }{
4626   \l_tmpa_tl
4627 }
4628 }
4629 \stex_ref_new_doc_target:n \sdefinitionid
4630 \stex_smsmode_do:
4631 }{
4632   \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
4633   \stex_if_smsmode:F {
4634     \clist_set:No \l_tmpa_clist \sdefinitiontype
4635     \tl_clear:N \l_tmpa_tl
4636     \clist_map_inline:Nn \l_tmpa_clist {
4637       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
4638         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
4639       }
4640     }
4641     \tl_if_empty:NTF \l_tmpa_tl {
4642       \__stex_statements_sdefinition_end:
4643     }{
4644       \l_tmpa_tl
4645     }
4646     \end{stex_annotate_env}
4647   }
4648 }

```

\stexpatchdefinition

```

4649 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
4650   \par\noindent\titllemph{Definition\tl_if_empty:NF \sdefinitiontitle {
4651     ~(\sdefinitiontitle)
4652   }~}
4653 }

```

```

4654 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\par\medskip}
4655
4656 \newcommand\stexpatchdefinition[3] [] {
4657   \str_set:Nx \l_tmpa_str{ #1 }
4658   \str_if_empty:NTF \l_tmpa_str {
4659     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
4660     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
4661   }{
4662     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
4663     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
4664   }
4665 }

```

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

\inlinedef inline:

```

4666 \keys_define:nn {stex / inlinedef }{
4667   type      .str_set_x:N = \sdefinitiontype,
4668   id        .str_set_x:N = \sdefinitionid,
4669   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4670   name      .str_set_x:N = \sdefinitionname
4671 }
4672 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
4673   \str_clear:N \sdefinitiontype
4674   \str_clear:N \sdefinitionid
4675   \str_clear:N \sdefinitionname
4676   \clist_clear:N \l__stex_statements_sdefinition_for_clist
4677   \keys_set:nn { stex / inlinedef }{ #1 }
4678 }
4679 \NewDocumentCommand \inlinedef { 0{} m } {
4680   \beginngroup
4681   \__stex_statements_inlinedef_args:n{ #1 }
4682   \stex_reactivate_macro:N \definiendum
4683   \stex_reactivate_macro:N \definame
4684   \stex_reactivate_macro:N \Definame
4685   \stex_reactivate_macro:N \premise
4686   \stex_reactivate_macro:N \definiens
4687   \stex_ref_new_doc_target:n \sdefinitionid
4688   \stex_if_smsmode:TF{
4689     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
4690   }{
4691     \seq_clear:N \l_tmpa_seq
4692     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4693       \tl_if_empty:nF{ ##1 }{
4694         \stex_get_symbol:n { ##1 }
4695         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4696           \l_stex_get_symbol_uri_str
4697         }
4698       }
4699     }
4700     \exp_args:Nnx
4701     \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
4702       \str_if_empty:NF \sdefinitiontype {
4703         \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{

```

```

4704     }
4705     #2
4706     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{ }\sdefinitionname } }
4707   }
4708 }
4709 \endgroup
4710 \stex_smsmode_do:
4711 }

```

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

## 31.2 Assertions

**sassertion**

```

4712
4713 \keys_define:nn {stex / sassertion }{
4714   type      .str_set_x:N = \sassertiontype,
4715   id        .str_set_x:N = \sassertionid,
4716   title     .tl_set:N    = \sassertiontitle ,
4717   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4718   name      .str_set_x:N = \sassertionname
4719 }
4720 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
4721   \str_clear:N \sassertiontype
4722   \str_clear:N \sassertionid
4723   \str_clear:N \sassertionname
4724   \clist_clear:N \l__stex_statements_sassertion_for_clist
4725   \tl_clear:N \sassertiontitle
4726   \keys_set:nn { stex / sassertion }{ #1 }
4727 }
4728
4729 %\tl_new:N \g__stex_statements_aftergroup_tl
4730
4731 \NewDocumentEnvironment{sassertion}{0{}}{
4732   \__stex_statements_sassertion_args:n{ #1 }
4733   \stex_reactivate_macro:N \premise
4734   \stex_reactivate_macro:N \conclusion
4735   \stex_if_smsmode:F {
4736     \seq_clear:N \l_tmpa_seq
4737     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4738       \tl_if_empty:nF{ ##1 }{
4739         \stex_get_symbol:n { ##1 }
4740         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4741           \l_stex_get_symbol_uri_str
4742         }
4743       }
4744     }
4745     \exp_args:Nnnx
4746     \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}
4747     \str_if_empty:NF \sassertiontype {
4748       \stex_annotate_invisible:nnn{type}{\sassertiontype}{ }
4749     }
4750     \clist_set:Nn \l_tmpa_clist \sassertiontype

```

```

4751 \tl_clear:N \l_tmpa_tl
4752 \clist_map_inline:Nn \l_tmpa_clist {
4753   \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
4754     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
4755   }
4756 }
4757 \tl_if_empty:NTF \l_tmpa_tl {
4758   \__stex_statements_sassertion_start:
4759 }{
4760   \l_tmpa_tl
4761 }
4762 }
4763 \str_if_empty:NTF \sassertionid {
4764   \str_if_empty:NF \sassertionname {
4765     \stex_ref_new_doc_target:n {}
4766   }
4767 } {
4768   \stex_ref_new_doc_target:n \sassertionid
4769 }
4770 \stex_smsmode_do:
4771 ){
4772   \str_if_empty:NF \sassertionname {
4773     \stex_symdecl_do:nn{ }\sassertionname}
4774     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4775   }
4776   \stex_if_smsmode:F {
4777     \clist_set:Nn \l_tmpa_clist \sassertiontype
4778     \tl_clear:N \l_tmpa_tl
4779     \clist_map_inline:Nn \l_tmpa_clist {
4780       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
4781         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
4782       }
4783     }
4784     \tl_if_empty:NTF \l_tmpa_tl {
4785       \__stex_statements_sassertion_end:
4786     }{
4787       \l_tmpa_tl
4788     }
4789     \end{stex_annotate_env}
4790   }
4791 }

```

\stexpatchassertion

```

4792
4793 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
4794   \par\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
4795     (\sassertiontitle)
4796   }~}
4797 }
4798 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
4799
4800 \newcommand\stexpatchassertion[3] [] {
4801   \str_set:Nx \l_tmpa_str{ #1 }
4802   \str_if_empty:NTF \l_tmpa_str {

```



```

4803     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
4804     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
4805   }{
4806     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
4807     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
4808   }
4809 }

```

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

\inlineass inline:

```

4810 \keys_define:nn {stex / inlineass }{
4811   type      .str_set_x:N = \sassertiontype,
4812   id        .str_set_x:N = \sassertionid,
4813   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4814   name      .str_set_x:N = \sassertionname
4815 }
4816 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
4817   \str_clear:N \sassertiontype
4818   \str_clear:N \sassertionid
4819   \str_clear:N \sassertionname
4820   \clist_clear:N \l__stex_statements_sassertion_for_clist
4821   \keys_set:nn { stex / inlineass }{ #1 }
4822 }
4823 \NewDocumentCommand \inlineass { 0{} m } {
4824   \begingroup
4825   \stex_reactivate_macro:N \premise
4826   \stex_reactivate_macro:N \conclusion
4827   \__stex_statements_inlineass_args:n{ #1 }
4828   \str_if_empty:NTF \sassertionid {
4829     \str_if_empty:NF \sassertionname {
4830       \stex_ref_new_doc_target:n { }
4831     }
4832   } {
4833     \stex_ref_new_doc_target:n \sassertionid
4834   }
4835
4836   \stex_if_smsmode:TF{
4837     \str_if_empty:NF \sassertionname {
4838       \stex_symdecl_do:nn{}{\sassertionname}
4839       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4840     }
4841   }{
4842     \seq_clear:N \l_tmpa_seq
4843     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4844       \tl_if_empty:nF{ ##1 }{
4845         \stex_get_symbol:n { ##1 }
4846         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4847           \l_stex_get_symbol_uri_str
4848         }
4849       }
4850     }
4851     \exp_args:Nnx
4852     \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{

```

```

4853     \str_if_empty:NF \sassertiontype {
4854       \stex_annotate_invisible:nnn{type}{\sassertiontype}{}}
4855   }
4856   #2
4857   \str_if_empty:NF \sassertionname {
4858     \stex_symdecl_do:nn{}{\sassertionname}
4859     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4860   }
4861 }
4862 }
4863 \endgroup
4864 \stex_smsmode_do:
4865 }

```

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

## 31.3 Examples

`sexample`

```

4866
4867 \keys_define:nn {stex / sexample }{
4868   type      .str_set_x:N = \exampletype,
4869   id        .str_set_x:N = \sexampleid,
4870   title     .tl_set:N   = \sexampletitle,
4871   for       .clist_set:N = \l__stex_statements_sexample_for_clist,
4872 }
4873 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
4874   \str_clear:N \sexampletype
4875   \str_clear:N \sexampleid
4876   \tl_clear:N \sexampletitle
4877   \clist_clear:N \l__stex_statements_sexample_for_clist
4878   \keys_set:nn { stex / sexample }{ #1 }
4879 }
4880
4881 \NewDocumentEnvironment{sexample}{0{}}{
4882   \__stex_statements_sexample_args:n{ #1 }
4883   \stex_reactivate_macro:N \premise
4884   \stex_reactivate_macro:N \conclusion
4885   \stex_if_smsmode:F {
4886     \seq_clear:N \l_tmpa_seq
4887     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4888       \tl_if_empty:nF{ ##1 }{
4889         \stex_get_symbol:n { ##1 }
4890         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4891           \l_stex_get_symbol_uri_str
4892         }
4893       }
4894     }
4895     \exp_args:Nnnx
4896     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
4897     \str_if_empty:NF \sexampletype {
4898       \stex_annotate_invisible:nnn{type}{\sexampletype}{}}
4899   }

```

```

4900 \clist_set:No \l_tmpa_clist \sexamplotype
4901 \tl_clear:N \l_tmpa_tl
4902 \clist_map_inline:Nn \l_tmpa_clist {
4903   \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
4904     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
4905   }
4906 }
4907 \tl_if_empty:NTF \l_tmpa_tl {
4908   \__stex_statements_sexample_start:
4909 }{
4910   \l_tmpa_tl
4911 }
4912 }
4913 \str_if_empty:NF \sexampleid {
4914   \stex_ref_new_doc_target:n \sexampleid
4915 }
4916 \stex_smsmode_do:
4917 }{
4918   \str_if_empty:NF \sexamplename { \stex_symdecl_do:nn{}{\sexamplename} }
4919   \stex_if_smsmode:F {
4920     \clist_set:No \l_tmpa_clist \sexamplotype
4921     \tl_clear:N \l_tmpa_tl
4922     \clist_map_inline:Nn \l_tmpa_clist {
4923       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
4924         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
4925       }
4926     }
4927     \tl_if_empty:NTF \l_tmpa_tl {
4928       \__stex_statements_sexample_end:
4929     }{
4930       \l_tmpa_tl
4931     }
4932     \end{stex_annotate_env}
4933   }
4934 }

```

\stexpatchexample

```

4935
4936 \cs_new_protected:Nn \__stex_statements_sexample_start: {
4937   \par\noindent\titllemph{Example~\tl_if_empty:NF \sexamplename {
4938     (\sexamplename)
4939   }~}
4940 }
4941 \cs_new_protected:Nn \__stex_statements_sexample_end: { \par\medskip}
4942
4943 \newcommand\stexpatchexample[3]{} {
4944   \str_set:Nx \l_tmpa_str{ #1 }
4945   \str_if_empty:NTF \l_tmpa_str {
4946     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
4947     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
4948   }{
4949     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
4950     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
4951   }

```

4952 }

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

`\inlineex` inline:

```

4953 \keys_define:nn {stex / inlineex }{
4954   type      .str_set_x:N = \sexamplotype,
4955   id        .str_set_x:N = \sexampleid,
4956   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
4957   name      .str_set_x:N = \sexamplename
4958 }
4959 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
4960   \str_clear:N \sexamplotype
4961   \str_clear:N \sexampleid
4962   \str_clear:N \sexamplename
4963   \clist_clear:N \l__stex_statements_sexample_for_clist
4964   \keys_set:nn { stex / inlineex }{ #1 }
4965 }
4966 \NewDocumentCommand \inlineex { 0{ } m } {
4967   \begingroup
4968   \stex_reactivate_macro:N \premise
4969   \stex_reactivate_macro:N \conclusion
4970   \__stex_statements_inlineex_args:n{ #1 }
4971   \str_if_empty:NF \sexampleid {
4972     \stex_ref_new_doc_target:n \sexampleid
4973   }
4974   \stex_if_smsmode:TF{
4975     \str_if_empty:NF \sexamplename { \stex_symdecl_do:nn{}{\sexamplename} }
4976   }{
4977     \seq_clear:N \l_tmpa_seq
4978     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4979       \tl_if_empty:nF{ ##1 }{
4980         \stex_get_symbol:n { ##1 }
4981         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4982           \l_stex_get_symbol_uri_str
4983         }
4984       }
4985     }
4986     \exp_args:Nnx
4987     \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpa_seq {},}{
4988       \str_if_empty:NF \sexamplotype {
4989         \stex_annotate_invisible:nnn{type}{\sexamplotype}{}
4990       }
4991       #2
4992       \str_if_empty:NF \sexamplename { \stex_symdecl_do:nn{}{\sexamplename} }
4993     }
4994   }
4995   \endgroup
4996   \stex_smsmode_do:
4997 }

```

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

## 31.4 Logical Paragraphs

sparagraph

```

4998 \keys_define:nn { stex / sparagraph } {
4999   id       .str_set_x:N = \sparagraphid ,
5000   title    .tl_set:N    = \l_stex_sparagraph_title_tl ,
5001   type     .str_set_x:N = \sparagraphtype ,
5002   for      .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5003   from     .tl_set:N    = \sparagraphfrom ,
5004   to       .tl_set:N    = \sparagraphto ,
5005   start    .tl_set:N    = \l_stex_sparagraph_start_tl ,
5006   name     .str_set:N   = \sparagraphname
5007 }
5008
5009 \cs_new_protected:Nn \stex_sparagraph_args:n {
5010   \tl_clear:N \l_stex_sparagraph_title_tl
5011   \tl_clear:N \sparagraphfrom
5012   \tl_clear:N \sparagraphto
5013   \tl_clear:N \l_stex_sparagraph_start_tl
5014   \str_clear:N \sparagraphid
5015   \str_clear:N \sparagraphtype
5016   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5017   \str_clear:N \sparagraphname
5018   \keys_set:nn { stex / sparagraph } { #1 }
5019 }
5020 \newif\if@in@omtext\@in@omtextfalse
5021
5022 \NewDocumentEnvironment {sparagraph} { 0{ } } {
5023   \stex_sparagraph_args:n { #1 }
5024   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5025     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
5026   }{
5027     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
5028   }
5029   \@in@omtexttrue
5030   \stex_if_smsmode:F {
5031     \seq_clear:N \l_tmpa_seq
5032     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5033       \tl_if_empty:NF{ ##1 }{
5034         \stex_get_symbol:n { ##1 }
5035         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5036           \l_stex_get_symbol_uri_str
5037         }
5038       }
5039     }
5040     \exp_args:Nnnx
5041     \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
5042     \str_if_empty:NF \sparagraphtype {
5043       \stex_annotate_invisible:nnn{type}{\sparagraphtype}{ }
5044     }
5045     \str_if_empty:NF \sparagraphfrom {
5046       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{ }
5047     }
5048     \str_if_empty:NF \sparagraphto {

```

```

5049     \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5050   }
5051   \clist_set:No \l_tmpa_clist \sparagraphtype
5052   \tl_clear:N \l_tmpa_tl
5053   \clist_map_inline:Nn \sparagraphtype {
5054     \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
5055       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
5056     }
5057   }
5058   \tl_if_empty:NTF \l_tmpa_tl {
5059     \__stex_statements_sparagraph_start:
5060   }{
5061     \l_tmpa_tl
5062   }
5063 }
5064 \clist_set:No \l_tmpa_clist \sparagraphtype
5065 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5066   {
5067     \stex_reactivate_macro:N \definiendum
5068     \stex_reactivate_macro:N \definame
5069     \stex_reactivate_macro:N \Definame
5070     \stex_reactivate_macro:N \premise
5071     \stex_reactivate_macro:N \definiens
5072   }
5073   \str_if_empty:NTF \sparagraphid {
5074     \str_if_empty:NTF \sparagraphname {
5075       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5076         \stex_ref_new_doc_target:n {}
5077       }
5078     } {
5079       \stex_ref_new_doc_target:n {}
5080     }
5081   } {
5082     \stex_ref_new_doc_target:n \sparagraphid
5083   }
5084   \exp_args:NNx
5085   \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5086     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5087       \tl_if_empty:nF{ ##1 }{
5088         \stex_get_symbol:n { ##1 }
5089         \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5090       }
5091     }
5092   }
5093   \stex_smsmode_do:
5094   \ignorespacesandpars
5095 }{
5096   \str_if_empty:NF \sparagraphname {
5097     \stex_symdecl_do:nn{}{\sparagraphname}
5098     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5099   }
5100   \stex_if_smsmode:F {
5101     \clist_set:No \l_tmpa_clist \sparagraphtype
5102     \tl_clear:N \l_tmpa_tl

```

```

5103 \clist_map_inline:Nn \l_tmpa_clist {
5104   \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
5105     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
5106   }
5107 }
5108 \tl_if_empty:NTF \l_tmpa_tl {
5109   \__stex_statements_sparagraph_end:
5110 }{
5111   \l_tmpa_tl
5112 }
5113 \end{stex_annotate_env}
5114 }
5115 }

```

# \stexpatchparagraph

```

5116
5117 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
5118   \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5119     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
5120       \titleemph{\l_stex_sparagraph_title_tl}:~
5121     }
5122   }{
5123     \titleemph{\l_stex_sparagraph_start_tl}~
5124   }
5125 }
5126 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
5127
5128 \newcommand\stexpatchparagraph[3]{} {
5129   \str_set:Nx \l_tmpa_str{ #1 }
5130   \str_if_empty:NTF \l_tmpa_str {
5131     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
5132     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
5133   }{
5134     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2 }
5135     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
5136   }
5137 }
5138
5139 \keys_define:nn { stex / inlinepara } {
5140   id      .str_set_x:N = \sparagraphid ,
5141   type    .str_set_x:N = \sparagraphtype ,
5142   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5143   from    .tl_set:N    = \sparagraphfrom ,
5144   to      .tl_set:N    = \sparagraphto ,
5145   name    .str_set:N   = \sparagraphname
5146 }
5147 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
5148   \tl_clear:N \sparagraphfrom
5149   \tl_clear:N \sparagraphto
5150   \str_clear:N \sparagraphid
5151   \str_clear:N \sparagraphtype
5152   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5153   \str_clear:N \sparagraphname
5154   \keys_set:nn { stex / inlinepara }{ #1 }

```

```

5155 }
5156 \NewDocumentCommand \inlinepara { 0{} m } {
5157   \begingroup
5158   \_stex_statements_inlinepara_args:n{ #1 }
5159   \clist_set:No \l_tmpa_clist \sparagraphtype
5160   \str_if_empty:NTF \sparaagraphid {
5161     \str_if_empty:NTF \sparaagraphname {
5162       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5163         \stex_ref_new_doc_target:n {}
5164       }
5165     } {
5166       \stex_ref_new_doc_target:n {}
5167     }
5168   } {
5169     \stex_ref_new_doc_target:n \sparaagraphid
5170   }
5171   \stex_if_smsmode:TF{
5172     \str_if_empty:NF \sparaagraphname {
5173       \stex_symdecl_do:nn{}{\sparaagraphname}
5174       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparaagraphname}
5175     }
5176   }{
5177     \seq_clear:N \l_tmpa_seq
5178     \clist_map_inline:Nn \l__stex_statements_sparaagraph_for_clist {
5179       \tl_if_empty:NF{ ##1 }{
5180         \stex_get_symbol:n { ##1 }
5181         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5182           \l_stex_get_symbol_uri_str
5183         }
5184       }
5185     }
5186     \exp_args:Nnx
5187     \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
5188       \str_if_empty:NF \sparaagraphtype {
5189         \stex_annotate_invisible:nnn{type}{\sparaagraphtype}{}
5190       }
5191       \str_if_empty:NF \sparaagraphfrom {
5192         \stex_annotate_invisible:nnn{from}{\sparaagraphfrom}{}
5193       }
5194       \str_if_empty:NF \sparaagraphto {
5195         \stex_annotate_invisible:nnn{to}{\sparaagraphto}{}
5196       }
5197       \str_if_empty:NF \sparaagraphname {
5198         \stex_symdecl_do:nn{}{\sparaagraphname}
5199         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparaagraphname}
5200       }
5201       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5202         \clist_map_inline:Nn \l_tmpa_seq {
5203           \stex_ref_new_sym_target:n {##1}
5204         }
5205       }
5206     } #2
5207   }
5208 }

```



```

5209 \endgroup
5210 \stex_smsmode_do:
5211 }
5212
(End definition for \stexpatchparagraph. This function is documented on page ??.)
5213 </package>

```

## Chapter 32

# The Implementation

### 32.1 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>11</sup>

```
5214 <*package>
5215 <@@=stex_sproof>
5216
5217 %%%%%%%%%% sproof.dtx %%%%%%%%%%
5218
```

### 32.2 Proofs

We first define some keys for the proof environment.

```
5219 \keys_define:nn { stex / spf } {
5220   id          .str_set_x:N = \spfid,
5221   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,
5222   from        .tl_set:N    = \l__stex_sproof_spf_from_tl ,
5223   proofend    .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
5224   type        .str_set_x:N = \spftype,
5225   title       .tl_set:N    = \spftitle,
5226   continues   .tl_set:N    = \l__stex_sproof_spf_continues_tl,
5227   functions   .tl_set:N    = \l__stex_sproof_spf_functions_tl,
5228   method      .tl_set:N    = \l__stex_sproof_spf_method_tl
5229 }
5230 \cs_new_protected:Nn \l__stex_sproof_spf_args:n {
5231   \str_clear:N \spfid
5232   \tl_clear:N \l__stex_sproof_spf_for_tl
5233   \tl_clear:N \l__stex_sproof_spf_from_tl
5234   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
5235   \str_clear:N \spftype
5236   \tl_clear:N \spftitle
5237   \tl_clear:N \l__stex_sproof_spf_continues_tl
5238   \tl_clear:N \l__stex_sproof_spf_functions_tl
```

---

<sup>11</sup>EdNOTE: need an implementation for L<sup>A</sup>T<sub>E</sub>X<sub>M</sub>L

```

5239 \tl_clear:N \l__stex_sproof_spf_method_tl
5240 \bool_set_false:N \l__stex_sproof_inc_counter_bool
5241 \keys_set:nn { stex / spf }{ #1 }
5242 }

```

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

```

5243 \str_set:Nn\c__stex_sproof_flow_str{inline}

```

(End definition for `\c__stex_sproof_flow_str`.)

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>7</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.

```

5244 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
5245 \cs_new_protected:Npn \sproofnumber {
5246   \int_set:Nn \l_tmpa_int {1}
5247   \bool_while_do:nn {
5248     \int_compare_p:nNn {
5249       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5250     } > 0
5251   }{
5252     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
5253     \int_incr:N \l_tmpa_int
5254   }
5255 }
5256 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
5257   \int_set:Nn \l_tmpa_int {1}
5258   \bool_while_do:nn {
5259     \int_compare_p:nNn {
5260       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5261     } > 0
5262   }{
5263     \int_incr:N \l_tmpa_int
5264   }
5265   \int_compare:nNnF \l_tmpa_int = 1 {
5266     \int_decr:N \l_tmpa_int
5267   }
5268   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
5269     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1

```

---

<sup>7</sup>This gets the labeling right but only works 8 levels deep



```

5314 }
5315 \clist_if_in:NnT \l_tmpa_clist {finnish}{
5316   \input{sproof-finnish.ldf}
5317 }
5318 \clist_if_in:NnT \l_tmpa_clist {french}{
5319   \input{sproof-french.ldf}
5320 }
5321 \clist_if_in:NnT \l_tmpa_clist {russian}{
5322   \input{sproof-russian.ldf}
5323 }
5324 \makeatother
5325 }{}
5326 }

```

spfsketch

```

5327 \newcommand\spfsketch[2] [] {
5328   \beginingroup
5329   \let \premise \stex_proof_premise:
5330   \__stex_sproof_spf_args:n{#1}
5331   \stex_if_smsmode:TF {
5332     \str_if_empty:NF \spfid {
5333       \stex_ref_new_doc_target:n \spfid
5334     }
5335   }{
5336     \seq_clear:N \l_tmpa_seq
5337     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5338       \tl_if_empty:nF{ ##1 }{
5339         \stex_get_symbol:n { ##1 }
5340         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5341           \l_stex_get_symbol_uri_str
5342         }
5343       }
5344     }
5345     \exp_args:Nnx
5346     \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
5347       \str_if_empty:NF \spftype {
5348         \stex_annotate_invisible:nnn{type}{\spftype}{-}
5349       }
5350       \clist_set:No \l_tmpa_clist \spftype
5351       \tl_set:Nn \l_tmpa_tl {
5352         \titleemph{
5353           \tl_if_empty:NTF \spftitle {
5354             \spf@proofsketch@kw
5355           }{
5356             \spftitle
5357           }
5358         }::~
5359       }
5360       \clist_map_inline:Nn \l_tmpa_clist {
5361         \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5362           \tl_clear:N \l_tmpa_tl
5363         }
5364       }
5365       \str_if_empty:NF \spfid {

```

```

5366         \stex_ref_new_doc_target:n \spfid
5367     }
5368     \l_tmpa_tl #2 \sproofend
5369 }
5370 }
5371 \endgroup
5372 \stex_smsmode_do:
5373 }
5374

```

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

**spfeq** This is very similar to \spfsketch, but uses a computation array<sup>1213</sup>

```

5375 \newenvironment{spfeq}[2][]{
5376   \__stex_sproof_spf_args:n{#1}
5377   \let \premise \stex_proof_premise:
5378   \stex_if_smsmode:TF {
5379     \str_if_empty:NF \spfid {
5380       \stex_ref_new_doc_target:n \spfid
5381     }
5382   }{
5383     \seq_clear:N \l_tmpa_seq
5384     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5385       \tl_if_empty:NF{ ##1 }{
5386         \stex_get_symbol:n { ##1 }
5387         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5388           \l_stex_get_symbol_uri_str
5389         }
5390       }
5391     }
5392     \exp_args:Nnnx
5393     \begin{stex_annotate_env}{spfeq}{\seq_use:Nn \l_tmpa_seq {,}}
5394     \str_if_empty:NF \spftype {
5395       \stex_annotate_invisible:nnn{type}{\spftype}{}
5396     }
5397
5398     \clist_set:No \l_tmpa_clist \spftype
5399     \tl_clear:N \l_tmpa_tl
5400     \clist_map_inline:Nn \l_tmpa_clist {
5401       \tl_if_exist:cT {__stex_sproof_spfeq_##1_start:}{
5402         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_start:}}
5403       }
5404       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5405         \tl_set:Nn \l_tmpa_tl {\use:n{}}
5406       }
5407     }
5408     \tl_if_empty:NTF \l_tmpa_tl {
5409       \__stex_sproof_spfeq_start:
5410     }{
5411       \l_tmpa_tl
5412     }{-#2}

```

<sup>12</sup>EDNOTE: This should really be more like a tabular with an ensuremath in it. or invoke text on the last column

<sup>13</sup>EDNOTE: document above

```

5413 \str_if_empty:NF \spfid {
5414 \stex_ref_new_doc_target:n \spfid
5415 }
5416 \begin{displaymath}\begin{array}{rc1l}
5417 }
5418 \stex_smsmode_do:
5419 }{
5420 \stex_if_smsmode:F {
5421 \end{array}\end{displaymath}
5422 \clist_set:No \l_tmpa_clist \spftype
5423 \tl_clear:N \l_tmpa_tl
5424 \clist_map_inline:Nn \l_tmpa_clist {
5425 \tl_if_exist:cT {__stex_sproof_spfeq_##1_end:}{
5426 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_end:}}
5427 }
5428 }
5429 \tl_if_empty:NTF \l_tmpa_tl {
5430 \__stex_sproof_spfeq_end:
5431 }{
5432 \l_tmpa_tl
5433 }
5434 \end{stex_annotate_env}
5435 }
5436 }
5437
5438 \cs_new_protected:Nn \__stex_sproof_spfeq_start: {
5439 \titleemph{
5440 \tl_if_empty:NTF \spftitle {
5441 \spf@proof@kw
5442 }{
5443 \spftitle
5444 }
5445 }:
5446 }
5447 \cs_new_protected:Nn \__stex_sproof_spfeq_end: {\sproofend}
5448
5449 \newcommand\stexpatchspfeq[3] [] {
5450 \str_set:Nx \l_tmpa_str{ #1 }
5451 \str_if_empty:NTF \l_tmpa_str {
5452 \tl_set:Nn \__stex_sproof_spfeq_start: { #2 }
5453 \tl_set:Nn \__stex_sproof_spfeq_end: { #3 }
5454 }{
5455 \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_start:\endcsname{ #2 }
5456 \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_end:\endcsname{ #3 }
5457 }
5458 }
5459

```

(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.

```

5460 \newenvironment{sproof}[2] []{

```

```

5461 \let \premise \stex_proof_premise:
5462 \intarray_gzero:N \l__stex_sproof_counter_intarray
5463 \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
5464 \__stex_sproof_spf_args:n{#1}
5465 \stex_if_smsmode:TF {
5466   \str_if_empty:NF \spfid {
5467     \stex_ref_new_doc_target:n \spfid
5468   }
5469 }{
5470   \seq_clear:N \l_tmpa_seq
5471   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5472     \tl_if_empty:NF{ ##1 }{
5473       \stex_get_symbol:n { ##1 }
5474       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5475         \l_stex_get_symbol_uri_str
5476       }
5477     }
5478   }
5479   \exp_args:Nnnx
5480   \begin{stex_annotate_env}{sproof}{\seq_use:Nn \l_tmpa_seq {,}}
5481   \str_if_empty:NF \spftype {
5482     \stex_annotate_invisible:nnn{type}{\spftype}{}}
5483   }
5484
5485   \clist_set:No \l_tmpa_clist \spftype
5486   \tl_clear:N \l_tmpa_tl
5487   \clist_map_inline:Nn \l_tmpa_clist {
5488     \tl_if_exist:cT {__stex_sproof_sproof_##1_start:}{
5489       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_start:}}
5490     }
5491     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5492       \tl_set:Nn \l_tmpa_tl {\use:n{}}
5493     }
5494   }
5495   \tl_if_empty:NTF \l_tmpa_tl {
5496     \__stex_sproof_sproof_start:
5497   }{
5498     \l_tmpa_tl
5499   }{~#2}
5500   \str_if_empty:NF \spfid {
5501     \stex_ref_new_doc_target:n \spfid
5502   }
5503   \begin{description}
5504   }
5505   \stex_smsmode_do:
5506 }{
5507   \stex_if_smsmode:F{
5508     \end{description}
5509     \clist_set:No \l_tmpa_clist \spftype
5510     \tl_clear:N \l_tmpa_tl
5511     \clist_map_inline:Nn \l_tmpa_clist {
5512       \tl_if_exist:cT {__stex_sproof_sproof_##1_end:}{
5513         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_end:}}
5514       }

```



```

5515     }
5516     \tl_if_empty:NTF \l_tmpa_tl {
5517         \__stex_sproof_sproof_end:
5518     }{
5519         \l_tmpa_tl
5520     }
5521     \end{stex_annotate_env}
5522 }
5523 }
5524
5525 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
5526     \par\noindent\titleemph{
5527         \tl_if_empty:NTF \spftype {
5528             \spf@proof@kw
5529         }{
5530             \spftype
5531         }
5532     }:
5533 }
5534 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
5535
5536 \newcommand\stexpatchsproof[3] [] {
5537     \str_set:Nx \l_tmpa_str{ #1 }
5538     \str_if_empty:NTF \l_tmpa_str {
5539         \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
5540         \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
5541     }{
5542         \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
5543         \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
5544     }
5545 }

```

\spfidea

```

5546 \newcommand\spfidea[2] []{
5547     \__stex_sproof_spf_args:n{#1}
5548     \titleemph{
5549         \tl_if_empty:NTF \spftype {Proof~Idea}{
5550             \spftype
5551         }:
5552     }~#2
5553     \sproofend
5554 }

```

(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.

spfstep

```

5555 \newenvironment{spfstep}[1] []{
5556     \__stex_sproof_spf_args:n{#1}
5557     \stex_if_smsmode:TF {

```

```

5558 \str_if_empty:NF \spfid {
5559 \stex_ref_new_doc_target:n \spfid
5560 }
5561 }{
5562 \@in@omtexttrue
5563 \seq_clear:N \l_tmpa_seq
5564 \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5565 \tl_if_empty:nF{ ##1 }{
5566 \stex_get_symbol:n { ##1 }
5567 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5568 \l_stex_get_symbol_uri_str
5569 }
5570 }
5571 }
5572 \exp_args:Nnnx
5573 \begin{stex_annotate_env}{spfstep}{\seq_use:Nn \l_tmpa_seq {,}}
5574 \str_if_empty:NF \spftype {
5575 \stex_annotate_invisible:nnn{type}{\spftype}{}
5576 }
5577 \clist_set:No \l_tmpa_clist \spftype
5578 \tl_set:Nn \l_tmpa_tl {
5579 \item[\sproofnumber]
5580 \bool_set_true:N \l__stex_sproof_inc_counter_bool
5581 }
5582 \clist_map_inline:Nn \l_tmpa_clist {
5583 \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5584 \tl_clear:N \l_tmpa_tl
5585 }
5586 }
5587 \l_tmpa_tl
5588 \tl_if_empty:NF \spftitle {
5589 {(\titleemph{\spftitle})\enspace}
5590 }
5591 \str_if_empty:NF \spfid {
5592 \stex_ref_new_doc_target:n \spfid
5593 }
5594 }
5595 \stex_smsmode_do:
5596 \ignorespacesandpars
5597 }{
5598 \bool_if:NT \l__stex_sproof_inc_counter_bool {
5599 \__stex_sproof_inc_counter:
5600 }
5601 \stex_if_smsmode:F {
5602 \end{stex_annotate_env}
5603 }
5604 }

```

sproofcomment

```

5605 \newenvironment{sproofcomment}[1][]{
5606 \__stex_sproof_spf_args:n{#1}
5607 \clist_set:No \l_tmpa_clist \spftype
5608 \tl_set:Nn \l_tmpa_tl {
5609 \item[\sproofnumber]

```

```

5610 \bool_set_true:N \l__stex_sproof_inc_counter_bool
5611 }
5612 \clist_map_inline:Nn \l_tmpa_clist {
5613   \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5614     \tl_clear:N \l_tmpa_tl
5615   }
5616 }
5617 \l_tmpa_tl
5618 }{
5619   \bool_if:NT \l__stex_sproof_inc_counter_bool {
5620     \__stex_sproof_inc_counter:
5621   }
5622 }

```

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) `proproof` environment is started.

```

5623 \newenvironment{subproof}[2][]{
5624   \__stex_sproof_spf_args:n{#1}
5625   \stex_if_smsmode:TF{
5626     \str_if_empty:NF \spfid {
5627       \stex_ref_new_doc_target:n \spfid
5628     }
5629   }{
5630     \seq_clear:N \l_tmpa_seq
5631     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5632       \tl_if_empty:nF{ ##1 }{
5633         \stex_get_symbol:n { ##1 }
5634         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5635           \l_stex_get_symbol_uri_str
5636         }
5637       }
5638     }
5639     \exp_args:Nnnx
5640     \begin{stex_annotate_env}{subproof}{\seq_use:Nn \l_tmpa_seq {,}}
5641     \str_if_empty:NF \spftype {
5642       \stex_annotate_invisible:nnn{type}{\spftype}{ }
5643     }
5644
5645     \clist_set:No \l_tmpa_clist \spftype
5646     \tl_set:Nn \l_tmpa_tl {
5647       \item[\sproofnumber]
5648       \bool_set_true:N \l__stex_sproof_inc_counter_bool
5649     }
5650     \clist_map_inline:Nn \l_tmpa_clist {
5651       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5652         \tl_clear:N \l_tmpa_tl
5653       }
5654     }
5655     \l_tmpa_tl
5656     \tl_if_empty:NF \spftitle {
5657       {(\titleemph{\spftitle})\enspace}
5658     }

```

```

5659     {~#2}
5660     \str_if_empty:NF \spfid {
5661       \stex_ref_new_doc_target:n \spfid
5662     }
5663   }
5664   \__stex_sproof_add_counter:
5665   \stex_smsmode_do:
5666 }{
5667   \__stex_sproof_remove_counter:
5668   \bool_if:NT \l__stex_sproof_inc_counter_bool {
5669     \__stex_sproof_inc_counter:
5670   }
5671   \stex_if_smsmode:F{
5672     \end{stex_annotate_env}
5673   }
5674 }

```

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

```

5675 \newenvironment{spfcases}[2][]{
5676   \tl_if_empty:nTF{#1}{
5677     \begin{subproof}[method=by-cases]{#2}
5678   }{
5679     \begin{subproof}[#1,method=by-cases]{#2}
5680   }
5681 }{
5682   \end{subproof}
5683 }

```

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

```

5684 \newenvironment{spfcase}[2][]{
5685   \__stex_sproof_spf_args:n{#1}
5686   \stex_if_smsmode:TF {
5687     \str_if_empty:NF \spfid {
5688       \stex_ref_new_doc_target:n \spfid
5689     }
5690   }{
5691     \seq_clear:N \l_tmpa_seq
5692     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5693       \tl_if_empty:nF{ ##1 }{
5694         \stex_get_symbol:n { ##1 }
5695         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5696           \l_stex_get_symbol_uri_str
5697         }
5698       }
5699     }
5700     \exp_args:Nnnx
5701     \begin{stex_annotate_env}{spfcase}{\seq_use:Nn \l_tmpa_seq {,}}
5702     \str_if_empty:NF \spftype {
5703       \stex_annotate_invisible:nnn{type}{\spftype}{}}
5704   }
5705   \clist_set:Nn \l_tmpa_clist \spftype
5706   \tl_set:Nn \l_tmpa_tl {
5707     \item[\sproofnumber]

```

```

5708     \bool_set_true:N \l__stex_sproof_inc_counter_bool
5709   }
5710   \clist_map_inline:Nn \l_tmpa_clist {
5711     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5712       \tl_clear:N \l_tmpa_tl
5713     }
5714   }
5715   \l_tmpa_tl
5716   \tl_if_empty:nF{#2}{
5717     \titleemph{#2}:~
5718   }
5719 }
5720 \__stex_sproof_add_counter:
5721 \stex_smsmode_do:
5722 ){
5723   \__stex_sproof_remove_counter:
5724   \bool_if:NT \l__stex_sproof_inc_counter_bool {
5725     \__stex_sproof_inc_counter:
5726   }
5727   \stex_if_smsmode:F{
5728     \clist_set:No \l_tmpa_clist \spftype
5729     \tl_set:Nn \l_tmpa_tl{\sproofend}
5730     \clist_map_inline:Nn \l_tmpa_clist {
5731       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5732         \tl_clear:N \l_tmpa_tl
5733       }
5734     }
5735     \l_tmpa_tl
5736     \end{stex_annotate_env}
5737   }
5738 }

```

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

```

5739 \newcommand\spfcasesketch[3][]{
5740   \begin{spfcase}[#1]{#2}#3\end{spfcase}
5741 }

```

## 32.3 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.

```

5742 \keys_define:nn { stex / just }{
5743   id      .str_set:x:N = \l__stex_sproof_just_id_str,
5744   method  .tl_set:N    = \l__stex_sproof_just_method_tl,
5745   premises .tl_set:N    = \l__stex_sproof_just_premises_tl,
5746   args    .tl_set:N    = \l__stex_sproof_just_args_tl
5747 }

```

The next three environments and macros are purely semantic, so we ignore the keyval arguments for now and only display the content.<sup>14</sup>

<sup>14</sup>EdNOTE: need to do something about the premise in draft mode.

**justification**

```
5748 \newenvironment{justification}[1] [] {}{}
```

**\premise**

```
5749 \newcommand\stex_proof_promise:[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.

```
5750 \newcommand\justarg[2] [] {#2}
```

```
5751 \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.

## Chapter 33

# STEX -Others Implementation

```
5752 <*package>
5753
5754 %%%%%%%%%% others.dtx %%%%%%%%%%
5755
5756 <@@=stex_others>
    Warnings and error messages
5757 % None
\MSC Math subject classifier
5758 \NewDocumentCommand \MSC {m} {
5759 % TODO
5760 }
    (End definition for \MSC. This function is documented on page ??.)
    Patching tikzinput, if loaded
5761 \@ifpackageloaded{tikzinput}{
5762 \RequirePackage{stex-tikzinput}
5763 }{}
5764 </package>
```

## Chapter 34

# STEX -Metatheory Implementation

```
5765 <*package>
5766 <@@=stex_modules>
5767
5768 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
5769
5770 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
5771 \begingroup
5772 \stex_module_setup:nn{
5773   ns=\c_stex_metatheory_ns_str,
5774   meta=NONE
5775 }{Metatheory}
5776 \stex_reactivate_macro:N \symdecl
5777 \stex_reactivate_macro:N \notation
5778 \stex_reactivate_macro:N \symdef
5779 \ExplSyntaxOff
5780 \csname stex_suppress_html:n\endcsname{
5781   % is-a (a:A, a \in A, a is an A, etc.)
5782   \symdecl{isa}[args=ai]
5783   \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
5784   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
5785   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
5786
5787   % bind (\forall, \Pi, \lambda etc.)
5788   \symdecl{bind}[args=Bi]
5789   \notation{bind}{forall}{\comp\forall #1.;#2}{##1 \comp, ##2}
5790   \notation{bind}{Pi}{\comp\prod_{#1}#2}{##1 \comp, ##2}
5791   \notation{bind}{depfun}{\comp( #1 \comp{ }\;\to\; ) #2}{##1 \comp, ##2}
5792
5793   % implicit bind
5794   \symdef{implicitbind}[args=Bi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
5795
5796   % dummy variable
5797   \symdecl{dummyvar}
5798   \notation{dummyvar}[underscore]{\comp\_}
5799   \notation{dummyvar}[dot]{\comp\cdot}
```



```

5800 \notation{dummyvar}[dash]{\comp{\rm --}}
5801
5802 %fromto (function space, Hom-set, implication etc.)
5803 \symdecl{fromto}[args=ai]
5804 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
5805 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
5806
5807 % mapto (lambda etc.)
5808 \symdecl{mapto}[args=Bi]
5809 %\notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
5810 %\notation{mapto}[lambda]{\comp\lambda #1 \comp.; #2}{#1 \comp, #2}
5811 %\notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.; #2}{#1 \comp, #2}
5812
5813 % function/operator application
5814 \symdecl{apply}[args=ia]
5815 \notation{apply}[prec=0;0x\infpres,parens]{#1 \comp( #2 \comp)}{##1 \comp, ##2}
5816 \notation{apply}[prec=0;0x\infpres,lambda]{#1 \; #2 }{##1 \; ; ##2}
5817
5818 % ‘‘type’’ of all collections (sets,classes,types,kinds)
5819 \symdecl{metacollection}
5820 \notation{metacollection}[U]{\comp{\mathcal{U}}}
5821 \notation{metacollection}[set]{\comp{\textsf{Set}}}
5822
5823 % collection of propositions/booleans/truth values
5824 \symdecl{prop}[name=proposition]
5825 \notation{prop}[prop]{\comp{\rm prop}}
5826 \notation{prop}[B00L]{\comp{\rm B00L}}
5827
5828 % sequences
5829 \symdecl{seqtype}[args=1]
5830 \notation{seqtype}[kleene]{#1^{\comp\ast}}
5831
5832 \symdef{sequence-index}[args=2,li,prec=nobrackets]{#{#1}_{#2}}
5833 \notation{sequence-index}[ui,prec=nobrackets]{#{#1}^{\comp\ast}}
5834
5835 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{,\ellipses}}{##1\comp,##2}
5836 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{,\ellipses,}#2}{##1\comp,##2}
5837 \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{,\ellipses,}#2\comp{,\ellipses,}#3}
5838
5839 % letin (‘‘let’’, local definitions, variable substitution)
5840 \symdecl{letin}[args=bii]
5841 \notation{letin}[let]{\comp{\rm let}}\;#1\comp{=}#2\; \comp{\rm in}}\;#3}
5842 \notation{letin}[subst]{#3 \comp[ #1 \comp/ #2 \comp]}
5843 \notation{letin}[frac]{#3 \comp[ \frac{#2}{#1} \comp]}
5844
5845 % structures
5846 \symdecl*{module-type}[args=1]
5847 \notation{module-type}{\mathtt{MOD} #1}
5848 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
5849 \notation{mathstruct}[angle,prec=nobrackets]{\comp\angle #1 \comp\rangle}{##1 \comp, ##2}
5850
5851 }
5852 \ExplSyntaxOn
5853 \stex_add_to_current_module:n{

```

```

5854 \let\nappa\apply
5855 \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
5856 \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
5857 \def\livar{\csname sequence-index\endcsname[li]}
5858 \def\uivar{\csname sequence-index\endcsname[ui]}
5859 \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
5860 \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
5861 \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
5862 }
5863 \__stex_modules_end_module:
5864 \endgroup
5865 \</package>

```

## Chapter 35

# Tikzinput Implementation

```
5866 <*package>
5867
5868 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
5869
5870 \ProvidesExplPackage{tikzinput}{2022/02/26}{3.0.1}{tikzinput package}
5871 \RequirePackage{l3keys2e}
5872
5873 \keys_define:nn { tikzinput } {
5874   image .bool_set:N = \c_tikzinput_image_bool,
5875   image .default:n = false ,
5876   unknown .code:n = {}
5877 }
5878
5879 \ProcessKeysOptions { tikzinput }
5880
5881 \bool_if:NTF \c_tikzinput_image_bool {
5882   \RequirePackage{graphicx}
5883
5884   \providecommand\usetikzlibrary[]{}
5885   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
5886 }{
5887   \RequirePackage{tikz}
5888   \RequirePackage{standalone}
5889
5890   \newcommand \tikzinput [2] [] {
5891     \setkeys{Gin}{#1}
5892     \ifx \Gin@ewidth \Gin@exclamation
5893       \ifx \Gin@eheight \Gin@exclamation
5894         \input { #2 }
5895       \else
5896         \resizebox{!}{ \Gin@eheight }{
5897           \input { #2 }
5898         }
5899       \fi
5900     \else
5901       \ifx \Gin@eheight \Gin@exclamation
5902         \resizebox{ \Gin@ewidth }{!}{
5903           \input { #2 }
```

```

5904     }
5905     \else
5906         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
5907             \input { #2 }
5908         }
5909     \fi
5910 \fi
5911 }
5912 }
5913
5914 \newcommand \ctikzinput [2] [] {
5915     \begin{center}
5916         \tikzinput [1] {#2}
5917     \end{center}
5918 }
5919
5920 \@ifpackageloaded{stex}{
5921     \RequirePackage{stex-tikzinput}
5922 }{}
5923
5924 </package>
5925 <*stex>
5926 \ProvidesExplPackage{stex-tikzinput}{2022/02/26}{3.0.1}{stex-tikzinput}
5927 \RequirePackage{stex}
5928 \RequirePackage{tikzinput}
5929
5930 \newcommand\mhtikzinput [2] [] {%
5931     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
5932     \stex_in_repository:nn\Gin@mhrepos{
5933         \tikzinput [1]{\mhpath{##1}{#2}}
5934     }
5935 }
5936 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [1] {#2}\end{center}}
5937 </stex>

```

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

## Chapter 36

# document-structure.sty Implementation

### 36.1 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.

```
5938 \*cls)
5939 \@@=document_structure)
5940 \ProvidesExplClass{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure Class}
5941 \RequirePackage{13keys2e}
```

### 36.2 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`

```
5942 \keys_define:nn{ document-structure / pkg }{
5943   class      .str_set_x:N = \c_document_structure_class_str,
5944   minimal    .bool_set:N = \c_document_structure_minimal_bool,
5945   report     .code:n      = {
5946     \ClassWarning{document-structure}{the option 'report' is deprecated, use 'class=report',
5947     \str_set:Nn \c_document_structure_class_str {report}
5948   },
5949   book       .code:n      = {
5950     \ClassWarning{document-structure}{the option 'book' is deprecated, use 'class=book', ins
5951     \str_set:Nn \c_document_structure_class_str {book}
5952   },
5953   bookpart   .code:n      = {
5954     \ClassWarning{document-structure}{the option 'bookpart' is deprecated, use 'class=book,t
5955     \str_set:Nn \c_document_structure_class_str {book}
5956     \str_set:Nn \c_document_structure_topsect_str {chapter}
5957   },
```

```

5958 docopt      .str_set_x:N = \c_document_structure_docopt_str,
5959 unknown     .code:n      = {
5960   \PassOptionsToPackage{ \CurrentOption }{ document-structure }
5961 }
5962 }
5963 \ProcessKeysOptions{ document-structure / pkg }
5964 \str_if_empty:NT \c_document_structure_class_str {
5965   \str_set:Nn \c_document_structure_class_str {article}
5966 }
5967 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
5968   {\c_document_structure_class_str}
5969

```

### 36.3 Beefing up the document environment

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

```

5970 \RequirePackage{document-structure}
5971 \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>X<sub>M</sub>L.<sup>15</sup>

```

5972 \keys_define:nn { document-structure / document }{
5973   id .str_set_x:N = \c_document_structure_document_id_str
5974 }
5975 \let\__document_structure_orig_document=\document
5976 \renewcommand{\document}[1][]{
5977   \keys_set:nn{ document-structure / document }{ #1 }
5978   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
5979   \__document_structure_orig_document
5980 }

```

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

```

5981 }
5982 \</cls>

```

### 36.4 Implementation: document-structure Package

```

5983 <*package>
5984 \ProvidesExplPackage{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure}
5985 \RequirePackage{l3keys2e}

```

### 36.5 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>15</sup>EDNOTE: faking documentkeys for now. @HANG, please implement

```

5986
5987 \keys_define:nn{ document-structure / pkg }{
5988   class      .str_set_x:N = \c_document_structure_class_str,
5989   topsect    .str_set_x:N = \c_document_structure_topsect_str,
5990   % showignores .bool_set:N = \c_document_structure_showignores_bool,
5991 }
5992 \ProcessKeysOptions{ document-structure / pkg }
5993 \str_if_empty:NT \c_document_structure_class_str {
5994   \str_set:Nn \c_document_structure_class_str {article}
5995 }
5996 \str_if_empty:NT \c_document_structure_topsect_str {
5997   \str_set:Nn \c_document_structure_topsect_str {section}
5998 }

```

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

```

5999 \RequirePackage{xspace}
6000 \RequirePackage{comment}
6001 \AddToHook{begindocument}{
6002   \ltx@ifpackageloaded{babel}{
6003     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6004     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6005       \makeatletter\input{document-structure-ngerman.ldf}\makeatother
6006     }
6007   }{}
6008 }

```

`\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.

```

6009 \int_new:N \l_document_structure_section_level_int
6010 \str_case:VnF \c_document_structure_topsect_str {
6011   {part}}{
6012     \int_set:Nn \l_document_structure_section_level_int {0}
6013   }
6014   {chapter}}{
6015     \int_set:Nn \l_document_structure_section_level_int {1}
6016   }
6017 }{
6018   \str_case:VnF \c_document_structure_class_str {
6019     {book}}{
6020       \int_set:Nn \l_document_structure_section_level_int {0}
6021     }
6022     {report}}{
6023       \int_set:Nn \l_document_structure_section_level_int {0}
6024     }
6025   }{
6026     \int_set:Nn \l_document_structure_section_level_int {2}
6027   }
6028 }

```

## 36.6 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>16</sup>

EdN:16

```
6029 \def\current@section@level{document}%
6030 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
6031 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
6032 \cs_new_protected:Npn \skipomgroup {
6033   \ifcase\l_document_structure_section_level_int
6034   \or\stepcounter{part}
6035   \or\stepcounter{chapter}
6036   \or\stepcounter{section}
6037   \or\stepcounter{subsection}
6038   \or\stepcounter{subsubsection}
6039   \or\stepcounter{paragraph}
6040   \or\stepcounter{subparagraph}
6041   \fi
6042 }
```

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

`blindfragment`

```
6043 \newcommand\at@begin@blindomgroup[1]{%
6044 \newenvironment{blindfragment}
6045 {
6046   \int_incr:N\l_document_structure_section_level_int
6047   \at@begin@blindomgroup\l_document_structure_section_level_int
6048 }{}}
```

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

```
6049 \newcommand\omgroup@nonum[2]{
6050   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
6051   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
6052 }
```

*(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.

```
6053 \newcommand\omgroup@num[2]{
```

<sup>16</sup>EDNOTE: MK: we may have to experiment with the more powerful uppercasing macro from `mfirstuc.sty` once we internationalize.



```

6054 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
6055   \@nameuse{#1}{#2}
6056 }{
6057   \cs_if_exist:NTF\rdfmata@sectioning{
6058     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
6059   }{
6060     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
6061   }
6062 }
6063 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
6064 }

```

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

**sfragment**

```

6065 \keys_define:nn { document-structure / omgroup }{
6066   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
6067   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
6068   creators     .clist_set:N = \l__document_structure_omgroup_creators_clist,
6069   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
6070   srccite      .tl_set:N   = \l__document_structure_omgroup_srccite_tl,
6071   type         .tl_set:N   = \l__document_structure_omgroup_type_tl,
6072   short        .tl_set:N   = \l__document_structure_omgroup_short_tl,
6073   display      .tl_set:N   = \l__document_structure_omgroup_display_tl,
6074   intro        .tl_set:N   = \l__document_structure_omgroup_intro_tl,
6075   loadmodules  .bool_set:N = \l__document_structure_omgroup_loadmodules_bool
6076 }
6077 \cs_new_protected:Nn \__document_structure_omgroup_args:n {
6078   \str_clear:N \l__document_structure_omgroup_id_str
6079   \str_clear:N \l__document_structure_omgroup_date_str
6080   \clist_clear:N \l__document_structure_omgroup_creators_clist
6081   \clist_clear:N \l__document_structure_omgroup_contributors_clist
6082   \tl_clear:N \l__document_structure_omgroup_srccite_tl
6083   \tl_clear:N \l__document_structure_omgroup_type_tl
6084   \tl_clear:N \l__document_structure_omgroup_short_tl
6085   \tl_clear:N \l__document_structure_omgroup_display_tl
6086   \tl_clear:N \l__document_structure_omgroup_intro_tl
6087   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
6088   \keys_set:nn { document-structure / omgroup } { #1 }
6089 }

```

we define a switch for numbering lines and a hook for the beginning of groups: The \at@begin@omgroup macro allows customization. It is run at the beginning of the omgroup, i.e. after the section heading.

```

6090 \newif\if@mainmatter\@mainmattertrue
6091 \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.

```

6092 \keys_define:nn { document-structure / sectioning }{
6093   name .str_set_x:N = \l__document_structure_sect_name_str ,
6094   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
6095   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
6096   clear .default:n = {true} ,
6097   num .bool_set:N = \l__document_structure_sect_num_bool ,

```

```

6098   num      .default:n      = {true}
6099 }
6100 \cs_new_protected:Nn \__document_structure_sect_args:n {
6101   \str_clear:N \l__document_structure_sect_name_str
6102   \str_clear:N \l__document_structure_sect_ref_str
6103   \bool_set_false:N \l__document_structure_sect_clear_bool
6104   \bool_set_false:N \l__document_structure_sect_num_bool
6105   \keys_set:nn { document-structure / sectioning } { #1 }
6106 }
6107 \newcommand\omdoc@sectioning[3][]{
6108   \__document_structure_sect_args:n {#1}
6109   \let\omdoc@sect@name\l__document_structure_sect_name_str
6110   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
6111   \if@mainmatter% numbering not overridden by frontmatter, etc.
6112     \bool_if:NTF \l__document_structure_sect_num_bool {
6113       \omgroup@num{#2}{#3}
6114     }{
6115       \omgroup@nonum{#2}{#3}
6116     }
6117     \def\current@section@level{\omdoc@sect@name}
6118   \else
6119     \omgroup@nonum{#2}{#3}
6120   \fi
6121 }% 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.

```

6122 \newcommand\omgroup@redefine@addtocontents[1]{%
6123 %\edef\__document_structureimport{#1}%
6124 %\@for\@I:=\__document_structureimport\do{%
6125 %\edef\@path{\csname module@\@I @path\endcsname}%
6126 %\@ifundefined{tf@toc}\relax%
6127 %   {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
6128 %\ifx\hyper@anchor\undefined% hyperref.sty loaded?
6129 %\def\addcontentsline##1##2##3{%
6130 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6131 %\else% hyperref.sty not loaded
6132 %\def\addcontentsline##1##2##3{%
6133 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6134 %\fi
6135 }% 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.

```

6136 \newenvironment{sfragment}[2][]{% keys, title
6137 {
6138   \__document_structure_omgroup_args:n { #1 }%\sref@target%

```

If the `loadmodules` key is set on `\begin{sfragment}`, we redefine the `\addcontetsline` macro that determines how the sectioning commands below construct the entries for the table of contents.

```

6139   \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
6140     \omgroup@redefine@addtocontents{
6141       \@ifundefined{module@id}\used@modules%

```

```

6142     %{\@ifundefined{module@}\module@id @path}{\used@modules}\module@id}
6143   }
6144 }

```

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

```

6145 \int_incr:N\l_document_structure_section_level_int
6146 \ifcase\l_document_structure_section_level_int
6147   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
6148   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
6149   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
6150   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
6151   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
6152   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
6153   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
6154 \fi
6155 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
6156 \str_if_empty:NF \l__document_structure_omgroup_id_str {
6157   \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
6158 }
6159 }% for customization
6160 {}

```

and finally, we localize the sections

```

6161 \newcommand\omdoc@part@kw{Part}
6162 \newcommand\omdoc@chapter@kw{Chapter}
6163 \newcommand\omdoc@section@kw{Section}
6164 \newcommand\omdoc@subsection@kw{Subsection}
6165 \newcommand\omdoc@subsubsection@kw{Subsubsection}
6166 \newcommand\omdoc@paragraph@kw{paragraph}
6167 \newcommand\omdoc@subparagraph@kw{subparagraph}

```

## 36.7 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`

```

6168 \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).

```

6169 \cs_if_exist:NTF\frontmatter{
6170   \let\__document_structure_orig_frontmatter\frontmatter
6171   \let\frontmatter\relax
6172 }{
6173   \tl_set:Nn\__document_structure_orig_frontmatter{
6174     \clearpage
6175     \@mainmatterfalse
6176     \pagenumbering{roman}

```

```

6177 }
6178 }
6179 \cs_if_exist:NTF\backmatter{
6180   \let\__document_structure_orig_backmatter\backmatter
6181   \let\backmatter\relax
6182 }{
6183   \tl_set:Nn\__document_structure_orig_backmatter{
6184     \clearpage
6185     \@mainmatterfalse
6186     \pagenumbering{roman}
6187   }
6188 }

```

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.

```

6189 \newenvironment{frontmatter}{
6190   \__document_structure_orig_frontmatter
6191 }{
6192   \cs_if_exist:NTF\mainmatter{
6193     \mainmatter
6194   }{
6195     \clearpage
6196     \@mainmattertrue
6197     \pagenumbering{arabic}
6198   }
6199 }

```

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

```

6200 \newenvironment{backmatter}{
6201   \__document_structure_orig_backmatter
6202 }{
6203   \cs_if_exist:NTF\mainmatter{
6204     \mainmatter
6205   }{
6206     \clearpage
6207     \@mainmattertrue
6208     \pagenumbering{arabic}
6209   }
6210 }

```

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

```

6211 \@mainmattertrue\pagenumbering{arabic}

```

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

```

6212 \def \c__document_structure_document_str{document}
6213 \newcommand\afterprematurestop{}
6214 \def\prematurestop@endomgroup{
6215   \unless\ifx\@currenvir\c__document_structure_document_str
6216     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter\expandafter
6217     \expandafter\prematurestop@endomgroup

```

```

6218 \fi
6219 }
6220 \providecommand\prematurestop{
6221 \message{Stopping~sTeX~processing~prematurely}
6222 \prematurestop@endomgroup
6223 \afterprematurestop
6224 \end{document}
6225 }

```

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

## 36.8 Global Variables

**\setSGvar** set a global variable

```

6226 \RequirePackage{etoolbox}
6227 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

**\useSGvar** use a global variable

```

6228 \newrobustcmd\useSGvar[1]{%
6229 \@ifundefined{sTeX@Gvar@#1}
6230 {\PackageError{document-structure}
6231 {The sTeX Global variable #1 is undefined}
6232 {set it with \protect\setSGvar}}
6233 \@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.

```

6234 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
6235 \@ifundefined{sTeX@Gvar@#1}
6236 {\PackageError{document-structure}
6237 {The sTeX Global variable #1 is undefined}
6238 {set it with \protect\setSGvar}}
6239 {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}

```

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

## Chapter 37

# NotesSlides – Implementation

### 37.1 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.

```
6240 \*cls)
6241 \@@=notesslides)
6242 \ProvidesExplClass{notesslides}{2022/02/28}{3.1.0}{notesslides Class}
6243 \RequirePackage{13keys2e}
6244
6245 \keys_define:nn{notesslides / cls}{
6246   class .code:n = {
6247     \PassOptionsToClass{\CurrentOption}{document-structure}
6248     \str_if_eq:nnT{#1}{book}{
6249       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
6250     }
6251     \str_if_eq:nnT{#1}{report}{
6252       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
6253     }
6254   },
6255   notes .bool_set:N = \c__notesslides_notes_bool ,
6256   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
6257   unknown .code:n = {
6258     \PassOptionsToClass{\CurrentOption}{document-structure}
6259     \PassOptionsToClass{\CurrentOption}{beamer}
6260     \PassOptionsToPackage{\CurrentOption}{notesslides}
6261   }
6262 }
6263 \ProcessKeysOptions{ notesslides / cls }
6264 \bool_if:NTF \c__notesslides_notes_bool {
6265   \PassOptionsToPackage{notes=true}{notesslides}
6266 }{
6267   \PassOptionsToPackage{notes=false}{notesslides}
6268 }
6269 \</cls)
```

now we do the same for the notesslides package.

```

6270 <*package>
6271 \ProvidesExplPackage{notesslides}{2022/02/28}{3.1.0}{notesslides Package}
6272 \RequirePackage{13keys2e}
6273
6274 \keys_define:nn{notesslides / pkg}{
6275   topsect          .str_set_x:N = \c__notesslides_topsect_str,
6276   defaulttopsect   .str_set_x:N = \c__notesslides_defaulttopsec_str,
6277   notes            .bool_set:N = \c__notesslides_notes_bool ,
6278   slides           .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
6279   sectocframes     .bool_set:N = \c__notesslides_sectocframes_bool ,
6280   frameimages      .bool_set:N = \c__notesslides_frameimages_bool ,
6281   fiboxed          .bool_set:N = \c__notesslides_fiboxed_bool ,
6282   noproblems       .bool_set:N = \c__notesslides_noproblems_bool,
6283   unknown          .code:n      = {
6284     \PassOptionsToClass{\CurrentOption}{stex}
6285     \PassOptionsToClass{\CurrentOption}{tikzinput}
6286   }
6287 }
6288 \ProcessKeysOptions{ notesslides / pkg }
6289 \newif\ifnotes
6290 \bool_if:NTF \c__notesslides_notes_bool {
6291   \notesttrue
6292 }{
6293   \notesfalse
6294 }
6295

```

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

```

6296 \str_if_empty:NTF \c__notesslides_topsect_str {
6297   \str_set_eq:NN \__notesslides_topsect \c__notesslides_defaulttopsec_str
6298 }{
6299   \str_set_eq:NN \__notesslides_topsect \c__notesslides_topsect_str
6300 }
6301 </package>

```

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

```

6302 <*cls>
6303 \bool_if:NTF \c__notesslides_notes_bool {
6304   \LoadClass{document-structure}
6305 }{
6306   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
6307   \newcounter{Item}
6308   \newcounter{paragraph}
6309   \newcounter{subparagraph}
6310   \newcounter{Hfootnote}
6311   \RequirePackage{document-structure}
6312 }

```

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

```

6313 \RequirePackage{notesslides}
6314 </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.

```

6315 \*package>
6316 \bool_if:NT \c__notesslides_notes_bool {
6317   \RequirePackage{a4wide}
6318   \RequirePackage{marginnote}
6319   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
6320   \RequirePackage{mdframed}
6321   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
6322   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
6323 }
6324 \RequirePackage{stex-tikzinput}
6325 \RequirePackage{etoolbox}
6326 \RequirePackage{amssymb}
6327 \RequirePackage{amsmath}
6328 \RequirePackage{comment}
6329 \RequirePackage{textcomp}
6330 \RequirePackage{url}
6331 \RequirePackage{graphicx}
6332 \RequirePackage{pgf}

```

## 37.2 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>17</sup>

```

6333 \bool_if:NT \c__notesslides_notes_bool {
6334   \renewcommand\usetheme[2][\usepackage[#1]{beamernotestheme#2}]
6335 }
6336
6337
6338 \NewDocumentCommand \libusetheme {0{} m} {
6339   \bool_if:NTF \c__notesslides_notes_bool {
6340     \libusepackage[#1]{beamernotestheme#2}
6341   }{
6342     \libusepackage[#1]{beamertheme#2}
6343   }
6344 }

```

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

```

6345 \newcounter{slide}
6346 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
6347 \newlength{\slideheight}\setlength{\slideheight}{9cm}

```

---

<sup>17</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.



**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.

```

6348 \bool_if:NTF \c__notesslides_notes_bool {
6349   \renewenvironment{note}{\ignorespaces}{}
6350 }{
6351   \excludecomment{note}
6352 }

```

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.

```

6353 \bool_if:NT \c__notesslides_notes_bool {
6354   \newlength{\slideframewidth}
6355   \setlength{\slideframewidth}{1.5pt}

```

**frame** We first define the keys.

```

6356 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
6357   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
6358     \bool_set_true:N #1
6359   }{
6360     \bool_set_false:N #1
6361   }
6362 }
6363 \keys_define:nn{notesslides / frame}{
6364   label .str_set_x:N = \l__notesslides_frame_label_str,
6365   allowframebreaks .code:n = {
6366     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
6367   },
6368   allowdisplaybreaks .code:n = {
6369     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
6370   },
6371   fragile .code:n = {
6372     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
6373   },
6374   shrink .code:n = {
6375     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
6376   },
6377   squeeze .code:n = {
6378     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
6379   },
6380   t .code:n = {
6381     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
6382   },
6383 }
6384 \cs_new_protected:Nn \__notesslides_frame_args:n {
6385   \str_clear:N \l__notesslides_frame_label_str
6386   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
6387   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
6388   \bool_set_true:N \l__notesslides_frame_fragile_bool
6389   \bool_set_true:N \l__notesslides_frame_shrink_bool
6390   \bool_set_true:N \l__notesslides_frame_squeeze_bool
6391   \bool_set_true:N \l__notesslides_frame_t_bool

```

```

6392 \keys_set:nn { notesslides / frame }{ #1 }
6393 }

```

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

```

6394 \renewenvironment{frame}[1][]{
6395   \__notesslides_frame_args:n{#1}
6396   \sffamily
6397   \stepcounter{slide}
6398   \def\@currentlabel{\theslide}
6399   \str_if_empty:NF \l__notesslides_frame_label_str {
6400     \label{\l__notesslides_frame_label_str}
6401   }

```

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

```

6402 \def\itemize@level{outer}
6403 \def\itemize@outer{outer}
6404 \def\itemize@inner{inner}
6405 \renewcommand\newpage{\addtocounter{framenumber}{1}}
6406 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
6407 \renewenvironment{itemize}{
6408   \ifx\itemize@level\itemize@outer
6409     \def\itemize@label{$\rhd$}
6410   \fi
6411   \ifx\itemize@level\itemize@inner
6412     \def\itemize@label{$\scriptstyle\rhd$}
6413   \fi
6414   \begin{list}
6415     {\itemize@label}
6416     {\setlength{\labelsep}{.3em}
6417      \setlength{\labelwidth}{.5em}
6418      \setlength{\leftmargin}{1.5em}
6419     }
6420   \edef\itemize@level{\itemize@inner}
6421 }{
6422   \end{list}
6423 }

```

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

```

6424 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth]
6425 }{
6426   \medskip\miko@slidelabel\end{mdframed}
6427 }

```

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

`\frametitle`

```

6428 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
6429 }

```

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

EdN:18

`\pause`

```

18
6430 \bool_if:NT \c__notesslides_notes_bool {
6431   \newcommand\pause{}
6432 }

```

---

<sup>18</sup>EdNOTE: MK: fake it in notes mode for now

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

**nparagraph**

```
6433 \bool_if:NTF \c__notesslides_notes_bool {
6434   \newenvironment{nparagraph}[1] [] {\begin{sparagraph}[#1]}\end{sparagraph}}
6435 }{
6436   \excludecomment{nparagraph}
6437 }
```

**nfragment**

```
6438 \bool_if:NTF \c__notesslides_notes_bool {
6439   \newenvironment{nfragment}[2] [] {\begin{sfragment}[#1]{#2}}\end{sfragment}}
6440 }{
6441   \excludecomment{nfragment}
6442 }
```

**ndefinition**

```
6443 \bool_if:NTF \c__notesslides_notes_bool {
6444   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}\end{sdefinition}}
6445 }{
6446   \excludecomment{ndefinition}
6447 }
```

**nassertion**

```
6448 \bool_if:NTF \c__notesslides_notes_bool {
6449   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}\end{sassertion}}
6450 }{
6451   \excludecomment{nassertion}
6452 }
```

**nsproof**

```
6453 \bool_if:NTF \c__notesslides_notes_bool {
6454   \newenvironment{nsproof}[2] [] {\begin{sproof}[#1]{#2}}\end{sproof}}
6455 }{
6456   \excludecomment{nsproof}
6457 }
```

**nexample**

```
6458 \bool_if:NTF \c__notesslides_notes_bool {
6459   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}\end{sexample}}
6460 }{
6461   \excludecomment{nexample}
6462 }
```

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

```
6463 \def\inputref@preskip{\smallskip}
6464 \def\inputref@postskip{\medskip}
```

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

`\inputref*`

```
6465 \let\orig@inputref\inputref
6466 \def\inputref{\@ifstar\ninputref\orig@inputref}
6467 \newcommand\ninputref[2][]{
6468   \bool_if:NT \c__notesslides_notes_bool {
6469     \orig@inputref[#1]{#2}
6470   }
6471 }
```

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

### 37.3 Header and Footer Lines

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

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

```
6472 \newlength{\slidelogoheight}
6473
6474 \bool_if:NTF \c__notesslides_notes_bool {
6475   \setlength{\slidelogoheight}{.4cm}
6476 }{
6477   \setlength{\slidelogoheight}{1cm}
6478 }
6479 \newsavebox{\slidelogo}
6480 \sbox{\slidelogo}{\sTeX}
6481 \newrobustcmd{\setslidelogo}[1]{
6482   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
6483 }
```

(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.

```
6484 \def\source{Michael Kohlhase}% customize locally
6485 \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.

```
6486 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
6487 \newsavebox{\cclogo}
6488 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
6489 \newif\ifcchref\cchreffalse
6490 \AtBeginDocument{
6491   \@ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
6492 }
6493 \def\licensing{
6494   \ifcchref
```

```

6495     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
6496   \else
6497     {\usebox{\cclogo}}
6498   \fi
6499 }
6500 \newrobustcmd{\setlicensing}[2][]{
6501   \def\@url{#1}
6502   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
6503   \ifx\@url\@empty
6504     \def\licensing{\usebox{\cclogo}}
6505   \else
6506     \def\licensing{
6507       \ifcchref
6508         \href{#1}{\usebox{\cclogo}}
6509       \else
6510         {\usebox{\cclogo}}
6511       \fi
6512     }
6513   \fi
6514 }

```

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

EdN:19

\slidelabel Now, we set up the slide label for the article mode.<sup>19</sup>

```

6515 \newrobustcmd\miko@slidelabel{
6516   \vbox to \slidelogoheight{
6517     \vss\hbox to \slidewidth
6518     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
6519   }
6520 }

```

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

## 37.4 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.

```

6521 \def\Gin@mhrepos{}
6522 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
6523 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
6524 \newrobustcmd\frameimage[2][]{
6525   \stepcounter{slide}
6526   \bool_if:NT \c__notesslides_frameimages_bool {
6527     \def\Gin@ewidth{}\setkeys{Gin}{#1}
6528     \bool_if:NF \c__notesslides_notes_bool { \vfill }
6529     \begin{center}
6530       \bool_if:NTF \c__notesslides_fboxed_bool {
6531         \fbox{
6532           \ifx\Gin@ewidth\@empty
6533             \ifx\Gin@mhrepos\@empty
6534               \mhgraphics[width=\slidewidth,#1]{#2}
6535             \else

```

---

<sup>19</sup>EdNOTE: see that we can use the themes for the slides some day. This is all fake.

```

6536         \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
6537     \fi
6538 \else% Gin@ewidth empty
6539     \ifx\Gin@mhrepos\@empty
6540         \mhgraphics[#1]{#2}
6541     \else
6542         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
6543     \fi
6544 \fi% Gin@ewidth empty
6545 }
6546 }{
6547     \ifx\Gin@ewidth\@empty
6548         \ifx\Gin@mhrepos\@empty
6549             \mhgraphics[width=\slidewidth,#1]{#2}
6550         \else
6551             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
6552         \fi
6553         \ifx\Gin@mhrepos\@empty
6554             \mhgraphics[#1]{#2}
6555         \else
6556             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
6557         \fi
6558     \fi% Gin@ewidth empty
6559 }
6560 \end{center}
6561 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
6562 \bool_if:NF \c__notesslides_notes_bool { \vfill }
6563 }
6564 } % ifmks@sty@frameimages

```

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

## 37.5 Colors and Highlighting

We first specify sans serif fonts as the default.

```

6565 \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

```

6566 \AddToHook{begindocument}{
6567     \definecolor{green}{rgb}{0,.5,0}
6568     \definecolor{purple}{cmyk}{.3,1,0,.17}
6569 }

```

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`.

```

6570 % \def\STpresent#1{\textcolor{blue}{#1}}
6571 \def\defemph#1{\textcolor{magenta}{#1}}
6572 \def\symrefemph#1{\textcolor{cyan}{#1}}
6573 \def\compemph#1{\textcolor{blue}{#1}}
6574 \def\titleemph#1{\textcolor{blue}{#1}}
6575 \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.

```

6576 \pgfdeclareimage[width=.8em]{miko@small@dbend}{stex-dangerous-bend}
6577 \def\smalltextwarning{
6578   \pgfuseimage{miko@small@dbend}
6579   \xspace
6580 }
6581 \pgfdeclareimage[width=1.2em]{miko@dbend}{stex-dangerous-bend}
6582 \newrobustcmd\textwarning{
6583   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
6584   \xspace
6585 }
6586 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{stex-dangerous-bend}
6587 \newrobustcmd\bigtextwarning{
6588   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
6589   \xspace
6590 }

```

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

```

6591 \newrobustcmd\putgraphicsat[3]{
6592   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
6593 }
6594 \newrobustcmd\putat[2]{
6595   \begin{picture}(0,0)\put(#1){#2}\end{picture}
6596 }

```

## 37.6 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`.

```

6597 \bool_if:NT \c__notesslides_sectocframes_bool {
6598   \str_if_eq:VnTF \__notesslidesstopsect{part}{
6599     \newcounter{chapter}\counterwithin*{section}{chapter}
6600   }{
6601     \str_if_eq:VnT \__notesslidesstopsect{chapter}{
6602       \newcounter{chapter}\counterwithin*{section}{chapter}
6603     }
6604   }
6605 }

```

`\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
6606 \def\part@prefix{}
6607 \@ifpackageloaded{document-structure}{}{
6608   \str_case:VnF \__notesslidesstopsect {
6609     {part}{
6610       \int_set:Nn \l_document_structure_section_level_int {0}
6611       \def\thesection{\arabic{chapter}.\arabic{section}}

```

```

6612     \def\part@prefix{\arabic{chapter}.}
6613   }
6614   {chapter}{
6615     \int_set:Nn \l_document_structure_section_level_int {1}
6616     \def\thesection{\arabic{chapter}.\arabic{section}}
6617     \def\part@prefix{\arabic{chapter}.}
6618   }
6619   }{
6620     \int_set:Nn \l_document_structure_section_level_int {2}
6621     \def\part@prefix{}
6622   }
6623 }
6624
6625 \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 chooses the L<sup>A</sup>T<sub>E</sub>X sectioning macros according to `\section@level`.

#### sfragment

```

6626 \renewenvironment{sfragment}[2][]{
6627   \_document_structure_omgroup_args:n { #1 }
6628   \int_incr:N \l_document_structure_section_level_int
6629   \bool_if:NT \c__notesslides_sectocframes_bool {
6630     \stepcounter{slide}
6631     \begin{frame}[noframenumbering]
6632     \vfill\Large\centering
6633     \red{
6634       \ifcase\l_document_structure_section_level_int\or
6635         \stepcounter{part}
6636         \def\_notesslideslabel{\omdoc@part@kw~\Roman{part}}
6637         \def\currentsectionlevel{\omdoc@part@kw}
6638       \or
6639         \stepcounter{chapter}
6640         \def\_notesslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
6641         \def\currentsectionlevel{\omdoc@chapter@kw}
6642       \or
6643         \stepcounter{section}
6644         \def\_notesslideslabel{\part@prefix\arabic{section}}
6645         \def\currentsectionlevel{\omdoc@section@kw}
6646       \or
6647         \stepcounter{subsection}
6648         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
6649         \def\currentsectionlevel{\omdoc@subsection@kw}
6650       \or
6651         \stepcounter{subsubsection}
6652         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
6653         \def\currentsectionlevel{\omdoc@subsubsection@kw}
6654       \or
6655         \stepcounter{paragraph}
6656         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
6657         \def\currentsectionlevel{\omdoc@paragraph@kw}
6658       \else
6659         \def\_notesslideslabel{}

```



```

6660         \def\currentsectionlevel{\omdoc@paragraph@kw}
6661         \fi% end ifcase
6662         \_notesslideslabel%\sref@label@id\_notesslideslabel
6663         \quad #2%
6664     }%
6665     \vfill%
6666     \end{frame}%
6667 }
6668 \str_if_empty:NF \l__document_structure_omgroup_id_str {
6669     \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
6670 }
6671 }{}
6672 }

```

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

```

6673 \def\inserttheorembodyfont{\normalfont}
6674 %\bool_if:NF \c__notesslides_notes_bool {
6675 % \defbeamertemplate{theorem begin}{miko}
6676 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
6677 % \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
6678 % \inserttheorempunctuation\inserttheorembodyfont\xspace}
6679 % \defbeamertemplate{theorem end}{miko}{}}

```

and we set it as the default one.

```

6680 % \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.

```

6681 % \expandafter\def\csname Parent2\endcsname{}
6682 %}
6683
6684 \AddToHook{begindocument}{% this does not work for some reason
6685     \setbeamertemplate{theorems}[ams style]
6686 }
6687 \bool_if:NT \c__notesslides_notes_bool {
6688     \renewenvironment{columns}[1][{}]{%
6689         \par\noindent%
6690         \begin{minipage}%
6691             \slidewidth\centering\leavevmode%
6692     }{}%
6693     \end{minipage}\par\noindent%
6694 }%
6695 \newsavebox\columnbox%
6696 \renewenvironment<>{column}[2][{}]{%
6697     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
6698 }{}%
6699     \end{minipage}\end{lrbox}\usebox\columnbox%
6700 }%
6701 }
6702 \bool_if:NTF \c__notesslides_noproblems_bool {
6703     \newenvironment{problems}{}{}
6704 }{
6705     \excludecomment{problems}
6706 }

```

## 37.7 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.

```

6707 \gdef\printexcursions{}
6708 \newcommand\excursionref[2]{% label, text
6709   \bool_if:NT \c__notesslides_notes_bool {
6710     \begin{sparagraph}[title=Excursion]
6711       #2 \sref[fallback=the appendix]{#1}.
6712     \end{sparagraph}
6713   }
6714 }
6715 \newcommand\activate@excursion[2][]{
6716   \gappto\printexcursions{\inputref[#1]{#2}}
6717 }
6718 \newcommand\excursion[4][]{% repos, label, path, text
6719   \bool_if:NT \c__notesslides_notes_bool {
6720     \activate@excursion[#1]{#3}\excursionref{#2}{#4}
6721   }
6722 }

```

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

`\excursiongroup`

```

6723 \keys_define:nn{notesslides / excursiongroup }{
6724   id          .str_set_x:N = \l__notesslides_excursion_id_str,
6725   intro       .tl_set:N   = \l__notesslides_excursion_intro_tl,
6726   mhrepos     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
6727 }
6728 \cs_new_protected:Nn \__notesslides_excursion_args:n {
6729   \tl_clear:N \l__notesslides_excursion_intro_tl
6730   \str_clear:N \l__notesslides_excursion_id_str
6731   \str_clear:N \l__notesslides_excursion_mhrepos_str
6732   \keys_set:nn {notesslides / excursiongroup }{ #1 }
6733 }
6734 \newcommand\excursiongroup[1][]{
6735   \__notesslides_excursion_args:n{ #1 }
6736   \ifdefempty\printexcursions{}% only if there are excursions
6737   {\begin{note}
6738     \begin{sfragment}[#1]{Excursions}%
6739     \ifdefempty\l__notesslides_excursion_intro_tl{
6740       \inputref[\l__notesslides_excursion_mhrepos_str]{
6741         \l__notesslides_excursion_intro_tl
6742       }
6743     }
6744     \printexcursions%
6745     \end{sfragment}
6746     \end{note}}
6747 }
6748 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{\fi
6749 \package}

```

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

## Chapter 38

# The Implementation

### 38.1 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.

```
6750 <*package>
6751 <@@=problems>
6752 \ProvidesExplPackage{problem}{2022/02/26}{3.0.1}{Semantic Markup for Problems}
6753 \RequirePackage{l3keys2e,stex}
6754
6755 \keys_define:nn { problem / pkg }{
6756   notes      .default:n    = { true },
6757   notes      .bool_set:N   = \c__problems_notes_bool,
6758   gnotes     .default:n    = { true },
6759   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
6760   hints      .default:n    = { true },
6761   hints      .bool_set:N   = \c__problems_hints_bool,
6762   solutions  .default:n    = { true },
6763   solutions  .bool_set:N   = \c__problems_solutions_bool,
6764   pts        .default:n    = { true },
6765   pts        .bool_set:N   = \c__problems_pts_bool,
6766   min        .default:n    = { true },
6767   min        .bool_set:N   = \c__problems_min_bool,
6768   boxed      .default:n    = { true },
6769   boxed      .bool_set:N   = \c__problems_boxed_bool,
6770   unknown    .code:n       = {}
6771 }
6772 \newif\ifsolutions
6773
6774 \ProcessKeysOptions{ problem / pkg }
6775 \bool_if:NTF \c__problems_solutions_bool {
6776   \solutionstrue
6777 }{
6778   \solutionsfalse
6779 }
```

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

```
6780 \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.

```
6781 \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.

```
6782 \def\prob@problem@kw{Problem}
6783 \def\prob@solution@kw{Solution}
6784 \def\prob@hint@kw{Hint}
6785 \def\prob@note@kw{Note}
6786 \def\prob@gnote@kw{Grading}
6787 \def\prob@pt@kw{pt}
6788 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
6789 \AddToHook{begindocument}{
6790   \ltx@ifpackageloaded{babel}{
6791     \makeatletter
6792     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6793     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6794       \input{problem-ngerman.ldf}
6795     }
6796     \clist_if_in:NnT \l_tmpa_clist {finnish}{
6797       \input{problem-finnish.ldf}
6798     }
6799     \clist_if_in:NnT \l_tmpa_clist {french}{
6800       \input{problem-french.ldf}
6801     }
6802     \clist_if_in:NnT \l_tmpa_clist {russian}{
6803       \input{problem-russian.ldf}
6804     }
6805     \makeatother
6806   }{ }
6807 }
```

## 38.2 Problems and Solutions

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

```
6808 \keys_define:nn{ problem / problem }{
6809   id      .str_set:x:N = \l__problems_prob_id_str,
6810   pts     .tl_set:N    = \l__problems_prob_pts_tl,
6811   min     .tl_set:N    = \l__problems_prob_min_tl,
6812   title   .tl_set:N    = \l__problems_prob_title_tl,
6813   type    .tl_set:N    = \l__problems_prob_type_tl,
6814   refnum  .int_set:N   = \l__problems_prob_refnum_int
6815 }
6816 \cs_new_protected:Nn \__problems_prob_args:n {
```

```

6817 \str_clear:N \l__problems_prob_id_str
6818 \tl_clear:N \l__problems_prob_pts_tl
6819 \tl_clear:N \l__problems_prob_min_tl
6820 \tl_clear:N \l__problems_prob_title_tl
6821 \tl_clear:N \l__problems_prob_type_tl
6822 \int_zero_new:N \l__problems_prob_refnum_int
6823 \keys_set:nn { problem / problem }{ #1 }
6824 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
6825   \let\l__problems_prob_refnum_int\undefined
6826 }
6827 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

6828 \newcounter{problem}
6829 \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.

```

6830 \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

```

6831 \newcommand\prob@number{
6832   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
6833     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
6834   }{
6835     \int_if_exist:NTF \l__problems_prob_refnum_int {
6836       \prob@label{\int_use:N \l__problems_prob_refnum_int }
6837     }{
6838       \prob@label\theproblem
6839     }
6840   }
6841 }

```

(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.

```

6842 \newcommand\prob@title[3]{%
6843   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
6844     #2 \l__problems_inclprob_title_tl #3
6845   }{
6846     \tl_if_exist:NTF \l__problems_prob_title_tl {
6847       #2 \l__problems_prob_title_tl #3
6848     }{
6849       #1
6850     }
6851   }
6852 }

```

(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.

```

6853 \def\prob@heading{
6854   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
6855   %\sref@label{id}\prob@problem@kw~\prob@number}{~}
6856 }

```

(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`

```

6857 \newenvironment{sproblem}[1][{}]{
6858   \__problems_prob_args:n{#1}%\sref@target%
6859   \@in@omtexttrue% we are in a statement (for inline definitions)
6860   \stepcounter{problem}\record@problem
6861   \def\current@section@level{\prob@problem@kw}
6862   \tl_if_exist:NTF \l__problems_inclprob_type_tl {
6863     \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
6864   }{
6865     \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
6866   }
6867   \str_if_exist:NTF \l__problems_inclprob_id_str {
6868     \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
6869   }{
6870     \str_set_eq:NN \sproblemid \l__problems_prob_id_str
6871   }
6872
6873
6874   \clist_set:No \l_tmpa_clist \sproblemtype
6875   \tl_clear:N \l_tmpa_tl
6876   \clist_map_inline:Nn \l_tmpa_clist {
6877     \tl_if_exist:cT {\__problems_sproblem_##1_start:}{
6878       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_start:}}
6879     }
6880   }
6881   \tl_if_empty:NTF \l_tmpa_tl {
6882     \__problems_sproblem_start:
6883   }{
6884     \l_tmpa_tl
6885   }
6886   \stex_ref_new_doc_target:n \sproblemid
6887 }{
6888   \clist_set:No \l_tmpa_clist \sproblemtype
6889   \tl_clear:N \l_tmpa_tl
6890   \clist_map_inline:Nn \l_tmpa_clist {
6891     \tl_if_exist:cT {\__problems_sproblem_##1_end:}{
6892       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_end:}}
6893     }

```

```

6894 }
6895 \tl_if_empty:NTF \l_tmpa_tl {
6896   \__problems_sproblem_end:
6897 }{
6898   \l_tmpa_tl
6899 }
6900
6901
6902 \smallskip
6903 }
6904
6905
6906 \cs_new_protected:Nn \__problems_sproblem_start: {
6907   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
6908 }
6909 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
6910
6911 \newcommand\stexpatchproblem[3][] {
6912   \str_set:Nx \l_tmpa_str{ #1 }
6913   \str_if_empty:NTF \l_tmpa_str {
6914     \tl_set:Nn \__problems_sproblem_start: { #2 }
6915     \tl_set:Nn \__problems_sproblem_end: { #3 }
6916   }{
6917     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
6918     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
6919   }
6920 }
6921
6922
6923 \bool_if:NT \c__problems_boxed_bool {
6924   \surroundwithmdframed{problem}
6925 }

```

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

```

6926 \def\record@problem{
6927   \protected@write\@auxout{}
6928   {
6929     \string\@problem{\prob@number}
6930     {
6931       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
6932         \l__problems_inclprob_pts_tl
6933       }{
6934         \l__problems_prob_pts_tl
6935       }
6936     }%
6937     {
6938       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
6939         \l__problems_inclprob_min_tl
6940       }{
6941         \l__problems_prob_min_tl
6942       }
6943     }
6944   }
6945 }

```

(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).

```
6946 \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.

```
6947 \keys_define:nn { problem / solution }{
6948   id                .str_set_x:N = \l__problems_solution_id_str ,
6949   for               .tl_set:N    = \l__problems_solution_for_tl ,
6950   height            .dim_set:N   = \l__problems_solution_height_dim ,
6951   creators          .clist_set:N = \l__problems_solution_creators_clist ,
6952   contributors      .clist_set:N = \l__problems_solution_contributors_clist ,
6953   srccite           .tl_set:N    = \l__problems_solution_srccite_tl
6954 }
6955 \cs_new_protected:Nn \__problems_solution_args:n {
6956   \str_clear:N \l__problems_solution_id_str
6957   \tl_clear:N \l__problems_solution_for_tl
6958   \tl_clear:N \l__problems_solution_srccite_tl
6959   \clist_clear:N \l__problems_solution_creators_clist
6960   \clist_clear:N \l__problems_solution_contributors_clist
6961   \dim_zero:N \l__problems_solution_height_dim
6962   \keys_set:nn { problem / solution }{ #1 }
6963 }
```

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

```
6964 \newcommand\@startsolution[1][ ]{
6965   \__problems_solution_args:n { #1 }
6966   \@in@omtexttrue% we are in a statement.
6967   \bool_if:NF \c__problems_boxed_bool { \hrule }
6968   \smallskip\noindent
6969   {\textbf\prob@solution@kw : \enspace}
6970   \begin{small}
6971   \def\current@section@level{\prob@solution@kw}
6972   \ignorespacesandpars
6973 }
```

**\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.

```
6974 \newcommand\startsolutions{
6975   \specialcomment{solution}{\@startsolution}{
6976     \bool_if:NF \c__problems_boxed_bool {
6977       \hrule\medskip
6978     }
6979     \end{small}%
6980   }
6981   \bool_if:NT \c__problems_boxed_bool {
6982     \surroundwithmdframed{solution}
6983   }
6984 }
```



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

\stopsolutions

```
6985 \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.

```
6986 \ifsolutions
6987   \startsolutions
6988 \else
6989   \stopsolutions
6990 \fi
```

exnote

```
6991 \bool_if:NTF \c__problems_notes_bool {
6992   \newenvironment{exnote}[1][]{
6993     \par\smallskip\hrule\smallskip
6994     \noindent\textbf{\prob@note@kw : }\small
6995   }{
6996     \smallskip\hrule
6997   }
6998 }{
6999   \excludecomment{exnote}
7000 }
```

hint

```
7001 \bool_if:NTF \c__problems_notes_bool {
7002   \newenvironment{hint}[1][]{
7003     \par\smallskip\hrule\smallskip
7004     \noindent\textbf{\prob@hint@kw :~ }\small
7005   }{
7006     \smallskip\hrule
7007   }
7008   \newenvironment{exhint}[1][]{
7009     \par\smallskip\hrule\smallskip
7010     \noindent\textbf{\prob@hint@kw :~ }\small
7011   }{
7012     \smallskip\hrule
7013   }
7014 }{
7015   \excludecomment{hint}
7016   \excludecomment{exhint}
7017 }
```

gnote

```
7018 \bool_if:NTF \c__problems_notes_bool {
7019   \newenvironment{gnote}[1][]{
7020     \par\smallskip\hrule\smallskip
7021     \noindent\textbf{\prob@gnote@kw : }\small
7022   }{
7023     \smallskip\hrule
7024   }
7025 }{
7026   \excludecomment{gnote}
7027 }
```

### 38.3 Multiple Choice Blocks

```

7028 \newenvironment{mcb}{
7029   \begin{enumerate}
7030 }{
7031   \end{enumerate}
7032 }

```

we define the keys for the mcc macro

```

7033 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
7034   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
7035     \bool_set_true:N #1
7036   }{
7037     \bool_set_false:N #1
7038   }
7039 }
7040 \keys_define:nn { problem / mcc }{
7041   id          .str_set:N = \l__problems_mcc_id_str ,
7042   feedback    .tl_set:N   = \l__problems_mcc_feedback_tl ,
7043   T           .default:n   = { true } ,
7044   T           .bool_set:N  = \l__problems_mcc_t_bool ,
7045   F           .default:n   = { true } ,
7046   F           .bool_set:N  = \l__problems_mcc_f_bool ,
7047   Ttext       .code:n      = {
7048     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
7049   } ,
7050   Ftext       .code:n      = {
7051     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
7052   }
7053 }
7054 \cs_new_protected:Nn \l__problems_mcc_args:n {
7055   \str_clear:N \l__problems_mcc_id_str
7056   \tl_clear:N \l__problems_mcc_feedback_tl
7057   \bool_set_true:N \l__problems_mcc_t_bool
7058   \bool_set_true:N \l__problems_mcc_f_bool
7059   \bool_set_true:N \l__problems_mcc_Ttext_bool
7060   \bool_set_false:N \l__problems_mcc_Ftext_bool
7061   \keys_set:nn { problem / mcc }{ #1 }
7062 }

```

\mcc

```

7063 \newcommand\mcc[2][]{
7064   \l__problems_mcc_args:n{ #1 }
7065   \item #2
7066   \ifsolutions
7067     \
7068     \bool_if:NT \l__problems_mcc_t_bool {
7069       % TODO!
7070       % \ifcsstring{mcc@T}{T}{\mcc@Ttext}%
7071     }
7072     \bool_if:NT \l__problems_mcc_f_bool {

```

---

<sup>20</sup>EdNOTE: MK: maybe import something better here from a dedicated MC package

```

7073      % TODO!
7074      % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
7075    }
7076    \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
7077      !
7078    }{
7079      \l__problems_mcc_feedback_tl
7080    }
7081    \fi
7082  } %solutions

```

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

## 38.4 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.

```

7083
7084 \keys_define:nn{ problem / inclproblem }{
7085   id      .str_set:N = \l__problems_inclprob_id_str,
7086   pts     .tl_set:N  = \l__problems_inclprob_pts_tl,
7087   min     .tl_set:N  = \l__problems_inclprob_min_tl,
7088   title   .tl_set:N  = \l__problems_inclprob_title_tl,
7089   refnum  .int_set:N  = \l__problems_inclprob_refnum_int,
7090   type    .tl_set:N  = \l__problems_inclprob_type_tl,
7091   mhrepos .str_set:N = \l__problems_inclprob_mhrepos_str
7092 }
7093 \cs_new_protected:Nn \l__problems_inclprob_args:n {
7094   \str_clear:N \l__problems_prob_id_str
7095   \tl_clear:N \l__problems_inclprob_pts_tl
7096   \tl_clear:N \l__problems_inclprob_min_tl
7097   \tl_clear:N \l__problems_inclprob_title_tl
7098   \tl_clear:N \l__problems_inclprob_type_tl
7099   \int_zero_new:N \l__problems_inclprob_refnum_int
7100   \str_clear:N \l__problems_inclprob_mhrepos_str
7101   \keys_set:nn { problem / inclproblem }{ #1 }
7102   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
7103     \let\l__problems_inclprob_pts_tl\undefined
7104   }
7105   \tl_if_empty:NT \l__problems_inclprob_min_tl {
7106     \let\l__problems_inclprob_min_tl\undefined
7107   }
7108   \tl_if_empty:NT \l__problems_inclprob_title_tl {
7109     \let\l__problems_inclprob_title_tl\undefined
7110   }
7111   \tl_if_empty:NT \l__problems_inclprob_type_tl {
7112     \let\l__problems_inclprob_type_tl\undefined
7113   }
7114   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
7115     \let\l__problems_inclprob_refnum_int\undefined
7116   }
7117 }

```

```

7118
7119 \cs_new_protected:Nn \__problems_inclprob_clear: {
7120   \let\l__problems_inclprob_id_str\undefined
7121   \let\l__problems_inclprob_pts_tl\undefined
7122   \let\l__problems_inclprob_min_tl\undefined
7123   \let\l__problems_inclprob_title_tl\undefined
7124   \let\l__problems_inclprob_type_tl\undefined
7125   \let\l__problems_inclprob_refnum_int\undefined
7126   \let\l__problems_inclprob_mhrepos_str\undefined
7127 }
7128 \__problems_inclprob_clear:
7129
7130 \newcommand\includeproblem[2][ ]{
7131   \__problems_inclprob_args:n{ #1 }
7132   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
7133     \input{#2}
7134   }{
7135     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
7136       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
7137     }
7138   }
7139   \__problems_inclprob_clear:
7140 }

```

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

## 38.5 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.

```

7141 \AddToHook{enddocument}{
7142   \bool_if:NT \c__problems_pts_bool {
7143     \message{Total:~\arabic{pts}~points}
7144   }
7145   \bool_if:NT \c__problems_min_bool {
7146     \message{Total:~\arabic{min}~minutes}
7147   }
7148 }

```

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

```

7149 \def\pts#1{
7150   \bool_if:NT \c__problems_pts_bool {
7151     \marginpar{#1~\prob@pt@kw}
7152   }
7153 }
7154 \def\min#1{
7155   \bool_if:NT \c__problems_min_bool {
7156     \marginpar{#1~\prob@min@kw}
7157   }
7158 }

```

`\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.

```

7159 \newcounter{pts}
7160 \def\show@pts{
7161   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7162     \bool_if:NT \c__problems_pts_bool {
7163       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
7164       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
7165     }
7166   }{
7167     \tl_if_exist:NT \l__problems_prob_pts_tl {
7168       \bool_if:NT \c__problems_pts_bool {
7169         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
7170         \addtocounter{pts}{\l__problems_prob_pts_tl}
7171       }
7172     }
7173   }
7174 }

```

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

`\show@min`

```

7175 \newcounter{min}
7176 \def\show@min{
7177   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7178     \bool_if:NT \c__problems_min_bool {
7179       \marginpar{\l__problems_inclprob_min_tl\ min}
7180       \addtocounter{min}{\l__problems_inclprob_min_tl}
7181     }
7182   }{
7183     \tl_if_exist:NT \l__problems_prob_min_tl {
7184       \bool_if:NT \c__problems_min_bool {
7185         \marginpar{\l__problems_prob_min_tl\ min}
7186         \addtocounter{min}{\l__problems_prob_min_tl}
7187       }
7188     }
7189   }
7190 }
7191 </package>

```

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

## Chapter 39

# 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.

### 39.1 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.

```
7192 <@@=hwexam>
7193 <*cls>
7194 \ProvidesExplClass{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
7195 \RequirePackage{l3keys2e}
7196 \DeclareOption*{
7197   \PassOptionsToClass{\CurrentOption}{document-structure}
7198   \PassOptionsToPackage{\CurrentOption}{stex}
7199   \PassOptionsToPackage{\CurrentOption}{hwexam}
7200   \PassOptionsToPackage{\CurrentOption}{tikzinput}
7201 }
7202 \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.

```
7203 \LoadClass{document-structure}
7204 \RequirePackage{stex}
7205 \RequirePackage{hwexam}
7206 \RequirePackage{tikzinput}
7207 \RequirePackage{graphicx}
7208 \RequirePackage{a4wide}
7209 \RequirePackage{amssymb}
7210 \RequirePackage{amstext}
7211 \RequirePackage{amsmath}
```

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

```

7212 \newcommand\assig@default@type{\hwexam@assignment@kw}
7213 \def\document@hwexamtype{\assig@default@type}
7214 <@@=document_structure>
7215 \keys_define:nn { document-structure / document }{
7216 id .str_set_x:N = \c_document_structure_document_id_str,
7217 hwexamtype .tl_set:N = \document@hwexamtype
7218 }
7219 <@@=hwexam>
7220 </cls>

```

## Chapter 40

# Implementation: The hwexam Package

### 40.1 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.

```
7221 \*package>
7222 \ProvidesExplPackage{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
7223 \RequirePackage{13keys2e}
7224
7225 \newif\iftest\testfalse
7226 \DeclareOption{test}{\testtrue}
7227 \newif\ifmultiple\multiplefalse
7228 \DeclareOption{multiple}{\multipletrue}
7229 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
7230 \ProcessOptions
```

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

```
7231 \RequirePackage{keyval}[1997/11/10]
7232 \RequirePackage{problem}
```

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

```
7233 \newcommand\hwexam@assignment@kw{Assignment}
7234 \newcommand\hwexam@given@kw{Given}
7235 \newcommand\hwexam@due@kw{Due}
7236 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
7237 blank~for~extra~space}
7238 \def\hwexam@minutes@kw{minutes}
7239 \newcommand\correction@probs@kw{prob.}
7240 \newcommand\correction@pts@kw{total}
7241 \newcommand\correction@reached@kw{reached}
7242 \newcommand\correction@sum@kw{Sum}
7243 \newcommand\correction@grade@kw{grade}
7244 \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

```

7245 \AddToHook{begindocument}{
7246 \ltx@ifpackageloaded{babel}{
7247 \makeatletter
7248 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7249 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7250 \input{hwexam-ngerman.ldf}
7251 }
7252 \clist_if_in:NnT \l_tmpa_clist {finnish}{
7253 \input{hwexam-finnish.ldf}
7254 }
7255 \clist_if_in:NnT \l_tmpa_clist {french}{
7256 \input{hwexam-french.ldf}
7257 }
7258 \clist_if_in:NnT \l_tmpa_clist {russian}{
7259 \input{hwexam-russian.ldf}
7260 }
7261 \makeatother
7262 }{}
7263 }
7264

```

## 40.2 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.

```

7265 \newcounter{assignment}
7266 \numberproblemsin{assignment}
7267 \renewcommand\prob@label[1]{\assignment@number.#1}

```

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

```

7268 \keys_define:nn { hwexam / assignment } {
7269 id .str_set:N = \l__hwexam_assign_id_str,
7270 number .int_set:N = \l__hwexam_assign_number_int,
7271 title .tl_set:N = \l__hwexam_assign_title_tl,
7272 type .tl_set:N = \l__hwexam_assign_type_tl,
7273 given .tl_set:N = \l__hwexam_assign_given_tl,
7274 due .tl_set:N = \l__hwexam_assign_due_tl,
7275 loadmodules .code:n = {
7276 \bool_set_true:N \l__hwexam_assign_loadmodules_bool
7277 }
7278 }
7279 \cs_new_protected:Nn \__hwexam_assignment_args:n {
7280 \str_clear:N \l__hwexam_assign_id_str
7281 \int_set:Nn \l__hwexam_assign_number_int {-1}
7282 \tl_clear:N \l__hwexam_assign_title_tl
7283 \tl_clear:N \l__hwexam_assign_type_tl
7284 \tl_clear:N \l__hwexam_assign_given_tl
7285 \tl_clear:N \l__hwexam_assign_due_tl
7286 \bool_set_false:N \l__hwexam_assign_loadmodules_bool

```

```

7287 \keys_set:nn { hwexam / assignment }{ #1 }
7288 }

```

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.

```

7289 \newcommand\given@due[2]{
7290 \bool_lazy_all:nF {
7291 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
7292 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
7293 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
7294 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
7295 }{ #1 }
7296
7297 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
7298 \tl_if_empty:NF \l__hwexam_assign_given_tl {
7299 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
7300 }
7301 }{
7302 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
7303 }
7304
7305 \bool_lazy_or:nnF {
7306 \bool_lazy_and_p:nn {
7307 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
7308 }{
7309 \tl_if_empty_p:V \l__hwexam_assign_due_tl
7310 }
7311 }{
7312 \bool_lazy_and_p:nn {
7313 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
7314 }{
7315 \tl_if_empty_p:V \l__hwexam_assign_due_tl
7316 }
7317 }{ ,~ }
7318
7319 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
7320 \tl_if_empty:NF \l__hwexam_assign_due_tl {
7321 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
7322 }
7323 }{
7324 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
7325 }
7326
7327 \bool_lazy_all:nF {
7328 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
7329 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
7330 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
7331 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
7332 }{ #2 }
7333 }

```

`\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.

```

7334 \newcommand\assignment@title[3]{
7335 \tl_if_empty:NTF \l__hwexam_inclasssign_title_tl {
7336 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
7337 #1
7338 }{
7339 #2\l__hwexam_assign_title_tl#3
7340 }
7341 }{
7342 #2\l__hwexam_inclasssign_title_tl#3
7343 }
7344 }

```

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

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

```

7345 \newcommand\assignment@number{
7346 \int_compare:nNnTF \l__hwexam_inclasssign_number_int = {-1} {
7347 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
7348 \arabic{assignment}
7349 } {
7350 \int_use:N \l__hwexam_assign_number_int
7351 }
7352 }{
7353 \int_use:N \l__hwexam_inclasssign_number_int
7354 }
7355 }

```

(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.

```

7356 \newenvironment{assignment}[1][]{
7357 \__hwexam_assignment_args:n { #1 }
7358 %\sref@target
7359 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
7360 \global\stepcounter{assignment}
7361 }{
7362 \global\setcounter{assignment}{\int_use:N\l__hwexam_assign_number_int}
7363 }
7364 \setcounter{problem}{0}
7365 \def\current@section@level{\document@hwexamtype}
7366 %\sref@label@id{\document@hwexamtype \thesection}
7367 \begin{@assignment}
7368 }{
7369 \end{@assignment}
7370 }

```

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

```

7371 \def\ass@title{
7372 \protect\document@hwexamtype~\arabic{assignment}
7373 \assignment@title{}\{;\}{} -- \given@due{}\}{}
7374 }
7375 \ifmultiple
7376 \newenvironment{@assignment}{
7377 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
7378 \begin{sfragment}[loadmodules]{\ass@title}
7379 }{
7380 \begin{sfragment}{\ass@title}
7381 }
7382 }{
7383 \end{sfragment}
7384 }

```

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

```

7385 \else
7386 \newenvironment{@assignment}{
7387 \begin{center}\bf
7388 \Large@title\strut\\
7389 \document@hwexamtype~\arabic{assignment}\assignment@title{}\{;\}{}{\}{}
7390 \large\given@due{--\}{}\{;\}{}
7391 \end{center}
7392 }{}
7393 \fi% multiple

```

## 40.3 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.

```

7394 \keys_define:nn { hwexam / inclassignment } {
7395 %id .str_set_x:N = \l__hwexam_assign_id_str,
7396 number .int_set:N = \l__hwexam_inclassign_number_int,
7397 title .tl_set:N = \l__hwexam_inclassign_title_tl,
7398 type .tl_set:N = \l__hwexam_inclassign_type_tl,
7399 given .tl_set:N = \l__hwexam_inclassign_given_tl,
7400 due .tl_set:N = \l__hwexam_inclassign_due_tl,
7401 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
7402 }
7403 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
7404 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
7405 \tl_clear:N \l__hwexam_inclassign_title_tl
7406 \tl_clear:N \l__hwexam_inclassign_type_tl
7407 \tl_clear:N \l__hwexam_inclassign_given_tl
7408 \tl_clear:N \l__hwexam_inclassign_due_tl
7409 \str_clear:N \l__hwexam_inclassign_mhrepos_str
7410 \keys_set:nn { hwexam / inclassignment }{ #1 }
7411 }
7412 \__hwexam_inclassignment_args:n {}
7413
7414 \newcommand\inputassignment[2][{}]{

```

```

7415 \_hwexam_inclassnment_args:n { #1 }
7416 \str_if_empty:NTF \l__hwexam_inclassn_mhrepos_str {
7417   \input{#2}
7418 }{
7419   \stex_in_repository:nn{\l__hwexam_inclassn_mhrepos_str}{
7420     \input{\mhp{path}\l__hwexam_inclassn_mhrepos_str}{#2}}
7421   }
7422 }
7423 \_hwexam_inclassnment_args:n {}
7424 }
7425 \newcommand\includeassignment[2][]{
7426   \newpage
7427   \inputassignment[#1]{#2}
7428 }

```

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

## 40.4 Typesetting Exams

\quizheading

```

7429 \ExplSyntaxOff
7430 \newcommand\quizheading[1]{%
7431   \def\@tas{#1}%
7432   \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
7433   \ifx\@tas\@empty\else%
7434     \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
7435   \fi%
7436 }
7437 \ExplSyntaxOn

```

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

\testheading

```

7438
7439 \def\hwexamheader{\input{hwexam-default.header}}
7440
7441 \def\hwexamminutes{
7442   \tl_if_empty:NTF \testheading@duration {
7443     {\testheading@min}~\hwexam@minutes@kw
7444   }{
7445     \testheading@duration
7446   }
7447 }
7448
7449 \keys_define:nn { hwexam / testheading } {
7450   min .tl_set:N = \testheading@min,
7451   duration .tl_set:N = \testheading@duration,
7452   reqpts .tl_set:N = \testheading@reqpts,
7453   tools .tl_set:N = \testheading@tools
7454 }
7455 \cs_new_protected:Nn \_hwexam_testheading_args:n {
7456   \tl_clear:N \testheading@min
7457   \tl_clear:N \testheading@duration

```

```

7458 \tl_clear:N \testheading@reqpts
7459 \tl_clear:N \testheading@tools
7460 \keys_set:nn { hwexam / testheading }{ #1 }
7461 }
7462 \newenvironment{testheading}[1][]{
7463   \_hwexam_testheading_args:n{ #1 }
7464   \newcount\check@time\check@time=\testheading@min
7465   \advance\check@time by -\theassignment@totalmin
7466   \newif\if@bonuspoints
7467   \tl_if_empty:NTF \testheading@reqpts {
7468     \@bonuspointsfalse
7469   }{
7470     \newcount\bonus@pts
7471     \bonus@pts=\theassignment@totalpts
7472     \advance\bonus@pts by -\testheading@reqpts
7473     \edef\bonus@pts{\the\bonus@pts}
7474     \@bonuspointstrue
7475   }
7476   \edef\check@time{\the\check@time}
7477
7478   \makeatletter\hwexamheader\makeatother
7479 }{
7480   \newpage
7481 }

```

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

\testspace

```

7482 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

7483 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

7484 \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.

```

7485 <@=problems>
7486 \renewcommand\@problem[3]{
7487   \stepcounter{assignment@probs}
7488   \def\__problemspts{#2}
7489   \ifx\__problemspts\@empty\else
7490     \addtocounter{assignment@totalpts}{#2}
7491   \fi
7492   \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
7493   \xdef\correction@probs{\correction@probs & #1}%
7494   \xdef\correction@pts{\correction@pts & #2}
7495   \xdef\correction@reached{\correction@reached &}

```

```

7496 }
7497 <@@=hwexam>

```

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

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

```

7498 \newcounter{assignment@probs}
7499 \newcounter{assignment@totalpts}
7500 \newcounter{assignment@totalmin}
7501 \def\correction@probs{\correction@probs@kw}
7502 \def\correction@pts{\correction@pts@kw}
7503 \def\correction@reached{\correction@reached@kw}
7504 \stepcounter{assignment@probs}
7505 \newcommand\correction@table{
7506 \resizebox{\textwidth}{!}{%
7507 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
7508 &\multicolumn{\theassignment@probs}{c|}|%|
7509 {\footnotesize\correction@forgrading@kw} &\\\hline
7510 \correction@probs & \correction@sum@kw & \correction@grade@kw\\\hline
7511 \correction@pts & \theassignment@totalpts & \\\hline
7512 \correction@reached & & \[.7cm]\hline
7513 \end{tabular}}
7514 </package>

```

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

## 40.5 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}

```