

# The sTeX3 Package Collection \*

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

2022-05-07

## 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.

Running ‘pdflatex’ over sTeX-annotated documents formats them into normal-looking PDF. But sTeX also comes with a conversion pipeline into semantically annotated HTML5, which can host semantic added-value services that make the documents active (i.e. interactive and user-adaptive) and 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, and
- 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 2022-05-07)

# Contents

|          |   |           |
|----------|---|-----------|
| <b>I</b> | <b>Manual</b>   | <b>1</b>  |
| <b>1</b> | <b>What is sTeX?</b>  | <b>2</b>  |
| <b>2</b> | <b>Quickstart</b>   | <b>3</b>  |
| 2.1      | Setup   | 3         |
| 2.1.1    | Minimal Setup for the PDF-only Workflow                             | 3         |
| 2.1.2    | GIT-based Setup for the sTeX Development Version                    | 3         |
| 2.1.3    | sTeX Archives (Manual Setup)  | 4         |
| 2.1.4    | The sTeX IDE  | 4         |
| 2.1.5    | Manual Setup for Active Documents and Knowledge Management Services | 4         |
| 2.2      | A First sTeX Document   | 5         |
| 2.2.1    | OMDoc/xhtml Conversion  | 8         |
| <b>3</b> | <b>Creating sTeX Content</b>  | <b>10</b> |
| 3.1      | How Knowledge is Organized in sTeX                                  | 10        |
| 3.2      | sTeX Archives   | 11        |
| 3.2.1    | The Local MathHub-Directory   | 11        |
| 3.2.2    | The Structure of sTeX Archives                                      | 11        |
| 3.2.3    | MANIFEST.MF-Files   | 12        |
| 3.2.4    | Using Files in sTeX Archives Directly                               | 13        |
| 3.3      | Module, Symbol and Notation Declarations                            | 14        |
| 3.3.1    | The smodule-Environment   | 14        |
| 3.3.2    | Declaring New Symbols and Notations                                 | 16        |
|          | Operator Notations  | 19        |
| 3.3.3    | Argument Modes  | 19        |
|          | Mode-b Arguments  | 20        |
|          | Mode-a Arguments  | 20        |
|          | Mode-B Arguments  | 22        |
| 3.3.4    | Type and Definiens Components                                       | 22        |
| 3.3.5    | Precedences and Automated Bracketing                                | 23        |
| 3.3.6    | Variables   | 25        |
| 3.3.7    | Variable Sequences  | 26        |
| 3.4      | Module Inheritance and Structures                                   | 28        |
| 3.4.1    | Multilinguality and Translations                                    | 28        |
| 3.4.2    | Simple Inheritance and Namespaces                                   | 29        |
| 3.4.3    | The mathstructure Environment                                       | 31        |
| 3.4.4    | The copymodule Environment  | 34        |
| 3.4.5    | The interpretmodule Environment                                     | 35        |
| 3.5      | Primitive Symbols (The sTeX Metatheory)                             | 35        |
| <b>4</b> | <b>Using sTeX Symbols</b>   | <b>36</b> |
| 4.1      | \symref and its variants  | 36        |
| 4.2      | Marking Up Text and On-the-Fly Notations                            | 37        |
| 4.3      | Referencing Symbols and Statements                                  | 39        |

|           |   |           |
|-----------|---|-----------|
| <b>5</b>  | <b>sTeX Statements</b>                                | <b>40</b> |
| 5.1       | Definitions, Theorems, Examples, Paragraphs . . . . . | 40        |
| <b>6</b>  | <b>Highlighting and Presentation Customizations</b>   | <b>47</b> |
| <b>7</b>  | <b>Additional Packages</b>                            | <b>49</b> |
| 7.1       | Tikzinput: Treating TIKZ code as images . . . . .     | 49        |
| 7.2       | Modular Document Structuring . . . . .                | 50        |
| 7.3       | Slides and Course Notes . . . . .                     | 52        |
| 7.4       | Representing Problems and Solutions . . . . .         | 56        |
| 7.5       | Homeworks, Quizzes and Exams . . . . .                | 59        |
| <b>II</b> | <b>Documentation</b>                                  | <b>62</b> |
| <b>8</b>  | <b>sTeX-Basics</b>                                    | <b>63</b> |
| 8.1       | Macros and Environments . . . . .                     | 63        |
| 8.1.1     | HTML Annotations . . . . .                            | 63        |
| 8.1.2     | Babel Languages . . . . .                             | 64        |
| 8.1.3     | Auxiliary Methods . . . . .                           | 64        |
| <b>9</b>  | <b>sTeX-MathHub</b>                                   | <b>65</b> |
| 9.1       | Macros and Environments . . . . .                     | 65        |
| 9.1.1     | Files, Paths, URIs . . . . .                          | 65        |
| 9.1.2     | MathHub Archives . . . . .                            | 66        |
| 9.1.3     | Using Content in Archives . . . . .                   | 67        |
| <b>10</b> | <b>sTeX-References</b>                                | <b>68</b> |
| 10.1      | Macros and Environments . . . . .                     | 68        |
| 10.1.1    | Setting Reference Targets . . . . .                   | 68        |
| 10.1.2    | Using References . . . . .                            | 69        |
| <b>11</b> | <b>sTeX-Modules</b>                                   | <b>70</b> |
| 11.1      | Macros and Environments . . . . .                     | 70        |
| 11.1.1    | The <code>smodule</code> environment . . . . .        | 72        |
| <b>12</b> | <b>sTeX-Module Inheritance</b>                        | <b>74</b> |
| 12.1      | Macros and Environments . . . . .                     | 74        |
| 12.1.1    | SMS Mode . . . . .                                    | 74        |
| 12.1.2    | Imports and Inheritance . . . . .                     | 75        |
| <b>13</b> | <b>sTeX-Symbols</b>                                   | <b>77</b> |
| 13.1      | Macros and Environments . . . . .                     | 77        |
| <b>14</b> | <b>sTeX-Terms</b>                                     | <b>79</b> |
| 14.1      | Macros and Environments . . . . .                     | 79        |
| <b>15</b> | <b>sTeX-Structural Features</b>                       | <b>81</b> |
| 15.1      | Macros and Environments . . . . .                     | 81        |
| 15.1.1    | Structures . . . . .                                  | 81        |

|            |   |            |
|------------|---|------------|
| <b>16</b>  | <b><code>sTeX</code>-Statements</b>   | <b>82</b>  |
| 16.1       | Macros and Environments . . . . .   | 82         |
| <b>17</b>  | <b><code>sTeX</code>-Proofs: Structural Markup for Proofs</b>   | <b>83</b>  |
| <b>18</b>  | <b><code>sTeX</code>-Metatheory</b>   | <b>84</b>  |
| 18.1       | Symbols . . . . .   | 84         |
| <b>III</b> | <b>Extensions</b>   | <b>85</b>  |
| <b>19</b>  | <b><code>Tikzinput</code>: Treating <code>TIKZ</code> code as images</b>                                      | <b>86</b>  |
| 19.1       | Macros and Environments . . . . .   | 86         |
| <b>20</b>  | <b><code>document-structure</code>: Semantic Markup for Open Mathematical Documents in <code>LaTeX</code></b> | <b>87</b>  |
| <b>21</b>  | <b><code>NotesSlides</code> – Slides and Course Notes</b>   | <b>88</b>  |
| <b>22</b>  | <b><code>problem.sty</code>: An Infrastructure for formatting Problems</b>                                    | <b>89</b>  |
| <b>23</b>  | <b><code>hwexam.sty/cls</code>: An Infrastructure for formatting Assignments and Exams</b>                    | <b>90</b>  |
| <b>IV</b>  | <b>Implementation</b>   | <b>91</b>  |
| <b>24</b>  | <b><code>sTeX</code>-Basics Implementation</b>  | <b>92</b>  |
| 24.1       | The <code>sTeXDocument</code> Class . . . . .   | 92         |
| 24.2       | Preliminaries . . . . .   | 93         |
| 24.3       | Messages and logging . . . . .  | 94         |
| 24.4       | HTML Annotations . . . . .  | 95         |
| 24.5       | Babel Languages . . . . .   | 96         |
| 24.6       | Persistence . . . . .   | 97         |
| 24.7       | Auxiliary Methods . . . . .   | 98         |
| <b>25</b>  | <b><code>sTeX</code>-MathHub Implementation</b>   | <b>102</b> |
| 25.1       | Generic Path Handling . . . . .   | 102        |
| 25.2       | PWD and <code>kpsewhich</code> . . . . .  | 104        |
| 25.3       | File Hooks and Tracking . . . . .   | 105        |
| 25.4       | MathHub Repositories . . . . .  | 106        |
| 25.5       | Using Content in Archives . . . . .   | 111        |
| <b>26</b>  | <b><code>sTeX</code>-References Implementation</b>  | <b>115</b> |
| 26.1       | Document URIs and URLs . . . . .  | 115        |
| 26.2       | Setting Reference Targets . . . . .   | 117        |
| 26.3       | Using References . . . . .  | 119        |
| <b>27</b>  | <b><code>sTeX</code>-Modules Implementation</b>   | <b>122</b> |
| 27.1       | The <code>smodule</code> environment . . . . .  | 126        |
| 27.2       | Invoking modules . . . . .  | 132        |

|           |  |            |
|-----------|--|------------|
| <b>28</b> | <b>sTeX-Module Inheritance Implementation</b>  | <b>134</b> |
| 28.1      | SMS Mode . . . . .                             | 134        |
| 28.2      | Inheritance . . . . .                          | 138        |
| <b>29</b> | <b>sTeX-Symbols Implementation</b>             | <b>144</b> |
| 29.1      | Symbol Declarations . . . . .                  | 144        |
| 29.2      | Notations . . . . .                            | 151        |
| 29.3      | Variables . . . . .                            | 160        |
| <b>30</b> | <b>sTeX-Terms Implementation</b>               | <b>166</b> |
| 30.1      | Symbol Invocations . . . . .                   | 166        |
| 30.2      | Terms . . . . .                                | 173        |
| 30.3      | Notation Components . . . . .                  | 177        |
| 30.4      | Variables . . . . .                            | 179        |
| 30.5      | Sequences . . . . .                            | 181        |
| <b>31</b> | <b>sTeX-Structural Features Implementation</b> | <b>182</b> |
| 31.1      | Imports with modification . . . . .            | 183        |
| 31.2      | The feature environment . . . . .              | 191        |
| 31.3      | Structure . . . . .                            | 191        |
| <b>32</b> | <b>sTeX-Statements Implementation</b>          | <b>201</b> |
| 32.1      | Definitions . . . . .                          | 201        |
| 32.2      | Assertions . . . . .                           | 206        |
| 32.3      | Examples . . . . .                             | 210        |
| 32.4      | Logical Paragraphs . . . . .                   | 212        |
| <b>33</b> | <b>The Implementation</b>                      | <b>218</b> |
| 33.1      | Proofs . . . . .                               | 218        |
| 33.2      | Justifications . . . . .                       | 229        |
| <b>34</b> | <b>sTeX-Others Implementation</b>              | <b>230</b> |
| <b>35</b> | <b>sTeX-Metattheory Implementation</b>         | <b>231</b> |
| <b>36</b> | <b>Tikzinput Implementation</b>                | <b>234</b> |
| <b>37</b> | <b>document-structure.sty Implementation</b>   | <b>237</b> |
| 37.1      | Package Options . . . . .                      | 237        |
| 37.2      | Document Structure . . . . .                   | 238        |
| 37.3      | Front and Backmatter . . . . .                 | 242        |
| 37.4      | Global Variables . . . . .                     | 244        |
| <b>38</b> | <b>NotesSlides – Implementation</b>            | <b>245</b> |
| 38.1      | Class and Package Options . . . . .            | 245        |
| 38.2      | Notes and Slides . . . . .                     | 247        |
| 38.3      | Header and Footer Lines . . . . .              | 251        |
| 38.4      | Frame Images . . . . .                         | 253        |
| 38.5      | Colors and Highlighting . . . . .              | 254        |
| 38.6      | Sectioning . . . . .                           | 255        |
| 38.7      | Excursions . . . . .                           | 257        |

|  |            |
|--|------------|
| <b>39 The Implementation</b>                 | <b>259</b> |
| 39.1 Package Options . . . . .               | 259        |
| 39.2 Problems and Solutions . . . . .        | 260        |
| 39.3 Multiple Choice Blocks . . . . .        | 267        |
| 39.4 Including Problems . . . . .            | 268        |
| 39.5 Reporting Metadata . . . . .            | 270        |
| <b>40 Implementation: The hwexam Package</b> | <b>272</b> |
| 40.1 Package Options . . . . .               | 272        |
| 40.2 Assignments . . . . .                   | 273        |
| 40.3 Including Assignments . . . . .         | 276        |
| 40.4 Typesetting Exams . . . . .             | 277        |
| 40.5 Leftovers . . . . .                     | 279        |
| <b>41 References</b>                         | <b>280</b> |

# Part I

## Manual



Boxes like this one contain implementation details that are mostly relevant for more advanced use cases, might be useful to know when debugging, or might be good to know to better understand how something works. They can easily be skipped on a first read.



Boxes like this one explain how some  $\text{\texttt{STeX}}$  concept relates to the MMT/OMDoc system, philosophy or language; see [MMT; Koh06] for introductions.

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



# Chapter 2

## Quickstart

### 2.1 Setup

There are two ways of using  $\text{sTeX}$ : as a

1. way of writing  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  more modularly (object-oriented Math) for creating PDF documents or
2. foundation for authoring active documents in HTML5 instrumented with knowledge management services.

Both are legitimate and useful. The first requires a significantly smaller tool-chain, so we describe it first. The second requires a much more substantial (and experimental) toolchain of knowledge management systems. Both workflows profit from an integrated development environment (IDE), which (also) automates setup as far as possible (see [subsection 2.1.4](#)).

#### 2.1.1 Minimal Setup for the PDF-only Workflow

In the best of all worlds, there is no setup, as you already have a new version of  $\text{T}_{\text{E}}\text{XLive}$  on your system as a  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  enthusiast. If not now is the time to install it; see [\[TL\]](#). You can usually update  $\text{T}_{\text{E}}\text{XLive}$  via a package manager or the  $\text{T}_{\text{E}}\text{XLive}$  manager **tlmgr**.

Alternatively, you can install  $\text{sTeX}$  from CTAN, the Comprehensive  $\text{T}_{\text{E}}\text{X}$  Archive Network; see [\[ST\]](#) for details.

#### 2.1.2 GIT-based Setup for the $\text{sTeX}$ Development Version

If you want use the latest and greatest  $\text{sTeX}$  packages, you can that have not even been released to CTAN, then you can directly clone them from the  $\text{sTeX}$  development repository [\[sTeX\]](#) by the following command-line instructions:

```
cd <stexdir>
git clone https://github.com/slatex/sTeX.git
```

and keep it updated by pulling updates via `git pull` in the cloned  $\text{sTeX}$  directory. Then update your `TEXINPUTS` environment variable, e.g. by placing the following line in your `.bashrc`:

---

<sup>1</sup>NEW PART: MK: reorganized, we do not need the full MKM tool chain

```
export TEXINPUTS="$(TEXINPUTS):<sTeXDIR>//:"
```

### 2.1.3 sTeX Archives (Manual Setup)

Writing semantically annotated sTeX becomes much easier, if we can use well-designed libraries of already annotated content. sTeX provides such libraries as sTeX archives – i.e. GIT repositories at <https://gl.mathhub.info> – most prominently the SMGLoM libraries at <https://gl.mathhub.info/smgloom>.

To do so, we set up a **local MathHub** by creating a MathHub directory `<mhdir>`. Every sTeX archive as an **archive path** `<apath>` and a name `<archive>`. We can clone the sTeX archive by the following command-line instructions:

```
cd <mhdir>/<apath>
git clone https://gl.mathhub.info/smgloom/<archive>.git
```

Note that sTeX archives often depend on other archives, thus you should be prepared to clone these as well – e.g. if `pdflatex` reports missing files. 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](#)).

```
export MATHHUB="<mhdir>"
```

### 2.1.4 The sTeX IDE

We are currently working on an sTeX IDE as an sTeX plugin for VScode; see [\[Sla\]](#). It will feature a setup procedure that automates the setup described above (and below). For additional functionality see the (now obsolete) plugin for sTeX 1 [\[SLS; Stb\]](#).

### 2.1.5 Manual Setup for Active Documents and Knowledge Management Services

Foregoing on the sTeX IDE, we will need several additional (on top of the minimal setup above) pieces of software; namely:

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

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

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

Once set up, we can run `mmt` in a shell and download an archive along with all of its dependencies like this: `lmh install <name-of-repository>`, or a whole *group* of archives; for example, `lmh install smgloom` will download all `smgloom` 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>2</sup>EdNOTE: For now, we require the `sTeX`-branch, requiring manually compiling the MMT sources

## 2.2 A First sTeX Document

Having set everything up, we can write a first sTeX 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 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     \end{sdefinition}
20
21     \begin{sassertion}[name=geometricSeriesConverges,type=theorem]
22       The \symname{geometricSeries} \symname{converges} towards $1$.
23     \end{sassertion}
24 \end{smodule}
25 \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.

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, see [chapter 6](#).

Let's investigate this document in detail to understand the respective parts of the  $\text{\TeX}$  markup infrastructure:

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

**smodule** First, we open a new *module* called `GeometricSeries`. The main purpose of the `smodule` environment is to group the contents and associate it with a *globally unique* identifier (URI), which is computed from the name `GeometricSeries` and the document context. (Depending on your pdf viewer), the URI 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` from the  $\text{\TeX}$  archive `smglom/calculus`, and `realarith` from the  $\text{\TeX}$  archive `smglom/arithmetics`. 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  $\text{\TeX}$  symbols and associated semantic macros (e.g. `\infinitesum`, `\realdive`, `\realpower`) in the imported module available to the current module `GeometricSeries`. The module `GeometricSeries` “exports” all of these symbols to all modules imports it via an `\importmodule{GeometricSeries}` instruction. Additionally it exports the local symbol `\geometricSeries`.

---

**\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 the `amsthm` package.

```
... is the \symname{?series}
```

---

`\symname`

The `\symname`-command prints the name of a symbol, highlights it (based on customizable settings) and associates the text printed with the corresponding symbol.

Note that the argument of `\symref` can be a local or imported symbol (here the `series` symbol is imported from the `series` module).  $\text{\LaTeX}$  tries to determine the full symbol URI from the argument. If there are name clashes in or with the imported symbols, the name of the exporting module can be prepended to the symbol name before the `?` character.

If you hover over the word `series` in the pdf output, you should see a tooltip showing the full URI of the symbol used.

---

`\symref`

The `\symname`-command is a special case of the more general `\symref`-command, which allows customizing the precise text associated with a symbol. `\symref` takes two arguments the first is the symbol name, and the second a variant verbalization of the symbol, e.g. an inflection variant, a different language or a synonym. In our example `\symname{?series}` abbreviates `\symref{?series}{series}`.

```
The \definame{geometricSeries} ...
```

---

`\definame`  
`\definiendum`

The `sdefinition`-environment provides two additional macros, `\definame` and `\definiendum` which behave similarly to `\symname` and `\symref`, but explicitly mark the symbols as *being defined* in this environment, to allow for special highlighting.

```

\[\defeq{\geometricSeries}{\definiens{
  \infinitesum{\svar{n}}{1}{
    \realdivide[frac]{1}{
      \realpower{2}{\svar{n}}
    }
  }}
\].\]

```

The next snippet – set in a math environment – uses several semantic macros imported from (or recursively via) `series` and `realarithmetics`, such as `\defeq`, `\infinitesum`, 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. `\realdivide[frac]{a}{b}` will use the explicit notation named `frac` of the semantic macro `\realdivide`, which yields  $\frac{a}{b}$  instead of  $a/b$ .

---

`\svar` The `\svar{n}` command marks up the `n` as a variable with name `n` and notation `n`.

---



---

`\definiens` The `sdefinition`-environment additionally provides the `\definiens`-command, which allows for explicitly marking up its argument as the *definiens* of the symbol currently being defined.

---

### 2.2.1 OMDoc/xhtml Conversion

So, if we run `pdflatex` on our document, then  $\text{\TeX}$  yields pretty colors and tooltips<sup>1</sup>. But  $\text{\TeX}$  becomes a lot more powerful if we additionally convert our document to `xhtml` while preserving all the  $\text{\TeX}$  markup in the result.

#### TODO VSCode Plugin

Using `Ru\TeX` [RT], 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?infinitiesum" property="stex:OMBIND">
    <munderover displaystyle="true">
      <mo resource="...?series?infinitiesum" property="stex:comp">\Sigma</mo>
      <mrow>
        <mrow resource="1" property="stex:arg">
          <mi resource="var://n" property="stex:OMV">n</mi>
        </mrow>
        <mo resource="...?series?infinitiesum" property="stex:comp">=</mo>
        <mi resource="2" property="stex:arg">1</mi>
      </mrow>
      <mi resource="...?series?infinitiesum" 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">
          <msup resource="...realarith?exponentiation" property="stex:OMA">
            <mi resource="1" property="stex:arg">2</mi>
            <mrow resource="2" property="stex:arg">
              <mi resource="var://n" property="stex:OMV">n</mi>
            </mrow>
          </msup>
        </mrow>
      </mfrac>
    </mrow>
  </mrow>
```

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

...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>
```

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

**mathhub** ( $\langle\textit{directory}\rangle$ ) MathHub folder to search for repositories – this is not necessary if the `MATHHUB` system variable is set.

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

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

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

### 3.1 How Knowledge is Organized in sTeX

sTeX content is organized on multiple levels:

1. sTeX **archives** (see [section 3.2](#)) contain individual `.tex`-files.
2. These may contain sTeX **modules**, introduced via `\begin{smodule}{ModuleName}`.
3. 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.
4. sTeX **expressions** finally are built up from usages of semantic macros.

$\hookrightarrow M \rightarrow$

$\hookrightarrow M \rightarrow$

$\hookrightarrow T \rightarrow$

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

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



$\hookrightarrow M \rightarrow$   
 $\hookrightarrow M \rightarrow$   
 $\hookrightarrow T \rightarrow$

similar constructions) induce MMT `includes` and other *theory morphisms*, thus giving rise to a *theory graph* in the OMDOC sense [RK13].

- Symbol declarations induce OMDOC/MMT *constants*, with optional (formal) *type* and *definiens* components.
- Finally,  $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$  expressions are converted to OMDOC/MMT terms, which use the abstract syntax (and XML encoding) of OPENMATH [Bus+04].

## 3.2 $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$ Archives

### 3.2.1 The Local MathHub-Directory

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

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

1. If the  $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$  package is loaded with the option `mathhub=/path/to/mathhub`, then  $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$  will consider `/path/to/mathhub` as the local MathHub-directory.
2. If the `mathhub` package option is *not* set, but the macro `\mathhub` exists when the  $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$ -package is loaded, then this macro is assumed to point to the local MathHub-directory; i.e. `\def\mathhub{/path/to/mathhub}\usepackage{stex}` will set the MathHub-directory as `path/to/mathhub`.
3. Otherwise,  $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$  will attempt to retrieve the system variable `MATHHUB`, assuming it will point to the local MathHub-directory. Since this variant needs setting up only *once* and is machine-specific (rather than defined in tex code), it is compatible with collaborating and sharing tex content, and hence recommended.
4. Finally, if all else fails,  $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$  will look for a file `~/.stex/mathhub.path`. If this file exists,  $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$  will assume that it contains the path to the local MathHub-directory. This method is recommended on systems where it is difficult to set environment variables.

### 3.2.2 The Structure of $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$ Archives

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

Each such archive needs two subdirectories:

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

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

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

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

- `/source/mod/` – individual  $\text{\TeX}$  modules, containing symbol declarations, notations, and `\begin{sparagraph}[type=symdoc,for=...]` environments for “encyclopaedic” 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 `\inputrefing` snippets, definitions, examples etc. in some desirable order
- `/source/tikz/` – tikz images, as individual `.tex`-files
- `/source/pic/` – image files.<sup>3</sup>

### 3.2.3 MANIFEST.MF-Files

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

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

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

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

`source-base` or

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

<sup>3</sup>EdNOTE: MK: bisher habe ich immer PIC subdirs, soll ich das ändern?

**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.  $\text{\texttt{STeX}}$  ignores this field, but MMT can pick up on them to resolve dependencies, e.g. for `lmh install`.

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

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

---

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

---

|                                       |  |
|---------------------------------------|--|
| $\text{\texttt{\backslash inputref}}$ | $\text{\texttt{\backslash inputref}}[\text{Some/Archive}]\{\text{some/file}\}$ behaves like $\text{\texttt{\backslash mhinput}}$ , but wraps the input in a <code>\begingroup ... \endgroup</code> . When converting to <code>xhtml</code> , the file is not input at all, and instead an <code>html</code> -annotation is inserted that references the file, e.g. for lazy loading. |
|---------------------------------------|--|

In the majority of practical cases  $\text{\texttt{\backslash inputref}}$  is likely to be preferred over  $\text{\texttt{\backslash mhinput}}$  because it leads to less duplication in the generated `xhtml`.

---

|                                      |   |
|--------------------------------------|---|
| $\text{\texttt{\backslash ifinput}}$ | Both $\text{\texttt{\backslash mhinput}}$ and $\text{\texttt{\backslash inputref}}$ set $\text{\texttt{\backslash ifinput}}$ 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. |
|--------------------------------------|---|

---

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

- $\text{\texttt{\backslash addmhbibresource}}\{\text{lib/refs.bib}\}$ , which specifies a bibliography in the `lib` folder in the local archive or
- $\text{\texttt{\backslash addmhbibresource}}[\text{HW/meta-inf}]\{\text{lib/refs.bib}\}$  in another.

---

|                                       |   |
|---------------------------------------|---|
| $\text{\texttt{\backslash libinput}}$ | $\text{\texttt{\backslash libinput}}\{\text{some/file}\}$ searches for a file <code>some/file</code> in |
|---------------------------------------|---|

- the `lib`-directory of the current archive, and
- the `lib`-directory of a `meta-inf`-archive in (any of) the archive groups containing the current archive

and include all found files in reverse order; e.g.  $\text{\texttt{\backslash libinput}}\{\text{preamble}\}$  in a `.tex`-file in `smglom/calculus` will *first* input `.../smglom/meta-inf/lib/preamble.tex` and then `../smglom/calculus/lib/preamble.tex`.

$\text{\texttt{\backslash libinput}}$  will throw an error if *no* candidate for `some/file` is found.

---

`\libusepackage` `\libusepackage`[package-options]{some/file} searches for a file `some/file.sty` in the same way that `\libinput` does, but will call `\usepackage`[package-options]{path/to/some/file} instead of `\input`.  
`\libusepackage` throws an error if not *exactly one* 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.

`\libusepackage` is particularly useful in `preamble.tex` when we want to use custom packages that are not part of  $\text{\TeX}$ Live. In this case we commit the respective packages in one of the `lib` folders and use `\libusepackage` to load them.

## 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 keyword arguments, all of which are optional:

`title` (*<token list>*) to display in customizations.

`type` (*<string>\**) for use in customizations.

`deprecate` (*<module>*) if set, will throw a warning when loaded, urging to use *<module>* instead.

`id` (*<string>*) for cross-referencing.

`ns` (*<URI>*) 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` (*<language>*) if not set, computed from the current file name (e.g. `foo.en.tex`).

`sig` (*<language>*) 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.

$\hookrightarrow$  M  $\rightarrow$  An  $\text{\LaTeX}$  module corresponds to an MMT/OMDoc *theory*. As such it  
 $\hookrightarrow$  M  $\rightarrow$  gets assigned a module URI (*universal resource identifier*) of the form  
 $\hookrightarrow$  T  $\hookrightarrow$   $\langle namespace \rangle ? \langle module-name \rangle$ .

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 `\smodulename`, `\smoduletype` and `\smoduleid`.

For example:

### Example 2

Input:

```
1 \stexpatchmodule[display]
2   {\textbf{Module (\smodulename)}}\par
3   {\par\noindent\textbf{End of Module (\smodulename)}}
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.

$\hookrightarrow$  `\symdecl` introduces a new OMDoc/MMT constant in the current module ( $\Rightarrow$  OMDoc/MMT theory). Correspondingly, they get assigned the URI `<module-URI>?<constant-name>`.

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.

So far we have gained exactly ... nothing by adding the arity information: we cannot do anything with the arguments in the text.

We will now see what we can gain with more machinery.

`\notation`

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 $\binarysymbol{a}{b}$
```

Output:

First:  $a$ ; Second:  $b$

-   $\rightarrow$  Applications of semantic macros, such as `\binarysymbol{a}{b}` are translated to
-   $\rightarrow$  MMT/OMDOC as OMA-terms with head `<OMS name="...?binarysymbol"/>`.
-   $\rightarrow$  Semantic macros with no arguments correspond to OMS directly.

`\comp`

For many semantic services e.g. semantic highlighting or **wikification** (linking user-visible notation components to the definition of the respective symbol they come from), we need to specify the notation components. Unfortunately, there is currently no way the  $\text{\LaTeX}$  engine can infer this by itself, so we have to specify it manually in the notation specification. 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  $\text{\LaTeX}$  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  $\text{\LaTeX}$  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  $\text{\LaTeX}$  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` command combines the functionality of both `\symdecl` and `\notation` with the optional arguments of both:

### Example 7

Input:

```
1 \symdef{newbinarysymbol}[h1,args=2]
2   {\comp{\text{1.: }}#1\comp{\text{; 2.: }}#2}
3 $\newbinarysymbol{a}{b}$
```

Output:

```
1.: a; 2.: b
```

We just declared a new symbol `newbinarysymbol` with `args=2` and immediately provided it with a notation with identifier `h1`. Since `h1` is the *first* (and so far, only) notation supplied for `newbinarysymbol`, using `\newbinarysymbol` without optional argument defaults to this notation.

But one man's meat is another man's poison: it is very subjective what the “default notation” of an operator should be. Different communities have different practices. For instance, the complex unit is written as *i* in Mathematics and as *j* in electrical engineering.



So to allow modular specification and facilitate re-use of document fragments  $\text{\LaTeX}$  allows to re-set notation defaults.

## $\backslash\text{setnotation}$

The first notation provided will stay the default notation unless explicitly changed – this is enabled by the  $\backslash\text{setnotation}$  command:  $\backslash\text{setnotation}\{\text{symbolname}\}\{\text{notation-id}\}$  sets the default notation of  $\backslash\text{symbolname}$  to  $\text{notation-id}$ , i.e. henceforth,  $\backslash\text{symbolname}$  behaves like  $\backslash\text{symbolname}[\text{notation-id}]$  from now on.

Often, a default notation is set right after the corresponding notation is introduced – the starred version  $\backslash\text{notation}^*$  for that reason introduces a new notation and immediately sets it to be the new default notation. So expressed differently, the *first*  $\backslash\text{notation}$  for a symbol behaves exactly like  $\backslash\text{notation}^*$ , and  $\backslash\text{notation}^*\{\text{foo}\}[\text{bar}]\{\dots\}$  behaves exactly like  $\backslash\text{notation}\{\text{foo}\}[\text{bar}]\{\dots\}\backslash\text{setnotation}\{\text{foo}\}[\text{bar}]$ .

## Operator Notations

Once we have a semantic macro with arguments, such as  $\backslash\text{newbinarysymbol}$ , the semantic macro represents the *application* of the symbol to a list of arguments. What if we want to refer to the operator *itself*, though?

We can do so by supplying the  $\backslash\text{notation}$  (or  $\backslash\text{symdef}$ ) with an *operator notation*, indicated with the optional argument  $\text{op=}$ . We can then invoke the operator notation using  $\backslash\text{symbolname}![\text{notation-identifier}]$ . Since operator notations never take arguments, we do not need to use  $\backslash\text{comp}$  in it, the whole notation is wrapped in a  $\backslash\text{comp}$  automatically:

### Example 8

Input:

```
1 \notation{newbinarysymbol}[ab, op={\text{a:}\cdot\text{; b:}\cdot}]
2 {\comp{\text{a:}}#1\comp{\text{; b:}}#2} \symname{newbinarysymbol} is also
3 occasionally written $\newbinarysymbol![ab]$
```

Output:

$\text{newbinarysymbol}$  is also occasionally written  $a: \cdot ; b: \cdot$

$\hookrightarrow$   $\backslash\text{symbolname}!$  is translated to OMDoc/MMT as  $\langle\text{OMS name}=\dots?\text{symbolname}\rangle/$   
 $\hookrightarrow$  directly.

## 3.3.3 Argument Modes

The notations so far used *simple* arguments which we call *mode-i* arguments. Declaring a new symbol with  $\backslash\text{symdecl}\{\text{foo}\}[\text{args}=3]$  is equivalent to writing  $\backslash\text{symdecl}\{\text{foo}\}[\text{args}=iii]$ , indicating that the semantic macro takes three mode-i arguments. However, there are three more argument modes which we will investigate now, namely mode-b, mode-a and mode-B arguments.

## Mode-b Arguments

A mode-b argument represents a *variable* that is *bound* by the symbol in its application, making the symbol a *binding operator*. Typical examples of binding operators are e.g. sums  $\sum$ , products  $\prod$ , integrals  $\int$ , quantifiers like  $\forall$  and  $\exists$ , that  $\lambda$ -operator, etc.

$\hookrightarrow$  Mode-b arguments behave exactly like mode-i arguments within T<sub>E</sub>X, but applications of binding operators, i.e. symbols with mode-b arguments, are translated to OMBIND-terms in OMDoc/MMT, rather than OMA.

For example, we can implement a summation operator binding an index variable and taking lower and upper index bounds and the expression to sum over like this:

### Example 9

Input:

```
1 \symdef{summation}[args=biii]
2   {\mathop{\comp{sum}}_{\#1\comp{=}\#2}^{\#3\#4}}
3   $\summation{\svar{x}}{1}{\svar{n}}{\svar{x}}^2$
```

Output:

$$\sum_{x=1}^n x^2$$

where the variable  $x$  is now *bound* by the `\summation`-symbol in the expression.

## Mode-a Arguments

Mode-a arguments represent a *flexary argument sequence*, i.e. a sequence of arguments of arbitrary length. Formally, operators that take arbitrarily many arguments don't "exist", but in informal mathematics, they are ubiquitous. Mode-a arguments allow us to write e.g. `\addition{a,b,c,d,e}` rather than having to write something like `\addition{a}{\addition{b}{\addition{c}{\addition{d}{e}}}}`!

`\notation` (and consequently `\symdef`, too) take one additional argument for each mode-a argument that indicates how to "accumulate" a comma-separated sequence of arguments. This is best demonstrated on an example.

Let's say we want an operator representing quantification over an ascending chain of elements in some set, i.e. `\ascendingchain{S}{a,b,c,d,e}{t}` should yield  $\forall a <_S b <_S c <_S d <_S e. t$ . The "base"-notation for this operator is simply `{\comp{forall} \#2\comp{.},\#3}`, where `\#2` represents the full notation fragment *accumulated* from `{a,b,c,d,e}`.

The *additional* argument to `\notation` (or `\symdef`) takes the same arguments as the base notation and two *additional* arguments `\#1` and `\#2` representing successive pairs in the mode-a argument, and accumulates them into `\#2`, i.e. to produce  $a <_S b <_S c <_S d <_S e$ , we do `{\#1 \comp{<}}_{\#1} \#2`:

### Example 10

Input:

```

1 \symdef{ascendingchain}[args=iai]
2   {\comp{\forall} #2\comp{. \,} #3}
3   {##1 \comp{<}_{#1} ##2}
4
5 Tadaa: $\ascendingchain{S}{a,b,c,d,e}{t}$

```

Output:

Tadaa:  $\forall a <_S b <_S c <_S d <_S e. t$

If this seems overkill, keep in mind that you will rarely need the single-hash arguments #1,#2 etc. in the a-notation-argument. For a much more representative and simpler example, we can introduce flexary addition via:

### Example 11

Input:

```

1 \symdef{addition}[args=a]{#1}{##1 \comp{+} ##2}
2
3 Tadaa: $\addition{a,b,c,d,e}$

```

Output:

Tadaa:  $a+b+c+d+e$

**The assoc-key** We mentioned earlier that “formally”, flexary arguments don’t really “exist”. Indeed, formally, addition is usually defined as a binary operation, quantifiers bind a single variable etc.

Consequently, we can tell  $\text{\LaTeX}$  (or, rather, MMT/OMDOC) how to “resolve” flexary arguments by providing `\symdecl` or `\symdef` with an optional `assoc`-argument, as in `\symdecl{addition}[args=a,assoc=bin]`. The possible values for the `assoc`-key are:

**bin:** A binary, associative argument, e.g. as in `\addition`

**binl:** A binary, left-associative argument, e.g.  $a^{b^{c^d}}$ , which stands for  $((a^b)^c)^d$

**binr:** A binary, right-associative argument, e.g. as in  $A \rightarrow B \rightarrow C \rightarrow D$ , which stands for  $A \rightarrow (B \rightarrow (C \rightarrow D))$

**pre:** Successively prefixed, e.g. as in  $\forall x, y, z. P$ , which stands for  $\forall x. \forall y. \forall z. P$

**conj:** Conjunctive, e.g. as in  $a = b = c = d$  or  $a, b, c, d \in A$ , which stand for  $a = d \wedge b = d \wedge c = d$  and  $a \in A \wedge b \in A \wedge c \in A \wedge d \in A$ , respectively

**pwconj:** Pairwise conjunctive, e.g. as in  $a \neq b \neq c \neq d$ , which stands for  $a \neq b \wedge a \neq c \wedge a \neq d \wedge b \neq c \wedge b \neq d \wedge c \neq d$

As before, at the PDF level, this annotation is invisible (and without effect), but at the level of the generated OMDoc/MMT this leads to more semantical expressions.

## Mode-B Arguments

Finally, mode-B arguments simply combine the functionality of both `a` and `b` - i.e. they represent an arbitrarily long sequence of variables to be bound, e.g. for implementing quantifiers:

### Example 12

Input:

```
1 \symdef{quantforall}[args=Bi]
2   {\comp{\forall}#1\comp{.}#2}
3   {##1\comp,##2}
4
5 \$\quantforall{\svar{x},\svar{y},\svar{z}}{P}$
```

Output:

$\forall x,y,z.P$

## 3.3.4 Type and Definiens Components

`\symdecl` and `\symdef` take two more optional arguments.  $\text{\TeX}$  largely ignores them (except for special situations we will talk about later), but MMT can pick up on them for additional services. These are the `type` and `def` keys, which expect expressions in math-mode (ideally using semantic macros, of course!)

The `type` and `def` keys correspond to the `type` and `definiens` components of

- $\hookrightarrow$  OMDOC/MMT constants.
- $\rightarrow$  Correspondingly, the name “type” should be taken with a grain of salt, since
- $\rightarrow$  OMDOC/MMT – being foundation-independent – does not a priori implement a fixed typing system.

The `type`-key allows us to provide additional information (given the necessary  $\text{\TeX}$  symbols), e.g. for addition on natural numbers:

### Example 13

Input:

```
1 \symdef{Nat}[type=\set]{\comp{\mathbb N}}
2 \symdef{addition}[
3   type=\funtype{\Nat,\Nat}{\Nat},
4   op=+,
5   args=a
6 ]{#1}{##1 \comp+ ##2}
7
8 \symname{addition} is an operation $\funtype{\Nat,\Nat}{\Nat}$
```

Output:

`addition` is an operation  $\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$

The `def`-key allows for declaring symbols as abbreviations:

#### Example 14

Input:

```
1 \symdef{successor}[
2   type=\funtype{\Nat}{\Nat},
3   def=\fun{\svar{x}}{\addition{\svar{x},1}},
4   op=\mathtt{succ},
5   args=1
6 ]{\comp{\mathtt{succ}{}#1\comp{}}}
7
8 The \symname{successor} operation $\funtype{\Nat}{\Nat}$
9 is defined as $\fun{\svar{x}}{\addition{\svar{x},1}}$
```

Output:

The `successor` operation  $\mathbb{N} \rightarrow \mathbb{N}$  is defined as  $x \mapsto x+1$

### 3.3.5 Precedences and Automated Bracketing

Having done `\addition`, the obvious next thing to implement is `\multiplication`. This is straight-forward in theory:

#### Example 15

Input:

```
1 \symdef{multiplication}[
2   type=\funtype{\Nat,\Nat}{\Nat},
3   op=\cdot,
4   args=a
5 ]{\#1}{\#1 \comp\cdot \#2}
6
7 \symname{multiplication} is an operation $\funtype{\Nat,\Nat}{\Nat}$
```

Output:

`multiplication` is an operation  $\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$

However, if we *combine* `\addition` and `\multiplication`, we notice a problem:

#### Example 16

Input:

```
1 $\addition{a,\multiplication{b,\addition{c,\multiplication{d,e}}}}$
```

Output:

$a+b \cdot c+d \cdot e$

We all know that  $\cdot$  binds stronger than  $+$ , so the output  $a+b\cdot c+d\cdot e$  does not actually reflect the term we wrote. We can of course insert parentheses manually

#### Example 17

Input:

```
1 $\addition{a,\multiplication{b,(\addition{c,\multiplication{d,e}})}}$
```

Output:

$$a+b\cdot(c+d\cdot e)$$

but we can also do better by supplying *precedences* and have  $\text{\TeX}$  insert parentheses automatically.

For that purpose, `\notation` (and hence `\symdef`) take an optional argument `prec=<opprec>;<argprec1>x...x<argprec n>`.

We will investigate the precise meaning of `<opprec>` and the `<argprec>`s shortly – in the vast majority of cases, it is perfectly sufficient to think of `prec=` taking a single number and having that be *the* precedence of the notation, where lower precedences (somewhat counterintuitively) bind stronger than higher precedences. So fixing our notations for `\addition` and `\multiplication`, we get:

#### Example 18

Input:

```
1 \notation{multiplication}[
2   op=\cdot,
3   prec=50
4 ]{#1}{##1 \comp\cdot ##2}
5 \notation{addition}[
6   op=+,
7   prec=100
8 ]{#1}{##1 \comp+ ##2}
9
10 $\addition{a,\multiplication{b,\addition{c,\multiplication{d,e}}}}$
```

Output:

$$a+b\cdot(c+d\cdot e)$$

Note that the precise numbers used for precedences are pretty arbitrary – what matters is which precedences are higher than which other precedences when used in conjunction.

---

`\infprec`  
`\neginfprec`

It is occasionally useful to have “infinitely” high or low precedences to enforce or forbid automated bracketing entirely – for those purposes, `\infprec` and `\neginfprec` exist (which are implemented as the maximal and minimal integer values accordingly).

More precisely, each notation takes

1. One *operator precedence* and
2. one *argument precedence* for each argument.

By default, all precedences are 0, unless the symbol takes no argument, in which case the operator precedence is `\neginfprec` (negative infinity). If we only provide a single number, this is taken as both the operator precedence and all argument precedences.

$\text{\S}\text{\TeX}$  decides whether to insert parentheses by comparing operator precedences to a *downward precedence*  $p_d$  with initial value `\infprec`. When encountering a semantic macro,  $\text{\S}\text{\TeX}$  takes the operator precedence  $p_{op}$  of the notation used and checks whether  $p_{op} > p_d$ . If so,  $\text{\S}\text{\TeX}$  insert parentheses.

When  $\text{\S}\text{\TeX}$  steps into an argument of a semantic macro, it sets  $p_d$  to the respective argument precedence of the notation used.

In the example above:



1.  $\text{\S}\text{\TeX}$  starts out with  $p_d = \text{\code{\infprec}}$ .
2.  $\text{\S}\text{\TeX}$  encounters `\addition` with  $p_{op} = 100$ . Since  $100 \not> \text{\code{\infprec}}$ , it inserts no parentheses.
3. Next,  $\text{\S}\text{\TeX}$  encounters the two arguments for `\addition`. Both have no specifically provided argument precedence, so  $\text{\S}\text{\TeX}$  uses  $p_d = p_{op} = 100$  for both and recurses.
4. Next,  $\text{\S}\text{\TeX}$  encounters `\multiplication{b,...}`, whose notation has  $p_{op} = 50$ .
5. We compare to the current downward precedence  $p_d$  set by `\addition`, arriving at  $p_{op} = 50 \not> 100 = p_d$ , so  $\text{\S}\text{\TeX}$  again inserts no parentheses.
6. Since the notation of `\multiplication` has no explicitly set argument precedences,  $\text{\S}\text{\TeX}$  uses the operator precedence for all arguments of `\multiplication`, hence sets  $p_d = p_{op} = 50$  and recurses.
7. Next,  $\text{\S}\text{\TeX}$  encounters the inner `\addition{c,...}` whose notation has  $p_{op} = 100$ .
8. We compare to the current downward precedence  $p_d$  set by `\multiplication`, arriving at  $p_{op} = 100 > 50 = p_d$  – which finally prompts  $\text{\S}\text{\TeX}$  to insert parentheses, and we proceed as before.

### 3.3.6 Variables

All symbol and notation declarations require a module with which they are associated, hence the commands `\symdecl`, `\notation`, `\symdef` etc. are disabled outside of `smodule`-environments.

Variables are different – variables are allowed everywhere, are not exported when the current module (if one exists) is imported (via `\importmodule` or `\usemodule`) and (also unlike symbol declarations) “disappear” at the end of the current  $\text{\TeX}$  group.

---

`\svar`

So far, we have always used variables using `\svar{n}`, which marks-up  $n$  as a variable with name `n`. More generally, `\svar[foo]{<texcode>}` marks-up the arbitrary `<texcode>` as representing a variable with name `foo`.

Of course, this makes it difficult to reuse variables, or introduce “functional” variables with arities  $> 0$ , or provide them with a type or definiens.

---

`\vardef`

For that, we can use the `\vardef` command. Its syntax is largely the same as that of `\symdef`, but unlike symbols, variables have only one notation (TODO: so far?), hence there is only `\vardef` and no `\vardecl`.

#### Example 19

Input:

```

1 \vardef{varf}[
2   name=f,
3   type=\funtype{\Nat}{\Nat},
4   op=f,
5   args=1,
6   prec=0;\neginfp
7 ]{\comp{f}#1}
8 \vardef{varn}[name=n,type=\Nat]{\comp{n}}
9 \vardef{varx}[name=x,type=\Nat]{\comp{x}}
10
11 Given a function $\varf!:\funtype{\Nat}{\Nat}$,
12 by $\addition{\varf!,\varn}$ we mean the function
13 $\fun{\varx}{\varf{\addition{\varx,\varn}}}$

```

Output:

Given a function  $f : \mathbb{N} \rightarrow \mathbb{N}$ , by  $f+n$  we mean the function  $x \mapsto f(x+n)$

(of course, “lifting” addition in the way described in the previous example is an operation that deserves its own symbol rather than abusing `\addition`, but... well.)

TODO: bind=forall/exists

### 3.3.7 Variable Sequences

Variable *sequences* occur quite frequently in informal mathematics, hence they deserve special support. Variable sequences behave like variables in that they disappear at the end of the current  $\text{\TeX}$  group and are not exported from modules, but their declaration is quite different.

---

`\varseq`

A variable sequence is introduced via the command `\varseq`, which takes the usual optional arguments `name` and `type`. It then takes a starting index, an end index and a *notation* for the individual elements of the sequence parametric in an index. Note that both the starting as well as the ending index may be variables.

This is best shown by example:

#### Example 20

Input:



```

1 \vardef{varn}[name=n,type=\Nat]{\comp{n}}
2 \varseq{seqa}[name=a,type=\Nat]{1}{\varn}{\comp{a}_{#1}}
3
4 The  $i$ th index of  $\seqa!$  is  $\seqa{i}$ .

```

Output:

The  $i$ th index of  $a_1, \dots, a_n$  is  $a_i$ .

Note that the syntax `\seqa!` now automatically generates a presentation based on the starting and ending index.

**TODO: more notations for invoking sequences.**

Notably, variable sequences are nicely compatible with `a`-type arguments, so we can do the following:

### Example 21

Input:

```
1  $\addition{\seqa}$ 
```

Output:

$a_1 + \dots + a_n$

Sequences can be *multidimensional* using the `args`-key, in which case the notation's arity increases and starting and ending indices have to be provided as a comma-separated list:

### Example 22

Input:

```

1 \vardef{varm}[name=m,type=\Nat]{\comp{m}}
2 \varseq{seqa}[
3   name=a,
4   args=2,
5   type=\Nat,
6 ]{1,1}{\varn,\varm}{\comp{a}_{#1}^{\#2}}
7
8  $\seqa!$  and  $\addition{\seqa}$ 

```

Output:

$a_1^1, \dots, a_n^m$  and  $a_1^1 + \dots + a_n^m$

We can also explicitly provide a “middle” segment to be used, like such:

### Example 23

Input:

```

1 \varseq{seqa}[
2   name=a,
3   type=\Nat,
4   args=2,
5   mid={\comp{a}_{\varn}^1,\comp{a}_1^2,\ellipses,\comp{a}_{1}^{\varm}}
6 ]{1,1}{\varn,\varm}{\comp{a}_{\#1}^{\#2}}
7
8 $\seqa!$ and $\addition{\seqa}$

```

Output:

$$a_1^1, \dots, a_n^1, a_1^2, \dots, a_1^m, \dots, a_n^m \text{ and } a_1^1 + \dots + a_n^1 + a_1^2 + \dots + a_1^m + \dots + a_n^m$$

### 3.4 Module Inheritance and Structures

The  $\mathsf{\S T E X}$  features for modular document management are inherited from the OM-Doc/MMT model that organizes knowledge into a graph, where the nodes are theories (called modules in  $\mathsf{\S T E X}$ ) and the edges are truth-preserving mappings (called theory morphisms in MMT). We have already seen modules/theories above.

Before we get into theory morphisms in  $\mathsf{\S T E X}$  we will see a very simple application of modules: managing multilinguality modularly.

#### 3.4.1 Multilinguality and Translations

If we load the  $\mathsf{\S T E X}$  document class or package with the option `lang=<lang>`,  $\mathsf{\S T E X}$  will load the appropriate `babel` language for you – e.g. `lang=de` will load the `babel` language `ngerman`. Additionally, it makes  $\mathsf{\S T E X}$  aware of the current document being set in (in this example) *german*. This matters for reasons other than mere `babel`-purposes, though:

Every *module* is assigned a language. If no  $\mathsf{\S T E X}$  package option is set that allows for inferring a language,  $\mathsf{\S T E X}$  will check whether the current file name ends in e.g. `.en.tex` (or `.de.tex` or `.fr.tex`, or...) and set the language accordingly. Alternatively, a language can be explicitly assigned via `\begin{smodule}[lang=<language>]{Foo}`.

Technically, each `smodule`-environment induces *two* OMDoc/MMT theories:  
 $\hookrightarrow$  `\begin{smodule}[lang=<lang>]{Foo}` generates a theory `some/namespace?Foo` that only contains the “formal” part of the module – i.e. exactly the content that is exported when using `\importmodule`.  
 $\rightsquigarrow$  Additionally, MMT generates a *language theory* `some/namespace/Foo?<lang>` that includes `some/namespace?Foo` and contains all the other document content – variable declarations, includes for each `\usemodule`, etc.

Notably, the language suffix in a filename is ignored for `\usemodule`, `\importmodule` and in generating/computing URIs for modules. This however allows for providing *translations* for modules between languages without needing to duplicate content:

If a module `Foo` exists in e.g. english in a file `Foo.en.tex`, we can provide a file `Foo.de.tex` right next to it, and write `\begin{smodule}[sig=en]{Foo}`. The `sig`-key

then signifies, that the “signature” of the module is contained in the *english* version of the module, which is immediately imported from there, just like `\importmodule` would.

Additionally to translating the informal content of a module file to different languages, it also allows for customizing notations between languages. For example, the *least common multiple* of two numbers is often denoted as  $\text{lcm}(a, b)$  in english, but is called *kleinstes gemeinsames Vielfaches* in german and consequently denoted as  $\text{kgV}(a, b)$  there.

We can therefore imagine a german version of an lcm-module looking something like this:

```
1 \begin{smodule}[sig=en]{lcm}
2   \notation*{lcm}[de]{\comp{\mathtt{kgV}}}{#1,#2}}
3
4   Das \symref{lcm}{kleinste gemeinsame Vielfache}
5    $\text{lcm}\{a,b\}$  von zwei Zahlen  $a,b$  ist...
6 \end{smodule}
```

If we now do `\importmodule{lcm}` (or `\usemodule{lcm}`) within a *german* document, it will also load the content of the german translation, including the `de`-notation for `\lcm`.

### 3.4.2 Simple Inheritance and Namespaces

---

`\importmodule`  
`\usemodule`

---

`\importmodule`[Some/Archive]{path?ModuleName} is only allowed within an `smodule`-environment and makes the symbols declared in `ModuleName` available therein. Additionally the symbols of `ModuleName` will be exported if the current module is imported somewhere else via `\importmodule`.

`\usemodule` behaves the same way, but without exporting the content of the used module.

It is worth going into some detail how exactly `\importmodule` and `\usemodule` resolve their arguments to find the desired module – which is closely related to the *namespace* generated for a module, that is used to generate its URI.



Ideally,  $\text{\LaTeX}$  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  $\text{\LaTeX}$  can resolve them accordingly. Largely, users need not concern themselves with namespaces at all, but for completeness sake, we describe how they are constructed:

- If `\begin{smodule}{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.



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.



Conversely, here is how namespaces/URIs and file paths are computed in import statements, exemplary `\importmodule`:

- `\importmodule{Foo}` outside of an archive refers to module `Foo` in the current namespace. Consequently, `Foo` must have been declared earlier in the same document or, if not, in a file `Foo[.<lang>].tex` in the same directory.
- The same statement *within* an archive refers to either the module `Foo` declared earlier in the same document, or otherwise to the module `Foo` in the archive's top-level namespace. In the latter case, it has to be declared in a file `Foo[.<lang>].tex` directly in the archive's `source`-folder.
- Similarly, in `\importmodule{some/path?Foo}` the path `some/path` refers to either the sub-directory and relative namespace path of the current directory and namespace outside of an archive, or relative to the current archive's top-level namespace and `source`-folder, respectively.

The module `Foo` must either be declared in the file `<top-directory>/some/path/Foo[.<lang>].tex`, or in `<top-directory>/some/path[.<lang>].tex` (which are checked in that order).

- Similarly, `\importmodule[Some/Archive]{some/path?Foo}` is resolved like the previous cases, but relative to the archive `Some/Archive` in the mathhub-directory.
- Finally, `\importmodule{full://uri?Foo}` naturally refers to the module `Foo` in the namespace `full://uri`. Since the file this module is declared in can not be determined directly from the URI, the module must be in memory already, e.g. by being referenced earlier in the same document.

Since this is less compatible with a modular development, using full URIs directly is strongly discouraged, unless the module is declared in the current file directly.

---

## `\STEXexport`

`\importmodule` and `\usemodule` import all symbols, notations, semantic macros and (recursively) `\importmodules`. If you want to additionally export e.g. convenience macros and other (S<sub>T</sub>E<sub>X</sub>) code from a module, you can use the command `\STEXexport{<code>}` in your module. Then `<code>` is executed (both immediately and) every time the current module is opened via `\importmodule` or `\usemodule`.



Note, that `\newcommand` defines macros *globally* and throws an error if the macro already exists, potentially leading to low-level L<sup>A</sup>T<sub>E</sub>X errors if we put a `\newcommand` in an `\STEXexport` and the `<code>` is executed more than once in a document – which can happen easily.

A safer alternative is to use macro definition principles, that are safe to use even if the macro being defined already exists, and ideally are local to the current T<sub>E</sub>X



group, such as `\def` or `\let`.

### 3.4.3 The `mathstructure` Environment

A common occurrence in mathematics is bundling several interrelated “declarations” together into *structures*. For example:

- A *monoid* is a structure  $\langle M, \circ, e \rangle$  with  $\circ : M \times M \rightarrow M$  and  $e \in M$  such that...
- A *topological space* is a structure  $\langle X, \mathcal{T} \rangle$  where  $X$  is a set and  $\mathcal{T}$  is a topology on  $X$
- A *partial order* is a structure  $\langle S, \leq \rangle$  where  $\leq$  is a binary relation on  $S$  such that...

This phenomenon is important and common enough to warrant special support, in particular because it requires being able to *instantiate* such structures (or, rather, structure *signatures*) in order to talk about (concrete or variable) *particular* monoids, topological spaces, partial orders etc.

`mathstructure` The `mathstructure` environment allows us to do exactly that. It behaves exactly like the `smodule` environment, but is itself only allowed inside an `smodule` environment, and allows for instantiation later on.

How this works is again best demonstrated by example:

#### Example 24

Input:

```
1 \begin{mathstructure}{monoid}
2   \symdef{universe}[type=\set]{\comp{U}}
3   \symdef{op}[
4     args=2,
5     type=\funtype{\universe,\universe}{\universe},
6     op=\circ
7   ]{\#1 \comp{\circ} \#2}
8   \symdef{unit}[type=\universe]{\comp{e}}
9 \end{mathstructure}
10
11 A \symname{monoid} is...
```

Output:

A `monoid` is...

Note that the `\symname{monoid}` is appropriately highlighted and (depending on your pdf viewer) shows a URI on hovering – implying that the `mathstructure` environment has generated a *symbol* `monoid` for us. It has not generated a semantic macro though, since we can not use the `monoid`-symbol *directly*. Instead, we can instantiate it, for example for integers:

#### Example 25

Input:

```

1 \symdef{Int}[type=\set]{\comp{\mathbb Z}}
2 \symdef{addition}[
3   type=\funtype{\Int,\Int}{\Int},
4   args=2,
5   op=+
6 ]{##1 \comp{+} ##2}
7 \symdef{zero}[type=\Int]{\comp{0}}
8
9 $\mathstruct{\Int,\addition!,\zero}$ is a \symname{monoid}.

```

Output:

$\langle \mathbb{Z}, +, 0 \rangle$  is a monoid.

So far, we have not actually instantiated monoid, but now that we have all the symbols to do so, we can:

### Example 26

Input:

```

1 \instantiate{intmonoid}{monoid}{\mathbb{Z}_{+,0}}[
2   universe = Int ,
3   op = addition ,
4   unit = zero
5 ]
6
7 $\intmonoid{universe}$, $\intmonoid{unit}$ and $\intmonoid{op}\{a\}\{b\}$.
8
9 Also: $\intmonoid!$

```

Output:

$\mathbb{Z}$ , 0 and  $a+b$ .  
Also:  $\mathbb{Z}_{+,0}$

### \instantiate

So summarizing: `\instantiate` takes four arguments: The (macro-)name of the instance, a key-value pair assigning declarations in the corresponding `mathstructure` to symbols currently in scope, the name of the `mathstructure` to instantiate, and lastly a notation for the instance itself.

It then generates a semantic macro that takes as argument the name of a declaration in the instantiated `mathstructure` and resolves it to the corresponding instance of that particular declaration.

`\instantiate` and `mathstructure` make use of the *Theories-as-Types* paradigm  $\hookrightarrow$  (see [MRK18]):  
 $\hookrightarrow$  `mathstructure{<name>}` simply creates a nested theory with name `<name>-structure`. The *constant* `<name>` is defined as `Mod(<name>-structure)` – a *dependent record type with manifest fields*, the fields of which are generated

$\hookrightarrow M$  from (and correspond to) the constants in `<name>-structure`.  
 $\hookrightarrow M$  `\instantiate` generates a constant whose definiens is a record term of type `Mod(<name>-structure)`, with the fields assigned based on the respective key-value-list.  
 $\rightsquigarrow T$

Notably, `\instantiate` throws an error if not *every* declaration in the instantiated `mathstructure` is being assigned.

You might consequently ask what the usefulness of `mathstructure` even is.

`\varinstantiate`

The answer is that we can also instantiate a `mathstructure` with a *variable*. The syntax of `\varinstantiate` is equivalent to that of `\instantiate`, but all of the key-value-pairs are optional, and if not explicitly assigned (to a symbol *or* a variable declared with `\vardef`) inherit their notation from the one in the `mathstructure` environment.

This allows us to do things like:

#### Example 27

Input:

```

1 \varinstantiate{varM}{monoid}{M}
2
3 A \symname{monoid} is a structure
4 $\varM! := \mathstruct{\varM{universe}, \varM{op}!, \varM{unit}}$
5 such that
6 $\varM{op}!: \funtype{\varM{universe}, \varM{universe}}{\varM{universe}}$ ...

```

Output:

A `monoid` is a structure  $M := \langle U, \circ, e \rangle$  such that  $\circ : U \times U \rightarrow U$  ...

.

and

#### Example 28

Input:

```

1 \varinstantiate{varMb}{monoid}{M_2}[universe = Int]
2
3 Let $\varMb! := \mathstruct{\varMb{universe}, \varMb{op}!, \varMb{unit}}$
4 be a \symname{monoid} on $\Int$ ...

```

Output:

Let  $M_2 := \langle \mathbb{Z}, \circ, e \rangle$  be a `monoid` on  $\mathbb{Z}$  ...

.

We will return to these two example later, when we also know how to handle the *axioms* of a monoid.

### 3.4.4 The copymodule Environment

TODO: explain

Given modules:

#### Example 29

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 30

Input:

```

1 \begin{smodule}{ring}
2   \begin{copymodule}{group}{addition}
3     \renamedecl[name=universe]{universe}{runiverse}
4     \renamedecl[name=plus]{operation}{rplus}
5     \renamedecl[name=zero]{unit}{rzero}
6     \renamedecl[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    \renamedecl[name=times]{operation}{rtimes}
14    \renamedecl[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: $\rtimes a\{rplus c\{rtimes d\}}$
19 \end{smodule}

```

Output:

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



TODO: explain donotclone

### 3.4.5 The `interpretmodule` Environment

TODO: explain

#### Example 31

Input:

```
1 \begin{smodule}{int}
2   \syndef{Integers}{\comp{\mathbb Z}}
3   \syndef{plus}[args=2,op=+]{#1 \comp+ #2}
4   \syndef{zero}{\comp0}
5   \syndef{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:

## 3.5 Primitive Symbols (The $\text{\S}\text{\TeX}$ Metatheory)

The `stex-metatheory` package contains  $\text{\S}\text{\TeX}$  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). As such, it serves as the default meta theory for any  $\text{\S}\text{\TeX}$  module.

We can also see the `stex-metatheory` as a foundation of mathematics in the sense of [Rab15], albeit an informal one (the ones discussed there are all formal foundations). The state of the `stex-metatheory` is necessarily incomplete, and will stay so for a long while: It arises as a collection of empirically useful symbols that are collected as more and more mathematics are encoded in  $\text{\S}\text{\TeX}$  and are classified as foundational.

Formal 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.

We make this theory part of the  $\text{\S}\text{\TeX}$  collection rather than encoding it in  $\text{\S}\text{\TeX}$  itself<sup>4</sup>

---

<sup>4</sup>EdNOTE: MK: why? continue

## Chapter 4

# Using $\text{\TeX}$ Symbols

Given a symbol declaration `\symdecl{symbolname}`, we obtain a semantic macro `\symbolname`. We can use this semantic macro in math mode to use its notation(s), and we can use `\symbolname!` in math mode to use its operator notation(s). What else can we do?

### 4.1 `\symref` and its variants

---

`\symref`  
`\symname`

---

We have already seen `\symname` and `\symref`, the latter being the more general.

`\symref{<symbolname>}{<code>}` marks-up `<code>` as referencing `<symbolname>`. Since quite often, the `<code>` should be (a variant of) the name of the symbol anyway, we also have `\symname{<symbolname>}`.

Note that `\symname` uses the *name* of a symbol, not its macroname. More precisely, `\symname` will insert the name of the symbol with “-” replaced by spaces. If a symbol does not have an explicit `name=` given, the two are equal – but for `\symname` it often makes sense to make the two explicitly distinct. For example:

#### Example 32

Input:

```
1 \symdef{Nat}[
2   name=natural-number,
3   type=\set
4 ]{\comp{\mathbb{N}}}
5
6 A \symname{Nat} is...
```

Output:

A natural number is...

`\symname` takes two additional optional arguments, `pre=` and `post=` that get prepended or appended respectively to the symbol name.

`\Symname`

Additionally, `\Symname` behaves exactly like `\symname`, but will capitalize the first letter of the name:

### Example 33

Input:

```
1 \Symname[post=s]{Nat} are...
```

Output:

```
Natural numbers are...
```



This is as good a place as any other to explain how  $\text{\TeX}$  resolves a string `symbolname` to an actual symbol.

If `\symbolname` is a semantic macro, then  $\text{\TeX}$  has no trouble resolving `symbolname` to the full URI of the symbol that is being invoked.

However, especially in `\symname` (or if a symbol was introduced using `\symdecl*` without generating a semantic macro), we might prefer to use the *name* of a symbol directly for readability – e.g. we would want to write `A \symname{natural-number} is...` rather than `A \symname{Nat} is...`.  $\text{\TeX}$  attempts to handle this case thusly:

If `string` does *not* correspond to a semantic macro `\string` and does *not* contain a `?`, then  $\text{\TeX}$  checks all symbols currently in scope until it finds one, whose name is `string`. If `string` is of the form `pre?name`,  $\text{\TeX}$  first looks through all modules currently in scope, whose full URI ends with `pre`, and then looks for a symbol with name `name` in those. This allows for disambiguating more precisely, e.g. by saying `\symname{Integers?addition}` or `\symname{RealNumbers?addition}` in the case where several `additions` are in scope.

## 4.2 Marking Up Text and On-the-Fly Notations

We can also use semantic macros outside of text mode though, which allows us to annotate arbitrary text fragments.

Let us assume again, that we have `\symdef{addition}[args=2]{#1 \comp+ #2}`. Then we can do

### Example 34

Input:

```
1 \addition{\comp{The sum of} \arg{\$svar{n}}\$} \comp{ and } \arg{\$svar{m}}\$}
2 is...
```

Output:

```
The sum of  $n$  and  $m$  is...
```

...which marks up the text fragment as representing an *application* of the `addition`-symbol to two argument  $n$  and  $m$ .

$\hookrightarrow$  M  $\rightarrow$  As expected, the above example is translated to OMDoc/MMT as an  
 $\rightarrow$  M  $\rightarrow$  OMA with `<OMS name="...?addition"/>` as head and `<OMV name="n"/>` and  
 $\rightarrow$  T  $\rightarrow$  `<OMV name="m"/>` as arguments.



Note the difference in treating “arguments” between math mode and text mode. In math mode the (in this case two) tokens/groups following the `\addition` macro are treated as arguments to the addition function, whereas in text mode the group following `\addition` is taken to be the ad-hoc presentation. We drill in on this now.

## \arg

In text mode, every semantic macro takes exactly one argument, namely the text-fragment to be annotated. The `\arg` command is only valid within the argument to a semantic macro and marks up the *individual arguments* for the symbol.

We can also use semantic macros in text mode to invoke an operator itself instead of its application, with the usual syntax using `!`:

### Example 35

Input:

```
1 \addition!{Addition} is...
```

Output:

```
Addition is...
```

Indeed, `\symbolname!{<code>}` is exactly equivalent to `\symref{symbolname}{<code>}` (the latter is in fact implemented in terms of the former).

`\arg` also allows us to switch the order of arguments around and “hide” arguments: For example, `\arg[3]{<code>}` signifies that `<code>` represents the *third* argument to the current operator, and `\arg*[i]{<code>}` signifies that `<code>` represents the *i*th argument, but it should not produce any output (it is exported in the xhtml however, so that MMT and other systems can pick up on it).<sup>5</sup>

### Example 36

Input:

```
1 \addition{\comp{adding}
2   \arg[2]{\svar{k}$}
3   \arg*{\svar{n}}{\svar{m}}}} yields...
```

Output:

<sup>5</sup>EdNOTE: MK: I do not understand why we have to/want to give the second `arg*`; I think this must be elaborated on.

adding  $k$  yields...

Note that since the second `\arg` has no explicit argument number, it automatically represents the first not-yet-given argument – i.e. in this case the first one.<sup>6</sup>

The same syntax can be used in math mod as well. This allows us to spontaneously introduce new notations on the fly. We can activate it using the starred variants of semantic macros:

### Example 37

Input:

```
1 Given $\addition{\svar{n}}{\svar{m}}$, then
2 $\addition*{
3   \arg*{\addition{\svar{n}}{\svar{m}}}
4   \comp{+}
5   \arg{\svar{k}}
6 }$ yields...
```

Output:

Given  $n+m$ , then  $+k$  yields...

## 4.3 Referencing Symbols and Statements

TODO: references documentation

<sup>6</sup>EdNOTE: MK: I do not understand this at all.

## Chapter 5

# sTeX Statements

### 5.1 Definitions, Theorems, Examples, Paragraphs

As mentioned earlier, we can semantically mark-up *statements* such as definitions, theorems, lemmata, examples, etc.

The corresponding environments for that are:

- `sdefinition` for definitions,
- `sassertion` for assertions, i.e. propositions that are declared to be *true*, such as theorems, lemmata, axioms,
- `sexample` for examples and counterexamples, and
- `sparagraph` for “other” semantic paragraphs, such as comments, remarks, conjectures, etc.

The *presentation* of these environments can be customized to use e.g. predefined theorem-environments, see [chapter 6](#) for details.

All of these environments take optional arguments in the form of `key=value`-pairs. Common to all of them are the keys `id=` (for cross-referencing, see [section 4.3](#)), `type=` for customization (see [chapter 6](#)) and additional information (e.g. definition principles, “difficulty” etc), as well as `title=` (for giving the paragraph a title), and finally `for=`.

The `for=` key expects a comma-separated list of existing symbols, allowing for e.g. things like

#### Example 38

Input:

```
1 \begin{sexample}[
2   id=additionandmultiplication.ex,
3   for={addition,multiplication},
4   type={trivial,boring},
5   title={An Example}
6 ]
7   $\addition{2,3}$ is $5$, $\multiplication{2,3}$ is $6$.
8 \end{sexample}
```

Output:

**Example 5.1.1** (An Example).  $2+3$  is 5,  $2\cdot 3$  is 6.

---

`\definiendum`  
`\definame`  
`\Definame`

---

**sdefinition** (and **sparagraph** with `type=symdoc`) introduce three new macros: **definiendum** behaves like **symref** (and **definame/Definame** like **symname/Symname**, respectively), but highlights the referenced symbol as *being defined* in the current definition.

$\hookrightarrow$  M  $\rightarrow$  The special `type=symdoc` for **sparagraph** is intended to be used for “informal definitions”, or encyclopedia-style descriptions for symbols.  
 $\hookrightarrow$  M  $\rightarrow$  The MMT system can use those (in lieu of an actual **sdefinition** in scope) to present to users, e.g. when hovering over symbols.  
 $\hookrightarrow$  T  $\rightarrow$

---

`\definiens`

---

Additionally, **sdefinition** (and **sparagraph** with `type=symdoc`) introduces `\definiens`[<optional sym which marks up <code> as being the explicit *definiens* of <optional symbolname> (in case `for=` has multiple symbols)].

All four statement environments – i.e. **sdefinition**, **sassertion**, **sexample**, and **sparagraph** – also take an optional parameter `name=` – if this one is given a value, the environment will generate a *symbol* by that name (but with no semantic macro). Not only does this allow for `\symref` et al, it allows us to resume our earlier example for monoids much more nicely:<sup>7</sup>

**Example 39**

Input:

---

<sup>7</sup>EdNOTE: MK: we should reference the example explicitly here.

```

1 \begin{mathstructure}{monoid}
2   \symdef{universe}[type=\set]{\comp{U}}
3   \symdef{op}[
4     args=2,
5     type=\funtype{\universe,\universe}{\universe},
6     op=\circ
7   ]{#1 \comp{\circ} #2}
8   \symdef{unit}[type=\universe]{\comp{e}}
9
10  \begin{sparagraph}[type=symdoc,for=monoid]
11    A \definame{monoid} is a structure
12    $\mathstruct{\universe,\op!,\unit}$
13    where $\op!:\funtype{\universe}{\universe}$ and
14    $\inset{\unit}{\universe}$ such that
15
16    \begin{sassertion}[name=associative,
17      type=axiom,
18      title=Associativity]
19      $\op!$ is associative
20    \end{sassertion}
21    \begin{sassertion}[name=isunit,
22      type=axiom,
23      title=Unit]
24      $\equal{\op{\svar{x}}{\unit}}{\svar{x}}$
25      for all $\inset{\svar{x}}{\universe}$
26    \end{sassertion}
27  \end{sparagraph}
28 \end{mathstructure}
29
30 An example for a \symname{monoid} is...

```

Output:

A **monoid** is a structure  $\langle U, \circ, e \rangle$  where  $\circ : U \rightarrow U$  and  $e \in U$  such that

**Axiom 5.1.2** (Associativity).  $\circ$  is associative

**Axiom 5.1.3** (Unit).  $x \circ e = x$  for all  $x \in U$

An example for a **monoid** is...

.

The main difference to before<sup>8</sup> is that the two **sassertions** now have **name=** attributes. Thus the **mathstructure monoid** now contains two additional symbols, namely the axioms for associativity and that  $e$  is a unit. Note that both symbols do not represent the mere *propositions* that e.g.  $\circ$  is associative, but *the assertion that it is actually true* that  $\circ$  is associative.

If we now want to instantiate **monoid** (unless with a variable, of course), we also need to assign **associative** and **neutral** to analogous assertions. So the earlier example

```

1 \instantiate{intmonoid}{monoid}{\mathbb{Z}_{+,0}}[
2   universe = Int ,
3   op = addition ,
4   unit = zero
5 ]

```

<sup>8</sup>EDNOTE: MK: reference



...will not work anymore. We now need to give assertions that addition is associative and that zero is a unit with respect to addition.<sup>2</sup>

The `stex-proof` package supplies macros and environment that allow to annotate the structure of mathematical proofs in  $\text{\LaTeX}$  document. This structure can be used by MKM systems for added-value services, either directly from the  $\text{\LaTeX}$  sources, or after translation.

We will go over the general intuition by way of a running example:

```

1 \begin{sproof}[id=simple-proof]
2   {We prove that  $\sum_{i=1}^n 2i-1 = n^2$  by induction over  $n$ }
3   \begin{spfcases}{For the induction we have to consider three cases:}
4     \begin{spfcase}{ $n=1$ }
5       \begin{spfstep}[type=inline] then we compute  $1=1^2$  \end{spfstep}
6     \end{spfcase}
7     \begin{spfcase}{ $n=2$ }
8       \begin{spfcomment}[type=inline]
9         This case is not really necessary, but we do it for the
10        fun of it (and to get more intuition).
11      \end{spfcomment}
12      \begin{spfstep}[type=inline] We compute  $1+3=2^2=4$ . \end{spfstep}
13    \end{spfcase}
14    \begin{spfcase}{ $n>1$ }
15      \begin{spfstep}[type=assumption,id=ind-hyp]
16        Now, we assume that the assertion is true for a certain  $k \geq 1$ ,
17        i.e.  $\sum_{i=1}^k (2i-1) = k^2$ .
18      \end{spfstep}
19      \begin{spfcomment}
20        We have to show that we can derive the assertion for  $n=k+1$  from
21        this assumption, i.e.  $\sum_{i=1}^{k+1} (2i-1) = (k+1)^2$ .
22      \end{spfcomment}
23      \begin{spfstep}
24        We obtain  $\sum_{i=1}^{k+1} (2i-1) = \sum_{i=1}^k (2i-1) + 2(k+1) - 1$ 
25        \spfjust[method=arith:split-sum]{by splitting the sum}.
26      \end{spfstep}
27      \begin{spfstep}
28        Thus we have  $\sum_{i=1}^{k+1} (2i-1) = k^2 + 2k + 1$ 
29        \spfjust[method=fertilize]{by inductive hypothesis}.
30      \end{spfstep}
31      \begin{spfstep}[type=conclusion]
32        We can \spfjust[method=simplify]{simplify} the right-hand side to
33         $(k+1)^2$ , which proves the assertion.
34      \end{spfstep}
35    \end{spfcase}
36    \begin{spfstep}[type=conclusion]
37      We have considered all the cases, so we have proven the assertion.
38    \end{spfstep}
39  \end{spfcases}
40 \end{sproof}

```

This yields the following result:

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

<sup>2</sup>Of course,  $\text{\LaTeX}$  can not check that the assertions are the “correct” ones – but if the assertions (both in monoid as well as those for addition and zero) are properly marked up, MMT can. **TODO: should**

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

**spproof** The **spproof** 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 **spfststep**, **spfstcomment**, and **spfstcases** environments that are used to markup the proof steps.

---

**\spfstidea** The **\spfstidea** macro allows to give a one-paragraph description of the proof idea.

---

**\spfstsketch** For one-line proof sketches, we use the **\spfstsketch** macro, which takes the same optional argument as **spproof** and another one: a natural language text that sketches the proof.

**spfststep** 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.

|   |  |
|---|--|
| <hr/> <hr/> <code>\spfjust</code>       | This evidence is marked up with the <code>\spfjust</code> macro in the <code>stex-proofs</code> 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 <code>KeyVal</code> argument, which can have the <code>method</code> 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).   |
| <hr/> <hr/> <code>\premise</code>       | The <code>\premise</code> macro allows to mark up part of the text as reference to an assertion that is used in the argumentation. In the running example we have used the <code>\premise</code> macro to identify the inductive hypothesis.   |
| <hr/> <hr/> <code>\justarg</code>       | <p>The <code>\justarg</code> macro is very similar to <code>\premise</code> 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 <code>\premise</code>. 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 <code>\justarg</code> macro.</p> <p>Note that both <code>\premise</code> and <code>\justarg</code> can be used with an empty second argument to mark up premises and arguments that are not explicitly mentioned in the text.</p> |
| <code>subproof</code>                   | The <code>spfcases</code> environment is used to mark up a subproof. This environment takes an optional <code>KeyVal</code> argument for semantic annotations and a second argument that allows to specify an introductory comment (just like in the <code>proof</code> environment). The <code>method</code> key can be used to give the name of the proof method executed to make this subproof.   |
| <code>spfcases</code>                   | The <code>spfcases</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.   |
| <code>spfcase</code>                    | The content of a <code>spfcases</code> environment are a sequence of case proofs marked up in the <code>spfcase</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>spfcase</code> environment is the same as that of a <code>sproof</code> , i.e. <code>spfsteps</code> , <code>spfcmmnts</code> , and <code>spfcases</code> environments.  |
| <hr/> <hr/> <code>\spfcasesketch</code> | <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.   |
| <code>spfcmmment</code>                 | The <code>spfcmmment</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> .   |

---

---

**`\sproofend`**

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:

The `stex-proofs` package provides the `\sproofend` macro for this.

---

---

**`\sProofEndSymbol`**

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

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

## Chapter 6

# Highlighting and Presentation Customizations

The environments starting with `s` (i.e. `smodule`, `sassertion`, `sexample`, `sdefinition`, `sparagraph` and `sproof`) by default produce no additional output whatsoever (except for the environment content of course). Instead, the document that uses them (whether directly or e.g. via `\inputref`) can decide how these environments are supposed to look like.

The `stexthm` package defines some default customizations that can be used, but of course many existing  $\text{\LaTeX}$  templates come with their own `definition`, `theorem` and similar environments that authors are supposed (or even required) to use. Their concrete syntax however is usually not compatible with all the additional arguments that  $\text{\LaTeX}$  allows for semantic information.

Therefore we introduced the separate environments `sdefinition` etc. instead of using `definition` directly. We allow authors to specify how these environments should be styled via the commands `stexpatch*`.

---

```
\stexpatchmodule
\stexpatchdefinition
\stexpatchassertion
\stexpatchexample
\stexpatchparagraph
\stexpatchproof
```

---

All of these commands take one optional and two proper arguments, i.e.

```
\stexpatch*{<type>}{<begin-code>}{<end-code>}.
```

After  $\text{\LaTeX}$  reads and processes the optional arguments for these environments, (some of) their values are stored in the macros `\s*<field>` (i.e. `sexampleid`, `\sassertionname`, etc.). It then checks for all the values `<type>` in the `type=`-list, whether an `\stexpatch*{<type>}` for the current environment has been called. If it finds one, it uses the patches `<begin-code>` and `<end-code>` to mark up the current environment. If no patch for (any of) the type(s) is found, it checks whether and `\stexpatch*` was called without optional argument.

For example, if we want to use a predefined `theorem` environment for `sassertions` with `type=theorem`, we can do

```
1 \stexpatchassertion[theorem]{\begin{theorem}}{\end{theorem}}
```

...or, rather, since e.g. `theorem`-like environments defined using `amsthm` take an optional title as argument, we can do:

```
1 \stexpatchassertion[theorem]
2   {\ifx\sassertiontitle\@empty
3     \begin{theorem}
```

```

4   \else
5     \begin{theorem}[\sassertiontitle]
6   \fi}
7 {\end{theorem}}

```

Or, if we want *all kinds of sdefinitions* to use a predefined definition-environment irrespective of their type=, then we can issue the following customization patch:

```

1 \stexpatchdefinition
2 {\ifx\sdefinitiontitle\@empty
3   \begin{definition}
4   \else
5     \begin{definition}[\sdefinitiontitle]
6   \fi}
7 {\end{definition}}

```

---

`\compemph`  
`\varemp`  
`\symrefemph`  
`\defemph`

---

Apart from the environments, we can control how  $\text{\TeX}$  highlights variables, notation components, `\symrefs` and `\definiendums`, respectively.

To do so, we simply redefine these four macros. For example, to highlight notation components (i.e. everything in a `\comp`) in blue, as in this document, we can do `\def\compemph#1{\textcolor{blue}{#1}}`. By default, `\compemph` et al do nothing.

---

`\compemph@uri`  
`\varemp@uri`  
`\symrefemph@uri`  
`\defemph@uri`

---

For each of the four macros, there exists an additional macro that takes the full URI of the relevant symbol currently being highlighted as a second argument. That allows us to e.g. use pdf tooltips and links. For example, this document uses<sup>9</sup>

```

1 \protected\def\symrefemph@uri#1#2{
2   \pdftooltip{
3     \srefsymuri{#2}{\symrefemph{#1}}
4   }{
5     URI:~\detokenize{#2}
6   }
7 }

```

By default, `\compemph@uri` is simply defined as `\compemph{#1}` (analogously for the other three commands).

## Chapter 7

# Additional Packages

### 7.1 Tikzinput: Treating TIKZ code as images

---

**image**

---

The behavior of the `ikzinput` package is determined by whether the `image` option is given. If it is not, then the `tikz` package is loaded, all other options are passed on to it and `\tikzinput{⟨file⟩}` inputs the TIKZ file `⟨file⟩.tex`; if not, only the `graphicx` package is loaded and `\tikzinput{⟨file⟩}` loads an image file `⟨file⟩.⟨ext⟩` generated from `⟨file⟩.tex`.

The selective input functionality of the `tikzinput` package assumes that the TIKZ pictures are externalized into a standalone picture file, such as the following one

```
1 \documentclass{standalone}
2 \usepackage{tikz}
3 \usetikzpackage{...}
4 \begin{document}
5   \begin{tikzpicture}
6     ...
7   \end{tikzpicture}
8 \end{document}
```

The `standalone` class is a minimal  $\text{\LaTeX}$  class that when loaded in a document that uses the `standalone` package: the preamble and the `document` environment are disregarded during loading, so they do not pose any problems. In effect, an `\input` of the file above only sees the `tikzpicture` environment, but the file itself is standalone in the sense that we can run  $\text{\LaTeX}$  over it separately, e.g. for generating an image file from it.

---

**\tikzinput**  
**\ctikzinput**

---

This is exactly where the `tikzinput` package comes in: it supplies the `\tikzinput` macro, which – depending on the `image` option – either directly inputs the TIKZ picture (source) or tries to load an image file generated from it.

Concretely, if the `image` option is not set for the `tikzinput` package, then `\tikzinput[⟨opt⟩]{⟨file⟩}` disregards the optional argument `⟨opt⟩` and inputs `⟨file⟩.tex` via `\input` and resizes it to as specified in the `width` and `height` keys. If it is, `\tikzinput[⟨opt⟩]{⟨file⟩}` expands to `\includegraphics[⟨opt⟩]{⟨file⟩}`.

`\ctikzinput` is a version of `\tikzinput` that is centered.

---

`\mhtikzinput`  
`\cmhtikzinput`

---

`\mhtizkinput` is a variant of `\tikzinput` that treats its file path argument as a relative path in a math archive in analogy to `\inputref`. To give the archive path, we use the `mhrepos=` key. Again, `\cmhtizkinput` is a version of `\mhtikzinput` that is centered.

---

`\libusetikzlibrary`

---

Sometimes, we want to supply archive-specific TIKZ libraries in the `lib` folder of the archive or the `meta-inf/lib` of the archive group. Then we need an analogon to `\libinput` for `\usetikzlibrary`. The `stex-tikzinput` package provides the `libusetikzlibrary` for this purpose.

## 7.2 Modular Document Structuring

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

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  $\text{\TeX}$  sources, or after translation. Currently, trans-document referencing provided by this package can only be used in the  $\text{\TeX}$  collection.

DAG models of documents allow to replace the “Copy and Paste” in the source document with a label-and-reference model where document are shared in the document source and the formatter does the copying during document formatting/presentation.

The `document-structure` package accepts 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> |

**sfragment** The structure of the document is given by nested `sfragment` environments. In the  $\text{\LaTeX}$  route, the `sfragment` environment is flexibly mapped to sectioning commands, inducing the proper sectioning level from the nesting of `sfragment` environments. Correspondingly, the `sfragment` environment takes an optional key/value argument for metadata followed by a regular argument for the (section) title of the sfragment. The optional metadata argument has the keys `id` for an identifier, `creators` and `contributors` for the Dublin Core metadata [DCM03]. The option `short` allows to give a short title for the generated section. If the title contains semantic macros, they need to be protected by `\protect`<sup>10</sup>, and we need to give the `loadmodules` key it needs no value. For instance we would have

```

1 \begin{smodule}{foo}
2   \symdef{bar}{B^a_r}
3   ...
4   \begin{sfragment}[id=sec.barderiv,loadmodules]
5     {Introducing $\protect\bar$ Derivations}
```

---

<sup>10</sup>EdNOTE: MK: still?



$\TeX$  automatically computes the sectioning level, from the nesting of `sfragment` environments.

But sometimes, we want to skip levels (e.g. to use a `\subsection*` as an introduction for a chapter).

**blindfragment** Therefore the `document-structure` package provides a variant `blindfragment` 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 `blindfragment` environment is useful e.g. for creating frontmatter at the correct level. The example below shows a typical setup for the outer document structure of a book with parts and chapters.

```

1 \begin{document}
2 \begin{blindfragment}
3 \begin{blindfragment}
4 \begin{frontmatter}
5 \maketitle\newpage
6 \begin{sfragment}{Preface}
7 ... <<preface>> ...
8 \end{sfragment}
9 \clearpage\setcounter{tocdepth}{4}\tableofcontents\clearpage
10 \end{frontmatter}
11 \end{blindfragment}
12 ... <<introductory remarks>> ...
13 \end{blindfragment}
14 \begin{sfragment}{Introduction}
15 ... <<intro>> ...
16 \end{sfragment}
17 ... <<more chapters>> ...
18 \bibliographystyle{alpha}\bibliography{kwarc}
19 \end{document}

```

Here we use two levels of `blindfragment`:

- The outer one groups the introductory parts of the book (which we assume to have a sectioning hierarchy topping at the part level). This `blindfragment` makes sure that the introductory remarks become a “chapter” instead of a “part”.
- The inner one groups the frontmatter<sup>3</sup> and makes the preface of the book a section-level construct.<sup>11</sup>

---

**\skipfragment** The `\skipfragment` “skips an `sfragment`”, 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 `\skipfragment`.

---

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

<sup>11</sup>EDNOTE: MK: We need a substitute for the “Note that here the `display=flow` on the `sfragment` environment prevents numbering as is traditional for prefaces.”

---

`\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 `sfragment` environment, where we do not know which sectioning level we will end up.

---

`\prematurestop`  
`\afterprematurestop`

---

For prematurely stopping the formatting of a document, `STEX` provides the `\prematurestop` macro. It can be used everywhere in a document and ignores all input after that – backing out of the `sfragment` 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.

`\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].

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 `STEX` preamble of the course notes file.

---

`\setSGvar`  
`\useSGvar`

---

`\setSGvar{<vname>}{<text>}` to set the global variable `<vname>` to `<text>` and `\useSGvar{<vname>}` to reference it.

---

`\ifSGvar`

---

With `\ifSGvar` we can test for the contents of a global variable: the macro call `\ifSGvar{<vname>}{<val>}{<text>}` tests the content of the global variable `<vname>`, only if (after expansion) it is equal to `<val>`, the conditional text `<text>` is formatted.

## 7.3 Slides and Course Notes

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

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 `STEX` 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.

---

slides  
notes  
sectocframes  
frameimages  
fiboxed

---

The notesslides class takes a variety of class options:

- The options `slides` and `notes` switch between slides mode and notes mode (see Section ??).
- If the option `sectocframes` is given, then for the `sfragments`, special frames with the `sfragment` title (and number) are generated.
- If the option `frameimages` is set, then slide mode also shows the `\frameimage`-generated frames (see section ??). If also the `fiboxed` option is given, the slides are surrounded by a box.

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



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

By interleaving the `frame` and `note` environments, we can build course notes as shown here:

```

1 \ifnotes\maketitle\else
2 \frame[noframenumbering]\maketitle\fi
3
4 \begin{note}
5   We start this course with ...
6 \end{note}
7
8 \begin{frame}
9   \frametitle{The first slide}
10  ...
11 \end{frame}
12 \begin{note}
13   ... and more explanatory text
14 \end{note}
15
16 \begin{frame}
17   \frametitle{The second slide}
18   ...
19 \end{frame}
20 ...

```

---

`\ifnotes`

---

Note the use of the `\ifnotes` conditional, which allows different treatment between `notes` and `slides` mode – manually setting `\notestru` or `\notesfalse` is strongly discouraged however.

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



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).

---

#### `\inputref*`

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: `\inputref*{foo}` is equivalent to `\begin{note}\inputref{foo}\end{note}`.

`nexample`, `nsproof`, `nassertion`

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 `nfragment`, `ndefinition`, `nexample`, `nsproof`, and `nassertion` environments.

---

#### `\setslidelogo`

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

---

#### `\setsource`

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.

---

#### `\setlicensing`

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.

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 `STEX` notes.

---

`\frameimage`  
`\mhframeimage`

---

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.

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

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

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

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

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

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

---

`\textwarning`

---

The `\textwarning` macro generates a warning sign: 

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:

```
1 \excursion{founif}{../ex/founif}{We will cover first-order unification in}
2 ...
3 \begin{appendix}\printexcursions\end{appendix}
```

---

`\excursion`

---

The `\excursion{⟨ref⟩}{⟨path⟩}{⟨text⟩}` is syntactic sugar for

```
1 \begin{nparagraph}[title=Excursion]
2   \activateexcursion{founif}{../ex/founif}
3   We will cover first-order unification in \sref{founif}.
4 \end{nparagraph}
```

---

`\activateexcursion`  
`\printexcursion`  
`\excursionref`

---

Here `\activateexcursion{⟨path⟩}` augments the `\printexcursions` macro by a call `\inputref{⟨path⟩}`. In this way, the `\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{⟨label⟩}` for that.

---

`\excursiongroup`

Finally, we usually want to put the excursions into an `sfragment` environment and add an introduction, therefore we provide the a variant of the `\printexcursions` macro: `\excursiongroup[id=<id>,intro=<path>]` is equivalent to

```
1 \begin{note}
2 \begin{sfragment}[id=<id>]{Excursions}
3   \inputref{<path>}
4   \printexcursions
5 \end{sfragment}
6 \end{note}
```



When option `book` which uses `\pagestyle{headings}` is given and semantic macros are given in the `sfragment` 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 `document-structure` package.

## 7.4 Representing Problems and Solutions

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

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

---

`solutions`  
`notes`  
`hints`  
`gnotes`  
`pts`  
`min`  
`boxed`  
`test`

---

The `problem` package takes the options `solutions` (should solutions be output?), `notes` (should the problem notes be presented?), `hints` (do we give the hints?), `gnotes` (do we show grading notes?), `pts` (do we display the points awarded for solving the problem?), `min` (do we display the estimated minutes for problem soling). If theses are specified, then the corresponding auxiliary parts of the problems are output, otherwise, they remain invisible.

The `boxed` option specifies that problems should be formatted in framed boxes so that they are more visible in the text. Finally, the `test` 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.

`problem` The main environment provided by the `problempackage` 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.

---

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

## Example 40

Input:

```

1 \documentclass{article}
2 \usepackage[solutions,hints,pts,min]{problem}
3 \begin{document}
4   \begin{sproblem}[id=elephants,pts=10,min=2,title=Fitting Elephants]
5     How many Elephants can you fit into a Volkswagen beetle?
6     \begin{hint}
7       Think positively, this is simple!
8     \end{hint}
9     \begin{exnote}
10      Justify your answer
11    \end{exnote}
12 \begin{solution}[for=elephants,height=3cm]
13   Four, two in the front seats, and two in the back.
14 \begin{gnote}
15   if they do not give the justification deduct 5 pts
16 \end{gnote}
17 \end{solution}
18 \end{sproblem}
19 \end{document}

```

Output:

**Problem 7.4.1 (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.

---

**Grading:** if they do not give the justification deduct 5 pts

---

**solution** The `solution` environment can be to specify a solution to a problem. If the package option `solutions` 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).

**hint,exnote,gnote** 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. The `gnote` (grading notes) environment can be used to document situations that may arise in grading.

---

**\startsolutions**  
**\stopsolutions**

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.

---

---

`\ifsolutions`

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.

`mcb` Multiple choice blocks can be formatted using the `mcb` environment, in which single choices are marked up with `\mcc` macro.

---

---

`\mcc`

`\mcc[⟨keyvals⟩]{⟨text⟩}` takes an optional key/value argument `⟨keyvals⟩` for choice meta-data and a required argument `⟨text⟩` for the proposed answer text. The following keys are supported

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

If we start the solutions, then we get

#### Example 41

Input:

```
1 \startsolutions
2 \begin{sproblem}[title=Functions,name=functions1]
3   What is the keyword to introduce a function definition in python?
4   \begin{mcb}
5     \mcc[T]{def}
6     \mcc[F,feedback=that is for C and C++]{function}
7     \mcc[F,feedback=that is for Standard ML]{fun}
8     \mcc[F,Ftext=Nooooooooo,feedback=that is for Java]{public static void}
9   \end{mcb}
10 \end{sproblem}
```

Output:

##### Problem 7.4.2 (Functions)

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

- ☐ `def`  
(**true**)
- ☐ `function`  
(**false**) (*that is for C and C++*)
- ☐ `fun`  
(**false**) (*that is for Standard ML*)
- ☐ `public static void`  
(**false**) (*that is for Java*)

---

<sup>12</sup>EdNOTE: MK: that did not work!



### Example 42

Input:

```
1 \stopsolutions
2 \begin{sproblem}[title=Functions,name=functions1]
3   What is the keyword to introduce a function definition in python?
4   \begin{mcb}
5     \mcc[T]{def}
6     \mcc[F,feedback=that is for C and C++){function}
7     \mcc[F,feedback=that is for Standard ML]{fun}
8     \mcc[F,Ftext=Noooooooooooo,feedback=that is for Java]{public static void}
9   \end{mcb}
10 \end{sproblem}
```

Output:

#### Problem 7.4.3 (Functions)

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

- ☐ def  
(true)
- ☐ function  
(false) (that is for C and C++)
- ☐ fun  
(false) (that is for Standard ML)
- ☐ public static void  
(false) (that is for Java)

### \includeproblem

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.

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` options are set. This allows to give students hints about the estimated time and the points to be awarded.

## 7.5 Homeworks, Quizzes and Exams

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 `roblem` package. It is designed to be compatible with `problems.sty`, and inherits some of the functionality.

|                             |  |
|-----------------------------|--|
| <code>solutions</code>      | The <code>wexam</code> package and class take the options <code>solutions</code> , <code>notes</code> , <code>hints</code> , <code>gnotes</code> , <code>pts</code> , <code>min</code> , and <code>boxed</code> that are just passed on to the <code>problems</code> package (cf. its documentation for a description of the intended behavior).   |
| <code>notes</code>          |  |
| <code>hints</code>          |  |
| <code>gnotes</code>         |  |
| <code>pts</code>            |  |
| <code>min</code>            |  |
| <hr/>                       |  |
| <code>assignment</code>     | This package supplies the <code>assignment</code> environment that groups problems into assignment sheets. It takes an optional <code>KeyVal</code> argument with the keys <code>number</code> (for the assignment number; if none is given, 1 is assumed as the default or — in multi-assignment documents — the ordinal of the <code>assignment</code> environment), <code>title</code> (for the assignment title; this is referenced in the title of the assignment sheet), <code>type</code> (for the assignment type; e.g. “quiz”, or “homework”), <code>given</code> (for the date the assignment was given), and <code>due</code> (for the date the assignment is due). |
| <code>number</code>         |  |
| <code>title</code>          |  |
| <code>type</code>           |  |
| <code>given</code>          |  |
| <code>due</code>            |  |
| <code>multiple</code>       | Furthermore, the <code>hwexam</code> package takes the option <code>multiple</code> 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.).   |
| <code>test</code>           | Finally, there is the option <code>test</code> that modifies the behavior to facilitate formatting tests. Only in <code>test</code> mode, the macros <code>\testspace</code> , <code>\testnewpage</code> , and <code>\testemptypage</code> have an effect: they generate space for the students to solve the given problems. Thus they can be left in the $\text{\LaTeX}$ source.  |
| <code>\testspace</code>     | <code>\testspace</code> takes an argument that expands to a dimension, and leaves vertical space accordingly. <code>\testnewpage</code> makes a new page in <code>test</code> mode, and <code>\testemptypage</code> generates an empty page with the cautionary message that this page was intentionally left empty.   |
| <code>\testnewpage</code>   |  |
| <code>\testemptypage</code> |  |
| <code>testheading</code>    | Finally, the <code>\testheading</code> takes an optional keyword argument where the keys <code>duration</code> specifies a string that specifies the duration of the test, <code>min</code> specifies the equivalent in number of minutes, and <code>reqpts</code> the points that are required for a perfect grade.   |
| <code>duration</code>       |  |
| <code>min</code>            |  |
| <code>reqpts</code>         |  |
|                             |  |
|                             | <pre>1 \title{320101 General Computer Science (Fall 2010)} 2 \begin{testheading}[duration=one hour,min=60,reqpts=27] 3   Good luck to all students! 4 \end{testheading}</pre>  |

Will result in

Name:

Matriculation Number:

## 320101 General Computer Science (Fall 2010)

2022-05-07

**You have one hour (sharp) for the test;**

Write the solutions to the sheet.

The estimated time for solving this exam is 60 minutes, leaving you 0 minutes for revising your exam.

You can reach 40 points if you solve all problems. You will only need 27 points for a perfect score, i.e. 13 points are bonus points.

*You have ample time, so take it slow and avoid rushing to mistakes!*

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

|         | To be used for grading, do not write here |       |       |     |     |     |     |     |     |     |     |       |
|---------|---|-------|-------|-----|-----|-----|-----|-----|-----|-----|-----|-------|
| prob.   | 7.4.1                                     | 7.4.2 | 7.4.3 | 1.1 | 2.1 | 2.2 | 2.3 | 3.1 | 3.2 | 3.3 | Sum | grade |
| total   | 10  |       |       | 4   | 4   | 6   | 6   | 4   | 4   | 2   | 40  |       |
| reached |   |       |       |     |     |     |     |     |     |     |     |       |

good luck

EdN:13

13

### \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.

---

<sup>13</sup>EDNOTE: MK: The first three “problems” come from the stex examples above, how do we get rid of this?

## Part II

# Documentation

# Chapter 8

## sTeX-Basics

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

### 8.1 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>*.

#### 8.1.1 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 `⟨content⟩` with

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

### 8.1.2 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`.

### 8.1.3 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 `⟨cs⟩` throw an error, indicating that it is only allowed in the context of `⟨environments⟩`.

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

---

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

---

## Chapter 9

# STEX-MathHub

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

### 9.1 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. |
|--------------------------------|---|

---

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

---

### 9.1.2 MathHub Archives

---

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

---

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

In all four 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><br>Change the current repository to <code>{&lt;repository-name&gt;}</code> (or not, if <code>{&lt;repository-name&gt;}</code> is empty), and passes its ID on to <code>{&lt;code&gt;}</code> as <code>#1</code> . Switches back to the previous repository after executing <code>{&lt;code&gt;}</code> . |
|-------------------------------------|---|

---



### 9.1.3 Using Content in Archives

|  |  |
|--|--|
| <hr/> <hr/> <code>\mhp</code> <hr/>  | <code>\mhp{&lt;archive-ID&gt;}{&lt;filename&gt;}</code>  |
|  | Expands to the full path of file <code>&lt;filename&gt;</code> in repository <code>&lt;archive-ID&gt;</code> . Does not check whether the file or the repository exist.  |
| <hr/> <hr/> <code>\inputref</code> <hr/> <code>\mhinput</code> <hr/>                     | <code>\inputref[&lt;archive-ID&gt;]{&lt;filename&gt;}</code><br>Both <code>\input</code> the file <code>&lt;filename&gt;</code> in archive <code>&lt;archive-ID&gt;</code> (relative to the <code>source-</code> subdirectory). <code>\mhinput</code> does so directly. <code>\inputref</code> does so within an <code>\begingroup... \endgroup-</code> block, and skips it in <code>html-</code> mode, 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[&lt;archive-ID&gt;]{&lt;filename&gt;}</code><br>Adds a <code>.bib</code> -file <code>&lt;filename&gt;</code> in archive <code>&lt;archive-ID&gt;</code> (relative to the top-directory of the archive!).   |
| <hr/> <hr/> <code>\libinput</code> <hr/>   | <code>\libinput{&lt;filename&gt;}</code><br>Inputs <code>&lt;filename&gt;.tex</code> from the <code>lib</code> folders in the current archive and the <code>meta-inf-</code> archive of the current archive group(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[&lt;args&gt;]{&lt;filename&gt;}</code><br>Like <code>\libinput</code> , but looks for <code>.sty</code> -files and calls <code>\usepackage[&lt;meta{args}&gt;]{&lt;Arg{filename}&gt;}</code> instead of <code>\input</code> .<br>Throws an error, if none or more than one suitable package file is found.  |
| <hr/> <hr/> <code>\mhgraphics</code> <hr/> <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> <hr/> <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 10

## STEX-References

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

### 10.1 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**

#### 10.1.1 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>*.

### 10.1.2 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 11

## STEX-Modules

This sub package contains code related to Modules

### 11.1 Macros and Environments

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

---

---

`\c_stex_module_<URI>_prop`

A property list with the following fields:

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

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

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

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

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

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

**meta** the metatheory of the module.

---

---

`\c_stex_module_<URI>_code`

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

---

---

`\c_stex_module_<URI>_constants`

The names of all constants declared in the module

---

---

`\c_stex_module_<URI>_constants`

The full URIs of all modules imported in this module

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

---

**\stex\_modules\_current\_namespace:**

---

Computes the current namespace as follows:

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

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

### 11.1.1 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** `\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 12

# STEX-Module Inheritance

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

### 12.1 Macros and Environments

#### 12.1.1 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.



|   |   |
|---|---|
| <hr/> <hr/> <code>\stex_file_in_smsmode:nn</code> | <code>\stex_in_smsmode:nn</code> $\{\langle filename \rangle\}$ $\{\langle code \rangle\}$  |
|   | Executes $\langle code \rangle$ in SMS mode, followed by the content of $\langle filename \rangle$ . $\langle code \rangle$ 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. |

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

### 12.1.2 Imports and Inheritance

|  |   |
|--|---|
| <hr/> <hr/> <code>\importmodule</code> | <code>\importmodule</code> [ $\langle archive-ID \rangle$ ] $\{\langle module-path \rangle\}$   |
|  | Imports a module by reading it from a file and “activating” it. $\text{\texttt{S\TeX}}$ determines the module and its containing file by passing its arguments on to <code>\stex_import_module_path:nn</code> . |

|                                     |   |
|-------------------------------------|---|
| <hr/> <hr/> <code>\usemodule</code> | <code>\importmodule</code> [ $\langle archive-ID \rangle$ ] $\{\langle module-path \rangle\}$   |
|                                     | Like <code>\importmodule</code> , 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 13

## STEX-Symbols

Code related to symbol declarations and notations

### 13.1 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>.<br/> 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>s (integer string; number of associative arguments),</li> </ul> |
| <hr/> <hr/> <code>\stex_all_symbols:n</code>  | Iterates over all currently available symbols. Requires two <code>\seq_map_break:</code> to break fully.   |
| <hr/> <hr/> <code>\stex_get_symbol:n</code>   | Computes the full URI of a symbol from a macro argument, e.g. the macro name, the macro itself, the full URI...  |
| <hr/> <hr/> <code>\notation</code>            | <p><code>\notation[&lt;args&gt;]{&lt;symbol&gt;}{&lt;notations<sup>+</sup>&gt;}</code><br/> 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><br/> Implements the core functionality of <code>\notation</code>, and is called by <code>\notation</code> and <code>\symdef</code>.<br/> Ultimately stores the notation in the property list<br/> <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>s (sequence of integer strings)</li> </ul>   |
| <hr/> <hr/> <code>\symdef</code>              | <p><code>\symdef[&lt;args&gt;]{&lt;symbol&gt;}{&lt;notations<sup>+</sup>&gt;}</code><br/> Combines <code>\symdecl</code> and <code>\notation</code> by introducing a new symbol and assigning a new notation for it.</p>   |

# Chapter 14

## STEX-Terms

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

### 14.1 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/> <code>\infprec</code><br><code>\neginfprec</code> <hr/>  | Maximal and minimal notation precedences.   |
| <hr/> <code>\dobrackets</code> <hr/>   | <code>\dobrackets {⟨body⟩}</code><br><br>Puts <i>⟨body⟩</i> in parentheses; scaled if in display mode unscaled otherwise. Uses the current $\text{\S}$ TEX brackets (by default ( and )), which can be changed temporarily using <code>\withbrackets</code> .   |
| <hr/> <code>\withbrackets</code> <hr/>   | <code>\withbrackets ⟨left⟩ ⟨right⟩ {⟨body⟩}</code><br><br>Temporarily (i.e. within <i>⟨body⟩</i> ) sets the brackets used by $\text{\S}$ TEX for automated bracketing (by default ( and )) to <i>⟨left⟩</i> and <i>⟨right⟩</i> .<br>Note that <i>⟨left⟩</i> and <i>⟨right⟩</i> need to be allowed after <code>\left</code> and <code>\right</code> in display-mode.   |
| <hr/> <code>\stex_term_custom:nn</code> <hr/>  | <code>\stex_term_custom:nn{⟨URI⟩}{⟨args⟩}</code><br><br>Implements custom one-time notation. Invoked by <code>\stex_invoke_symbol:n</code> in text mode, or if followed by <code>*</code> in math mode, or whenever followed by <code>!</code> .  |
| <hr/> <code>\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> <hr/> | <code>\comp{⟨args⟩}</code><br><br>Marks <i>⟨args⟩</i> as a notation component of the current symbol for highlighting, linking, etc.<br><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><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/> <code>\STEXinvisible</code> <hr/>  | Exports its argument as OMDoc (invisible), but does not produce PDF output. Useful e.g. for semantic macros that take arguments that are not part of the symbolic notation.   |
| <hr/> <code>\ellipses</code> <hr/>   | TODO  |

## Chapter 15

# TeX-Structural Features

Code related to structural features

### 15.1 Macros and Environments

#### 15.1.1 Structures

`mathstructure` TODO

## Chapter 16

# sTeX-Statements

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

### 16.1 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 17

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

## Chapter 18

# sTeX-Metatheory

### 18.1 Symbols

**Part III**  
**Extensions**

## Chapter 19

# Tikzinput: Treating TIKZ code as images

### 19.1 Macros and Environments

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

## Chapter 20

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

## Chapter 21

# NotesSlides – Slides and Course Notes

## Chapter 22

# `problem.sty`: An Infrastructure for formatting Problems

## Chapter 23

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



**Part IV**  
**Implementation**

## Chapter 24

# $\text{\TeX}$ -Basics Implementation

### 24.1 The $\text{\TeX}$ Document 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
8 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
9 \ProcessOptions
10
11 \bool_set_true:N \c_stex_document_class_bool
12
13 \RequirePackage{stex}
14
15 \stex_html_backend:TF {
16   \LoadClass{article}
17 }{
18   \LoadClass[border=1px,varwidth,crop=false]{standalone}
19   \setlength\textwidth{15cm}
20 }
21 \RequirePackage{standalone}
22
23
24 \clist_if_empty:NT \c_stex_languages_clist {
25   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
26   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
27   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
28   \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
29     \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
30       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
```

```

31     }
32   }
33   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
34   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
35     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
36     \prop_if_in:NoT \c_stex_languages_prop \l_tmpa_str {
37       \stex_debug:nn{language} {Language~\l_tmpa_str~
38         inferred~from~file~name}
39     }
40   }
41 }
42 }
43 </cls>

```

## 24.2 Preliminaries

```

44 <*package>
45
46 %%%%%%%%% basics.dtx %%%%%%%%%
47
48 \RequirePackage{expl3,l3keys2e,ltxcmds}
49 \ProvidesExplPackage{stex}{2022/03/03}{3.1.0}{sTeX package}
50
51 \bool_if_exist:NF \c_stex_document_class_bool {
52   \bool_set_false:N \c_stex_document_class_bool
53   \RequirePackage{standalone}
54 }
55
56 \message{^^J
57   *****^^J
58   *~This~is~sTeX~version~3.1.0*~^^J
59   *****^^J
60 ^^J}
61
62 %\RequirePackage{morewrites}
63 %\RequirePackage{amsmath}
64

```

Package options:

```

65 \keys_define:nn { stex } {
66   debug      .clist_set:N = \c_stex_debug_clist ,
67   lang       .clist_set:N = \c_stex_languages_clist ,
68   mathhub    .tl_set_x:N  = \mathhub ,
69   usesms     .bool_set:N  = \c_stex_persist_mode_bool ,
70   writesms   .bool_set:N  = \c_stex_persist_write_mode_bool ,
71   image      .bool_set:N  = \c_tikzinput_image_bool ,
72   unknown    .code:n      = {}
73 }
74 \ProcessKeysOptions { stex }

```

**\stex** The sTeX logo:

**\sTeX**

```

75 \RequirePackage{xspace}
76 \protected\def\stex{
77   \@ifundefined{texorpdfstring}{\let\texorpdfstring\@firstoftwo}{

```

```

78 \texorpdfstring{\raisebox{-.5ex}{S\kern-.5ex\TeX}}{sTeX}\xspace
79 }
80 \let\TeX\stex

```

(End definition for `\stex` and `\sTeX`. These functions are documented on page 63.)

## 24.3 Messages and logging

```

81 <@@=stex_log>
    Warnings and error messages
82 \msg_new:nnn{stex}{error/unknownlanguage}{
83   Unknown~language:~#1
84 }
85 \msg_new:nnn{stex}{warning/nomathhub}{
86   MATHHUB~system~variable~not~found~and~no~
87   \detokenize{\mathhub}~value~set!
88 }
89 \msg_new:nnn{stex}{error/deactivated-macro}{
90   The~\detokenize{#1}~command~is~only~allowed~in~#2!
91 }

```

`\stex_debug:nn` A simple macro issuing package messages with subpath.

```

92 \cs_new_protected:Nn \stex_debug:nn {
93   \clist_if_in:NnTF \c_stex_debug_clist { all } {
94     \msg_set:nnn{stex}{debug / #1}{
95       \Debug~#1:~#2\
96     }
97     \msg_none:nn{stex}{debug / #1}
98   }{
99     \clist_if_in:NnT \c_stex_debug_clist { #1 } {
100       \msg_set:nnn{stex}{debug / #1}{
101         \Debug~#1:~#2\
102       }
103       \msg_none:nn{stex}{debug / #1}
104     }
105   }
106 }

```

(End definition for `\stex_debug:nn`. This function is documented on page 63.)

Redirecting messages:

```

107 \clist_if_in:NnTF \c_stex_debug_clist {all} {
108   \msg_redirect_module:nnn{ stex }{ none }{ term }
109 }{
110   \clist_map_inline:Nn \c_stex_debug_clist {
111     \msg_redirect_name:nnn{ stex }{ debug / ##1 }{ term }
112   }
113 }
114
115 \stex_debug:nn{log}{debug~mode~on}

```

## 24.4 HTML Annotations

116 `<@=stex_annotate>`

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

117 `\tl_new:N \l_stex_html_arg_tl`

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

`\_stex_html_checkempty:n`

```
118 \cs_new_protected:Nn \_stex_html_checkempty:n {
119   \tl_set:Nn \l_stex_html_arg_tl { #1 }
120   \tl_if_empty:NT \l_stex_html_arg_tl {
121     \tl_set_eq:NN \l_stex_html_arg_tl \c_stex_html_emptyarg_tl
122   }
123 }
```

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

`\stex_if_do_html_p:` Whether to (locally) produce HTML output

`\stex_if_do_html:TF`

```
124 \bool_new:N \_stex_html_do_output_bool
125 \bool_set_true:N \_stex_html_do_output_bool
126
127 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
128   \bool_if:nTF \_stex_html_do_output_bool
129     \prg_return_true: \prg_return_false:
130 }
```

(End definition for `\stex_if_do_html:TF`. This function is documented on page 63.)

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

```
131 \cs_new_protected:Nn \stex_suppress_html:n {
132   \exp_args:Nne \use:nn {
133     \bool_set_false:N \_stex_html_do_output_bool
134     #1
135   }{
136     \stex_if_do_html:T {
137       \bool_set_true:N \_stex_html_do_output_bool
138     }
139   }
140 }
```

(End definition for `\stex_suppress_html:n`. This function is documented on page 63.)

`\stex_annotate:bnx`

`\stex_annotate_invisible:n`

`\stex_annotate_invisible:nnn`

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

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

```
141 \tl_if_exist:NF\stex@backend{
142   \ifcsname if@rustex\endcsname
143     \def\stex@backend{rustex}
144   \else
145     \ifcsname if@latexml\endcsname
```

```

146     \def\stex@backend{latexml}
147   \else
148     \def\stex@backend{pdflatex}
149   \fi
150 \fi
151 }
152 \input{stex-backend-\stex@backend.cfg}

```

(End definition for `\stex_annotate:nnn`, `\stex_annotate_invisible:n`, and `\stex_annotate_invisible:nnn`. These functions are documented on page 64.)

## 24.5 Babel Languages

```

153 <@@=stex_language>

```

`\c_stex_languages_prop`  
`\c_stex_language_abbrevs_prop`

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

```

154 \exp_args:NNx \prop_const_from_keyval:Nn \c_stex_languages_prop { \tl_to_str:n {
155   en = english ,
156   de = ngerman ,
157   ar = arabic ,
158   bg = bulgarian ,
159   ru = russian ,
160   fi = finnish ,
161   ro = romanian ,
162   tr = turkish ,
163   fr = french
164 }}
165
166 \exp_args:NNx \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop { \tl_to_str:n {
167   english = en ,
168   ngerman = de ,
169   arabic = ar ,
170   bulgarian = bg ,
171   russian = ru ,
172   finnish = fi ,
173   romanian = ro ,
174   turkish = tr ,
175   french = fr
176 }}
177 % todo: chinese simplified (zhs)
178 %       chinese traditional (zht)

```

(End definition for `\c_stex_languages_prop` and `\c_stex_language_abbrevs_prop`. These variables are documented on page 64.)

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

```

179 \cs_new_protected:Nn \stex_set_language:Nn {
180   \str_set:Nx \l_tmpa_str {#2}
181   \prop_get:NoNT \c_stex_languages_prop \l_tmpa_str #1 {
182     \ifx\@onlypreamble\@notprerr
183       \ltx@ifpackageloaded{babel}{
184         \exp_args:No \selectlanguage #1
185       }{}
186     \else
187       \exp_args:No \str_if_eq:nnTF #1 {turkish} {

```

```

188     \RequirePackage[#1,shorthands=:!]{babel}
189   }{
190     \RequirePackage[#1]{babel}
191   }
192   \fi
193 }
194 }
195
196 \clist_if_empty:NF \c_stex_languages_clist {
197   \bool_set_false:N \l_tmpa_bool
198   \clist_clear:N \l_tmpa_clist
199   \clist_map_inline:Nn \c_stex_languages_clist {
200     \str_set:Nx \l_tmpa_str {#1}
201     \str_if_eq:nnT {#1}{tr}{
202       \bool_set_true:N \l_tmpa_bool
203     }
204     \prop_get:NoNTF \c_stex_languages_prop \l_tmpa_str \l_tmpa_str {
205       \clist_put_right:No \l_tmpa_clist \l_tmpa_str
206     } {
207       \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
208     }
209   }
210   \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
211   \bool_if:NTF \l_tmpa_bool {
212     \RequirePackage[\clist_use:Nn \l_tmpa_clist,,shorthands=:!]{babel}
213   }{
214     \RequirePackage[\clist_use:Nn \l_tmpa_clist,]{babel}
215   }
216 }
217
218 \AtBeginDocument{
219   \stex_html_backend:T {
220     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
221     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
222     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
223     \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
224     \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
225       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
226       \stex_debug:nn{basics} {Language~\l_tmpa_str~
227         inferred~from~file~name}
228       \stex_annotate_invisible:nnn{language}{ \l_tmpa_str }{}
229     }
230   }
231 }

```

## 24.6 Persistence

```

232 <@@=stex_persist>
233 \bool_if:NTF \c_stex_persist_mode_bool {
234   \def \stex_persist:n #1 {}
235   \def \stex_persist:x #1 {}
236 }{
237   \bool_if:NTF \c_stex_persist_write_mode_bool {

```

```

238 \iow_new:N \c__stex_persist_iow
239 \iow_open:Nn \c__stex_persist_iow{\jobname.sms}
240 \AtEndDocument{
241   \iow_close:N \c__stex_persist_iow
242 }
243 \cs_new_protected:Nn \stex_persist:n {
244   \tl_set:Nn \l_tmpa_tl { #1 }
245   \regex_replace_all:nnN { \cP\# } { \cO\# } \l_tmpa_tl
246   \exp_args:NNo \iow_now:Nn \c__stex_persist_iow \l_tmpa_tl
247 }
248 \cs_generate_variant:Nn \stex_persist:n {x}
249 }{
250   \def \stex_persist:n #1 {}
251   \def \stex_persist:x #1 {}
252 }
253 }

```

## 24.7 Auxiliary Methods

`\stex_deactivate_macro:Nn`

```

254 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
255   \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
256   \def#1{
257     \msg_error:nnnn{stex}{error/deactivated-macro}{\detokenize{#1}}{#2}
258   }
259 }

```

(End definition for `\stex_deactivate_macro:Nn`. This function is documented on page 64.)

`\stex_reactivate_macro:N`

```

260 \cs_new_protected:Nn \stex_reactivate_macro:N {
261   \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname
262 }

```

(End definition for `\stex_reactivate_macro:N`. This function is documented on page 64.)

`\ignorespacesandpars`

```

263 \protected\def\ignorespacesandpars{
264   \begingroup\catcode13=10\relax
265   \@ifnextchar\par{
266     \endgroup\expandafter\ignorespacesandpars\@gobble
267   }{
268     \endgroup
269   }
270 }
271
272 \cs_new_protected:Nn \stex_copy_control_sequence:NNN {
273   \tl_set:Nx \_tmp_args_tl {\cs_argument_spec:N #2}
274   \exp_args:NNo \tl_remove_all:Nn \_tmp_args_tl \c_hash_str
275   \int_set:Nn \l_tmpa_int {\tl_count:N \_tmp_args_tl}
276
277   \tl_clear:N \_tmp_args_tl
278   \int_step_inline:nn \l_tmpa_int {
279     \tl_put_right:Nx \_tmp_args_tl {\exp_not:n{####}\exp_not:n{##1}}

```



```

280 }
281
282 \tl_set:Nn #3 {\cs_generate_from_arg_count:NNnn #1 \cs_set:Npn}
283 \tl_put_right:Nx #3 { {\int_use:N \l_tmpa_int}{
284   \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
285   \exp_after:wN\exp_after:wN\exp_after:wN {
286     \exp_after:wN #2 \_tmp_args_tl
287   }
288 }}
289 }
290 \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {cNN}
291 \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {NcN}
292 \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {ccN}
293
294 \cs_new_protected:Nn \stex_copy_control_sequence_ii:NNN {
295   \tl_set:Nx \_tmp_args_tl {\cs_argument_spec:N #2}
296   \exp_args:NNo \tl_remove_all:Nn \_tmp_args_tl \c_hash_str
297   \int_set:Nn \l_tmpa_int {\tl_count:N \_tmp_args_tl}
298
299   \tl_clear:N \_tmp_args_tl
300   \int_step_inline:nn \l_tmpa_int {
301     \tl_put_right:Nx \_tmp_args_tl {\exp_not:n{#####}\exp_not:n{##1}}
302   }
303
304   \edef \_tmp_args_tl {
305     \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
306     \exp_after:wN\exp_after:wN\exp_after:wN {
307       \exp_after:wN #2 \_tmp_args_tl
308     }
309   }
310
311   \exp_after:wN \def \exp_after:wN \_tmp_args_tl
312   \exp_after:wN ##\exp_after:wN 1 \exp_after:wN ##\exp_after:wN 2
313   \exp_after:wN { \_tmp_args_tl }
314
315   \edef \_tmp_args_tl {
316     \exp_after:wN \exp_not:n \exp_after:wN {
317       \_tmp_args_tl {####1}{####2}
318     }
319   }
320
321   \tl_set:Nn #3 {\cs_generate_from_arg_count:NNnn #1 \cs_set:Npn}
322   \tl_put_right:Nx #3 { {\int_use:N \l_tmpa_int}{
323     \exp_after:wN\exp_not:n\exp_after:wN{\_tmp_args_tl}
324   }}
325 }
326
327 \cs_generate_variant:Nn \stex_copy_control_sequence_ii:NNN {cNN}
328 \cs_generate_variant:Nn \stex_copy_control_sequence_ii:NNN {NcN}
329 \cs_generate_variant:Nn \stex_copy_control_sequence_ii:NNN {ccN}

```

(End definition for `\ignorespacesandpars`. This function is documented on page 64.)

`\MMTrule`

```

330 \NewDocumentCommand \MMTrule {m m}{
331   \seq_set_split:Nnn \l_tmpa_seq , {#2}
332   \int_zero:N \l_tmpa_int
333   \stex_annotate_invisible:nnn{mmtrule}{scala://#1}{
334     \seq_if_empty:NF \l_tmpa_seq {
335       $\seq_map_inline:Nn \l_tmpa_seq {
336         \int_incr:N \l_tmpa_int
337         \stex_annotate:nnn{arg}{i\int_use:N \l_tmpa_int}{##1}
338       }$
339     }
340   }
341 }
342
343 \NewDocumentCommand \MMTinclude {m}{
344   \stex_annotate_invisible:nnn{import}{#1}{-}
345 }
346
347 \tl_new:N \g_stex_document_title
348 \cs_new_protected:Npn \STEXtitle #1 {
349   \tl_if_empty:NT \g_stex_document_title {
350     \tl_gset:Nn \g_stex_document_title { #1 }
351   }
352 }
353 \cs_new_protected:Nn \stex_document_title:n {
354   \tl_if_empty:NT \g_stex_document_title {
355     \tl_gset:Nn \g_stex_document_title { #1 }
356     \stex_annotate_invisible:n{\noindent
357       \stex_annotate:nnn{doctitle}{-}{ #1 }
358     \par}
359   }
360 }
361 \AtBeginDocument {
362   \let \STEXtitle \stex_document_title:n
363   \tl_if_empty:NF \g_stex_document_title {
364     \stex_annotate_invisible:n{\noindent
365       \stex_annotate:nnn{doctitle}{-}{ \g_stex_document_title }
366     \par}
367   }
368   \let \stex_maketitle:\maketitle
369   \def \maketitle{
370     \tl_if_empty:NF \@title {
371       \exp_args:No \stex_document_title:n \@title
372     }
373     \stex_maketitle:
374   }
375 }
376
377 \cs_new_protected:Nn \stex_par: {
378   \mode_if_vertical:F{
379     \if@minipage\else\if@nobreak\else\par\fi\fi
380   }
381 }
382
383 \end{package}

```

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

## Chapter 25

# STEX -MathHub Implementation

```
384 <*package>
385
386 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
387
388 <@@=stex_path>
389
390 Warnings and error messages
391 \msg_new:nnn{stex}{error/norepository}{
392   No~archive~#1~found~in~#2
393 }
394 \msg_new:nnn{stex}{error/notinarchive}{
395   Not~currently~in~an~archive,~but~\detokenize{#1}~
396   needs~one!
397 }
398 \msg_new:nnn{stex}{error/nofile}{
399   \detokenize{#1}~could~not~find~file~#2
400 }
401 \msg_new:nnn{stex}{error/twofiles}{
402   \detokenize{#1}~found~two~candidates~for~#2
403 }
```

### 25.1 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`

```
402 \cs_new_protected:Nn \stex_path_from_string:Nn {
403   \str_set:Nx \l_tmpa_str { #2 }
404   \str_if_empty:NTF \l_tmpa_str {
405     \seq_clear:N #1
406   }{
407     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
408     \sys_if_platform_windows:T{
409       \seq_clear:N \l_tmpa_tl
```

```

410     \seq_map_inline:Nn #1 {
411       \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
412       \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
413     }
414     \seq_set_eq:NN #1 \l_tmpa_tl
415   }
416   \stex_path_canonicalize:N #1
417 }
418 }
419

```

(End definition for `\stex_path_from_string:Nn`. This function is documented on page 65.)

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

```

420 \cs_new_protected:Nn \stex_path_to_string:NN {
421   \exp_args:Nne \str_set:Nn #2 { \seq_use:Nn #1 / }
422 }
423
424 \cs_new:Nn \stex_path_to_string:N {
425   \seq_use:Nn #1 /
426 }

```

(End definition for `\stex_path_to_string:NN` and `\stex_path_to_string:N`. These functions are documented on page 65.)

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

```

427 \str_const:Nn \c__stex_path_dot_str {.}
428 \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.

```

429 \cs_new_protected:Nn \stex_path_canonicalize:N {
430   \seq_if_empty:NF #1 {
431     \seq_clear:N \l_tmpa_seq
432     \seq_get_left:NN #1 \l_tmpa_tl
433     \str_if_empty:NT \l_tmpa_tl {
434       \seq_put_right:Nn \l_tmpa_seq {}
435     }
436     \seq_map_inline:Nn #1 {
437       \str_set:Nn \l_tmpa_tl { ##1 }
438       \str_if_eq:NNF \l_tmpa_tl \c__stex_path_dot_str {
439         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
440           \seq_if_empty:NNTF \l_tmpa_seq {
441             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
442               \c__stex_path_up_str
443             }
444           }{
445             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
446             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
447               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
448                 \c__stex_path_up_str
449               }
450             }{

```

```

451         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
452     }
453 }
454 }{
455     \str_if_empty:NF \l_tmpa_tl {
456         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
457     }
458 }
459 }
460 }
461 \seq_gset_eq:NN #1 \l_tmpa_seq
462 }
463 }

```

(End definition for `\stex_path_canonicalize:N`. This function is documented on page 65.)

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

```

464 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
465     \seq_if_empty:NTF #1 {
466         \prg_return_false:
467     }{
468         \seq_get_left:NN #1 \l_tmpa_tl
469         \sys_if_platform_windows:TF{
470             \str_if_in:NnTF \l_tmpa_tl {:}{
471                 \prg_return_true:
472             }{
473                 \prg_return_false:
474             }
475         }{
476             \str_if_empty:NTF \l_tmpa_tl {
477                 \prg_return_true:
478             }{
479                 \prg_return_false:
480             }
481         }
482     }
483 }

```

(End definition for `\stex_path_if_absolute:N $\underline{TF}$` . This function is documented on page 65.)

## 25.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

484 \str_new:N\l_stex_kpsewhich_return_str
485 \cs_new_protected:Nn \stex_kpsewhich:n {\begingroup
486     \catcode'\ =12
487     \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
488     \tl_gset_eq:NN \l_tmpa_tl \l_tmpa_tl
489     \endgroup
490     \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
491     \tl_trim_spaces:N \l_stex_kpsewhich_return_str
492 }

```

(End definition for `\stex_kpsewhich:n`. This function is documented on page 65.)

We determine the PWD

`\c_stex_pwd_seq`  
`\c_stex_pwd_str`

```

493 \sys_if_platform_windows:TF{
494   \begingroup\escapechar=-1\catcode'\=12
495   \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
496   \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
497   \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_
498   }}{
499   \stex_kpsewhich:n{-var-value~PWD}
500 }
501
502 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
503 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
504 \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 65.)

## 25.3 File Hooks and Tracking

505 `<@@=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

506 `\seq_gclear_new:N\g_stex_files_stack`

(End definition for `\g_stex_files_stack`.)

`\c_stex_mainfile_seq`  
`\c_stex_mainfile_str`

```

507 \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
508 \stex_path_from_string:Nn \c_stex_mainfile_seq
509   \c_stex_mainfile_str

```

(End definition for `\c_stex_mainfile_seq` and `\c_stex_mainfile_str`. These variables are documented on page 65.)

`\g_stex_currentfile_seq`

510 `\seq_gclear_new:N\g_stex_currentfile_seq`

(End definition for `\g_stex_currentfile_seq`. This variable is documented on page 66.)

`\stex_filestack_push:n`

```

511 \cs_new_protected:Nn \stex_filestack_push:n {
512   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
513   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
514     \stex_path_from_string:Nn\g_stex_currentfile_seq{
515       \c_stex_pwd_str/#1
516     }

```

```

517 }
518 \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
519 \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
520 }

```

(End definition for `\stex_filestack_push:n`. This function is documented on page 66.)

`\stex_filestack_pop:`

```

521 \cs_new_protected:Nn \stex_filestack_pop: {
522   \seq_if_empty:NF\g__stex_files_stack{
523     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
524   }
525   \seq_if_empty:NTF\g__stex_files_stack{
526     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
527   }{
528     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
529     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
530   }
531 }

```

(End definition for `\stex_filestack_pop:`. This function is documented on page 66.)

Hooks for the current file:

```

532 \AddToHook{file/before}{
533   \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
534 }
535 \AddToHook{file/after}{
536   \stex_filestack_pop:
537 }

```

## 25.4 MathHub Repositories

```

538 <@=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.

```

539 \str_if_empty:NTF\mathhub{
540   \sys_if_platform_windows:TF{
541     \begingroup\escapechar=-1\catcode'\=12
542     \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str MATHHUB\c_percent_str}
543     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
544     \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_kpsewhich_return_str}}
545   }{
546     \stex_kpsewhich:n{-var-value-MATHHUB}
547   }
548   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
549 }
550 \str_if_empty:NT \c_stex_mathhub_str {
551   \sys_if_platform_windows:TF{
552     \begingroup\escapechar=-1\catcode'\=12
553     \exp_args:Nx\stex_kpsewhich:n{-var-value-HOME}
554     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
555     \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_kpsewhich_return_str}}
556   }{

```



```

557     \stex_kpsewhich:n{-var-value~HOME}
558   }
559   \ior_open:NnT \l_tmpa_ior{\l_stex_kpsewhich_return_str / .stex / mathhub.path}{
560     \begingroup\escapechar=-1\catcode'\=12
561     \ior_str_get:NN \l_tmpa_ior \l_tmpa_str
562     \sys_if_platform_windows:T{
563       \exp_args:NNx\str_replace_all:Nnn\l_tmpa_str{c_backslash_str}/
564     }
565     \str_gset_eq:NN \c_stex_mathhub_str\l_tmpa_str
566     \endgroup
567     \ior_close:N \l_tmpa_ior
568   }
569 }
570 \str_if_empty:NTF\c_stex_mathhub_str{
571   \msg_warning:nn{stex}{warning/nomathhub}
572 }{
573   \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
574   \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
575 }
576 }{
577   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
578   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
579     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
580       \c_stex_pwd_str/\mathhub
581     }
582   }
583   \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
584   \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
585 }

```

(End definition for `\mathhub`, `\c_stex_mathhub_seq`, and `\c_stex_mathhub_str`. These variables are documented on page 66.)

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

```

586 \cs_new_protected:Nn \_stex_mathhub_do_manifest:n {
587   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
588     \str_set:Nx \l_tmpa_str { #1 }
589     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
590     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
591     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
592     \_stex_mathhub_find_manifest:N \l_tmpa_seq
593     \seq_if_empty:NTF \l_stex_mathhub_manifest_file_seq {
594       \msg_error:nnxx{stex}{error/norepository}{#1}{
595         \stex_path_to_string:N \c_stex_mathhub_str
596       }
597     } {
598       \exp_args:No \_stex_mathhub_parse_manifest:n { \l_tmpa_str }
599     }
600   }
601 }

```

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

`\l_stex_mathhub_manifest_file_seq`

```
602 \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:

```
603 \cs_new_protected:Nn \__stex_mathhub_find_manifest:N {
604   \seq_set_eq:NN\l_tmpa_seq #1
605   \bool_set_true:N\l_tmpa_bool
606   \bool_while_do:Nn \l_tmpa_bool {
607     \seq_if_empty:NTF \l_tmpa_seq {
608       \bool_set_false:N\l_tmpa_bool
609     }{
610       \file_if_exist:nTF{
611         \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
612       }{
613         \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
614         \bool_set_false:N\l_tmpa_bool
615       }{
616         \file_if_exist:nTF{
617           \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
618         }{
619           \seq_put_right:Nn\l_tmpa_seq{META-INF}
620           \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
621           \bool_set_false:N\l_tmpa_bool
622         }{
623           \file_if_exist:nTF{
624             \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
625           }{
626             \seq_put_right:Nn\l_tmpa_seq{meta-inf}
627             \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
628             \bool_set_false:N\l_tmpa_bool
629           }{
630             \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
631           }
632         }
633       }
634     }
635   }
636   \seq_set_eq:NN\l__stex_mathhub_manifest_file_seq\l_tmpa_seq
637 }
```

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

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

```
638 \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:

```
639 \cs_new_protected:Nn \__stex_mathhub_parse_manifest:n {
640   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
641   \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
642   \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
```

```

643 \str_set:Nn \l_tmpa_str {##1}
644 \exp_args:NNoo \seq_set_split:Nnn
645   \l_tmpb_seq \c_colon_str \l_tmpa_str
646 \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
647   \exp_args:NNe \str_set:Nn \l_tmpb_tl {
648     \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
649   }
650   \exp_args:No \str_case:nnTF \l_tmpa_tl {
651     {id} {
652       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
653       { id } \l_tmpb_tl
654     }
655     {narration-base} {
656       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
657       { narr } \l_tmpb_tl
658     }
659     {url-base} {
660       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
661       { docurl } \l_tmpb_tl
662     }
663     {source-base} {
664       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
665       { ns } \l_tmpb_tl
666     }
667     {ns} {
668       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
669       { ns } \l_tmpb_tl
670     }
671     {dependencies} {
672       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
673       { deps } \l_tmpb_tl
674     }
675   }{}{}
676 }{}
677 }
678 \ior_close:N \c__stex_mathhub_manifest_ior
679 \stex_persist:x {
680   \prop_set_from_keyval:cn{ c_stex_mathhub_#1_manifest_prop }{
681     \exp_after:wN \prop_to_keyval:N \csname c_stex_mathhub_#1_manifest_prop\endcsname
682   }
683 }
684 }

```

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

`\stex_set_current_repository:n`

```

685 \cs_new_protected:Nn \stex_set_current_repository:n {
686   \stex_require_repository:n { #1 }
687   \prop_set_eq:Nc \l_stex_current_repository_prop {
688     c_stex_mathhub_#1_manifest_prop
689   }
690 }

```

(End definition for `\stex_set_current_repository:n`. This function is documented on page 66.)

`\stex_require_repository:n`

```

691 \cs_new_protected:Nn \stex_require_repository:n {
692   \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
693     \stex_debug:nn{mathhub}{Opening~archive:~#1}
694     \__stex_mathhub_do_manifest:n { #1 }
695   }
696 }

```

(End definition for `\stex_require_repository:n`. This function is documented on page 66.)

`\l_stex_current_repository_prop`

Current MathHub repository

```

697 %\prop_new:N \l_stex_current_repository_prop
698 \bool_if:NF \c_stex_persist_mode_bool {
699   \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
700   \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
701     \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
702   } {
703     \__stex_mathhub_parse_manifest:n { main }
704     \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
705     \l_tmpa_str
706     \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str_manifest_prop }
707     \c_stex_mathhub_main_manifest_prop
708     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
709     \stex_debug:nn{mathhub}{Current~repository:~
710     \prop_item:Nn \l_stex_current_repository_prop {id}
711   }
712 }
713 }

```

(End definition for `\l_stex_current_repository_prop`. This variable is documented on page 66.)

`\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.

```

714 \cs_new_protected:Nn \stex_in_repository:nn {
715   \str_set:Nx \l_tmpa_str { #1 }
716   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
717   \str_if_empty:NTF \l_tmpa_str {
718     \prop_if_exist:NTF \l_stex_current_repository_prop {
719       \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
720       \exp_args:Ne \l_tmpa_cs{
721         \prop_item:Nn \l_stex_current_repository_prop { id }
722       }
723     }{
724       \l_tmpa_cs{}
725     }
726   }{
727     \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}
728     \stex_require_repository:n \l_tmpa_str
729     \str_set:Nx \l_tmpa_str { #1 }
730     \exp_args:Nne \use:nn {
731       \stex_set_current_repository:n \l_tmpa_str
732       \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
733     }{
734       \stex_debug:nn{mathhub}{switching~back~to:~

```

```

735     \prop_if_exist:NTF \l_stex_current_repository_prop {
736       \prop_item:Nn \l_stex_current_repository_prop { id } :~
737       \meaning\l_stex_current_repository_prop
738     }{
739       no~repository
740     }
741   }
742   \prop_if_exist:NTF \l_stex_current_repository_prop {
743     \stex_set_current_repository:n {
744       \prop_item:Nn \l_stex_current_repository_prop { id }
745     }
746   }{
747     \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
748   }
749 }
750 }
751 }

```

(End definition for `\stex_in_repository:nn`. This function is documented on page 66.)

## 25.5 Using Content in Archives

`\mhpath`

```

752 \def \mhpath #1 #2 {
753   \exp_args:Ne \tl_if_empty:nTF{#1}{
754     \c_stex_mathhub_str /
755     \prop_item:Nn \l_stex_current_repository_prop { id }
756     / source / #2
757   }{
758     \c_stex_mathhub_str / #1 / source / #2
759   }
760 }

```

(End definition for `\mhpath`. This function is documented on page 67.)

`\inputref`

`\mhinput`

```

761 \newif \ifinputref \inputreffalse
762
763 \cs_new_protected:Nn \__stex_mathhub_mhinput:nn {
764   \stex_in_repository:nn {#1} {
765     \ifinputref
766       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
767     \else
768       \inputreftrue
769       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
770       \inputreffalse
771     \fi
772   }
773 }
774 \NewDocumentCommand \mhinput { 0{} m }{
775   \__stex_mathhub_mhinput:nn{ #1 }{ #2 }
776 }
777

```

```

778 \cs_new_protected:Nn \__stex_mathhub_inputref:nn {
779   \stex_in_repository:nn {#1} {
780     \stex_html_backend:TF {
781       \str_clear:N \l_tmpa_str
782       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
783         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
784       }
785
786       \tl_if_empty:nTF{ ##1 }{
787         \IfFileExists{#2}{
788           \stex_annotate_invisible:nnn{inputref}{
789             \l_tmpa_str / #2
790           }{}
791         }{
792           \input{#2}
793         }
794       }{
795         \IfFileExists{ \c_stex_mathhub_str / ##1 / source / #2 }{
796           \stex_annotate_invisible:nnn{inputref}{
797             \l_tmpa_str / #2
798           }{}
799         }{
800           \input{ \c_stex_mathhub_str / ##1 / source / #2 }
801         }
802       }
803
804     }{
805       \begingroup
806       \inputreftrue
807       \tl_if_empty:nTF{ ##1 }{
808         \input{#2}
809       }{
810         \input{ \c_stex_mathhub_str / ##1 / source / #2 }
811       }
812       \endgroup
813     }
814   }
815 }
816 \NewDocumentCommand \inputref { 0{} m}{
817   \__stex_mathhub_inputref:nn{ #1 }{ #2 }
818 }

```

(End definition for `\inputref` and `\mhinput`. These functions are documented on page 67.)

#### `\addmhbibresource`

```

819 \cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {
820   \stex_in_repository:nn {#1} {
821     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
822   }
823 }
824 \newcommand\addmhbibresource[2][]{
825   \__stex_mathhub_mhbibresource:nn{ #1 }{ #2 }
826 }

```

(End definition for `\addmhbibresource`. This function is documented on page 67.)

## `\libinput`

```
827 \cs_new_protected:Npn \libinput #1 {
828   \prop_if_exist:NF \l_stex_current_repository_prop {
829     \msg_error:nnn{stex}{error/notinarchive}\libinput
830   }
831   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
832     \msg_error:nnn{stex}{error/notinarchive}\libinput
833   }
834   \seq_clear:N \l__stex_mathhub_libinput_files_seq
835   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
836   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
837
838   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
839     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}
840     \IfFileExists{ \l_tmpa_str }{
841       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
842     }{}
843     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
844     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
845   }
846
847   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
848   \IfFileExists{ \l_tmpa_str }{
849     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
850   }{}
851
852   \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
853     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
854   }{
855     \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
856       \input{ ##1 }
857     }
858   }
859 }
```

(End definition for `\libinput`. This function is documented on page 67.)

## `\libusepackage`

```
860 \NewDocumentCommand \libusepackage {0{ } m} {
861   \prop_if_exist:NF \l_stex_current_repository_prop {
862     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
863   }
864   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
865     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
866   }
867   \seq_clear:N \l__stex_mathhub_libinput_files_seq
868   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
869   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
870
871   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
872     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2}
873     \IfFileExists{ \l_tmpa_str.sty }{
874       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
875     }{}
876   }
```

```

876 \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
877 \seq_put_right:No \l_tmpa_seq \l_tmpa_str
878 }
879
880 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2}
881 \IfFileExists{ \l_tmpa_str.sty }{
882 \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
883 }{}
884
885 \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
886 \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
887 }{
888 \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
889 \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
890 \usepackage[#1]{ #1 }
891 }
892 }{
893 \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
894 }
895 }
896 }

```

(End definition for `\libusepackage`. This function is documented on page 67.)

`\mhgraphics`  
`\cmhgraphics`

```

897
898 \AddToHook{begindocument}{
899 \ltx@ifpackageloaded{graphicx}{
900 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
901 \newcommand\mhgraphics[2][]{\%
902 \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
903 \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}}
904 \newcommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
905 }{}

```

(End definition for `\mhgraphics` and `\cmhgraphics`. These functions are documented on page 67.)

`\lstinputmhlisting`  
`\clstinputmhlisting`

```

906 \ltx@ifpackageloaded{listings}{
907 \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
908 \newcommand\lstinputmhlisting[2][]{\%
909 \def\lst@mhrepos{}\setkeys{lst}{#1}%
910 \lstinputlisting[#1]{\mhp\lst@mhrepos{#2}}}
911 \newcommand\clstinputmhlisting[2][]{\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
912 }{}
913 }
914
915 </package>

```

(End definition for `\lstinputmhlisting` and `\clstinputmhlisting`. These functions are documented on page 67.)



## Chapter 26

# STEX -References Implementation

```
916 <*package>
917
918 %%%%%%%%%% references.dtx %%%%%%%%%%
919
920 <@@=stex_refs>
    Warnings and error messages
921
```

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

```
922 %\iow_new:N \c__stex_refs_refs_iow
923 \AtBeginDocument{
924 % \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
925 }
926 \AtEndDocument{
927 % \iow_close:N \c__stex_refs_refs_iow
928 }
```

`\STEXreftitle`

```
929 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
930
931 \NewDocumentCommand \STEXreftitle { m } {
932   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
933 }
```

*(End definition for `\STEXreftitle`. This function is documented on page 68.)*

### 26.1 Document URIs and URLs

`\l_stex_current_docns_str`

```
934 \str_new:N \l_stex_current_docns_str
```

*(End definition for `\l_stex_current_docns_str`. This variable is documented on page 68.)*

`\stex_get_document_uri:`

```
935 \cs_new_protected:Nn \stex_get_document_uri: {
936   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
937   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
938   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
939   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
940   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
941
942   \str_clear:N \l_tmpa_str
943   \prop_if_exist:NT \l_stex_current_repository_prop {
944     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
945       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
946     }
947   }
948
949   \str_if_empty:NTF \l_tmpa_str {
950     \str_set:Nx \l_stex_current_docns_str {
951       file:/\stex_path_to_string:N \l_tmpa_seq
952     }
953   }{
954     \bool_set_true:N \l_tmpa_bool
955     \bool_while_do:Nn \l_tmpa_bool {
956       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
957       \exp_args:No \str_case:nnTF { \l_tmpb_str } {
958         {source} { \bool_set_false:N \l_tmpa_bool }
959       }{}{
960         \seq_if_empty:NT \l_tmpa_seq {
961           \bool_set_false:N \l_tmpa_bool
962         }
963       }
964     }
965
966     \seq_if_empty:NTF \l_tmpa_seq {
967       \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
968     }{
969       \str_set:Nx \l_stex_current_docns_str {
970         \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
971       }
972     }
973   }
974 }
```

(End definition for `\stex_get_document_uri:`. This function is documented on page 68.)

`\l_stex_current_docurl_str`

```
975 \str_new:N \l_stex_current_docurl_str
```

(End definition for `\l_stex_current_docurl_str`. This variable is documented on page 68.)

`\stex_get_document_url:`

```
976 \cs_new_protected:Nn \stex_get_document_url: {
977   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
978   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
979   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
```

```

980 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
981 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
982
983 \str_clear:N \l_tmpa_str
984 \prop_if_exist:NT \l_stex_current_repository_prop {
985   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
986     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
987       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
988     }
989   }
990 }
991
992 \str_if_empty:NTF \l_tmpa_str {
993   \str_set:Nx \l_stex_current_docurl_str {
994     file:/\stex_path_to_string:N \l_tmpa_seq
995   }
996 }{
997   \bool_set_true:N \l_tmpa_bool
998   \bool_while_do:Nn \l_tmpa_bool {
999     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1000     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1001       {source} { \bool_set_false:N \l_tmpa_bool }
1002     }{}{
1003       \seq_if_empty:NT \l_tmpa_seq {
1004         \bool_set_false:N \l_tmpa_bool
1005       }
1006     }
1007   }
1008 }
1009 \seq_if_empty:NTF \l_tmpa_seq {
1010   \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
1011 }{
1012   \str_set:Nx \l_stex_current_docurl_str {
1013     \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
1014   }
1015 }
1016 }
1017 }

```

(End definition for `\stex_get_document_url`:. This function is documented on page 68.)

## 26.2 Setting Reference Targets

```

1018 \str_const:Nn \c__stex_refs_url_str{URL}
1019 \str_const:Nn \c__stex_refs_ref_str{REF}
1020 \str_new:N \l__stex_refs_curr_label_str
1021 % @currentlabel -> number
1022 % @currentlabelname -> title
1023 % @currentHref -> name.number <- id of some kind
1024 % \theH# -> \arabic{section}
1025 % \the# -> number
1026 % \hyper@makecurrent{#}
1027 \int_new:N \l__stex_refs_unnamed_counter_int

```

`\stex_ref_new_doc_target:n`

```

1028 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
1029   \stex_get_document_uri:
1030   \str_clear:N \l__stex_refs_curr_label_str
1031   \str_set:Nx \l_tmpa_str { #1 }
1032   \str_if_empty:NT \l_tmpa_str {
1033     \int_incr:N \l__stex_refs_unnamed_counter_int
1034     \str_set:Nx \l_tmpa_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
1035   }
1036   \str_set:Nx \l__stex_refs_curr_label_str {
1037     \l_stex_current_docns_str?\l_tmpa_str
1038   }
1039   \seq_if_exist:cF{g__stex_refs_labels_\l_tmpa_str_seq}{
1040     \seq_new:c {g__stex_refs_labels_\l_tmpa_str_seq}
1041   }
1042   \seq_if_in:coF{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str {
1043     \seq_gput_right:co{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str
1044   }
1045   \stex_if_smsmode:TF {
1046     \stex_get_document_url:
1047     \str_gset_eq:cN {sref_url_\l__stex_refs_curr_label_str_str}\l_stex_current_docurl_str
1048     \str_gset_eq:cN {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_url_str
1049   }{
1050     %\iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter\unexpanded\expandafter{
1051     \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
1052     \immediate\write\@auxout{\stexauxadddocref{\l_stex_current_docns_str}{\l_tmpa_str}}
1053     \str_gset:cx {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_ref_str
1054   }
1055 }

```

(End definition for `\stex_ref_new_doc_target:n`. This function is documented on page 68.)

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

```

1056 \cs_new_protected:Npn \stexauxadddocref #1 #2 {
1057   \str_set:Nn \l_tmpa_str {#1?#2}
1058   \str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
1059   \seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
1060     \seq_new:c {g__stex_refs_labels_#2_seq}
1061   }
1062   \seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
1063     \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
1064   }
1065 }

```

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

```

1066 \AtEndDocument{
1067   \def\stexauxadddocref#1 #2 {}{}
1068 }

```

`\stex_ref_new_sym_target:n`

```

1069 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
1070   \stex_if_smsmode:TF {
1071     \str_if_exist:cF{sref_sym_#1_type}{
1072       \stex_get_document_url:
1073       \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str

```

```

1074     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
1075   }
1076   ){
1077     \str_if_empty:NF \l__stex_refs_curr_label_str {
1078       \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
1079       \immediate\write\@auxout{
1080         \exp_not:N\expandafter\def\exp_not:N\csname \exp_not:N\detokenize{sref_sym_#1_label_
1081           \l__stex_refs_curr_label_str
1082         }
1083       }
1084     }
1085   }
1086 }

```

(End definition for `\stex_ref_new_sym_target:n`. This function is documented on page 68.)

## 26.3 Using References

```

1087 \str_new:N \l__stex_refs_indocument_str

```

**\sref** Optional arguments:

```

1088
1089 \keys_define:nn { stex / sref } {
1090   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
1091   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
1092   pre           .tl_set:N = \l__stex_refs_pre_tl ,
1093   post          .tl_set:N = \l__stex_refs_post_tl ,
1094 }
1095 \cs_new_protected:Nn \__stex_refs_args:n {
1096   \tl_clear:N \l__stex_refs_linktext_tl
1097   \tl_clear:N \l__stex_refs_fallback_tl
1098   \tl_clear:N \l__stex_refs_pre_tl
1099   \tl_clear:N \l__stex_refs_post_tl
1100   \str_clear:N \l__stex_refs_repo_str
1101   \keys_set:nn { stex / sref } { #1 }
1102 }

```

The actual macro:

```

1103 \NewDocumentCommand \sref { 0{} m}{
1104   \__stex_refs_args:n { #1 }
1105   \str_if_empty:NTF \l__stex_refs_indocument_str {
1106     \str_set:Nx \l_tmpa_str { #2 }
1107     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
1108     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
1109       \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
1110         \seq_get_left:cNF {g__stex_refs_labels_\l_tmpa_str _seq} \l_tmpa_str {
1111           \str_clear:N \l_tmpa_str
1112         }
1113       }{
1114         \str_clear:N \l_tmpa_str
1115       }
1116     }{
1117       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1118       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str

```

```

1119 \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
1120 \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str_seq}{
1121   \str_set_eq:NN \l_tmpc_str \l_tmpa_str
1122   \str_clear:N \l_tmpa_str
1123   \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str_seq} {
1124     \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
1125       \str_range:nnn { ##1 }{-\l_tmpa_int}{ -1 }
1126     }{
1127       \seq_map_break:n {
1128         \str_set:Nn \l_tmpa_str { ##1 }
1129       }
1130     }
1131   }
1132 }{
1133   \str_clear:N \l_tmpa_str
1134 }
1135 }
1136 \str_if_empty:NTF \l_tmpa_str {
1137   \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_lin
1138 }{
1139   \str_if_eq:cNTF {sref_\l_tmpa_str_type} \c__stex_refs_ref_str {
1140     \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1141       \cs_if_exist:cTF{autoref}{
1142         \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1143       }{
1144         \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1145       }
1146     }{
1147       \ltx@ifpackageloaded{hyperref}{
1148         \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
1149       }{
1150         \l__stex_refs_linktext_tl
1151       }
1152     }
1153   }{
1154     \ltx@ifpackageloaded{hyperref}{
1155       \href{\use:c{sref_url_\l_tmpa_str_str}}{\tl_if_empty:NTF \l__stex_refs_linktext_t
1156     }{
1157       \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs
1158     }
1159   }
1160 }
1161 }{
1162   % TODO
1163 }
1164 }

```

(End definition for \sref. This function is documented on page 69.)

## \srefsym

```

1165 \NewDocumentCommand \srefsym { 0{} m}{
1166   \stex_get_symbol:n { #2 }
1167   \__stex_refs_sym_aux:nn{##1}{\l_stex_get_symbol_uri_str}
1168 }

```

```

1169
1170 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1171   \str_if_exist:cTF {sref_sym_#2 _label_str }{
1172     \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1173   }{
1174     \__stex_refs_args:n { #1 }
1175     \str_if_empty:NTF \l__stex_refs_indocument_str {
1176       \tl_if_exist:cTF{sref_sym_#2 _type}{
1177         % doc uri in \l_tmpb_str
1178         \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1179         \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1180           % reference
1181           \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1182             \cs_if_exist:cTF{autoref}{
1183               \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1184             }{
1185               \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1186             }
1187           }{
1188             \ltx@ifpackageloaded{hyperref}{
1189               \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1190             }{
1191               \l__stex_refs_linktext_tl
1192             }
1193           }
1194         }{
1195           % URL
1196           \ltx@ifpackageloaded{hyperref}{
1197             \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl \
1198           }{
1199             \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_re
1200           }
1201         }
1202       }{
1203         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_l
1204       }
1205     }{
1206       % TODO
1207     }
1208   }
1209 }

```

(End definition for \srefsym. This function is documented on page 69.)

**\srefsymuri**

```

1210 \cs_new_protected:Npn \srefsymuri #1 #2 {
1211   \__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1212 }

```

(End definition for \srefsymuri. This function is documented on page 69.)

```

1213 </package>

```

## Chapter 27

# STEX -Modules Implementation

```
1214 <*package>
1215
1216 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1217
1218 <@@=stex_modules>
1219
1220 Warnings and error messages
1221 \msg_new:nnn{stex}{error/unknownmodule}{
1222   No~module~#1~found
1223 }
1224 \msg_new:nnn{stex}{error/syntax}{
1225   Syntax~error:~#1
1226 }
1227 \msg_new:nnn{stex}{error/siglanguage}{
1228   Module~#1~declares~signature~#2,~but~does~not~
1229   declare~its~language
1230 }
1231 \msg_new:nnn{stex}{warning/deprecated}{
1232   #1~is~deprecated;~please~use~#2~instead!
1233 }
1234 \msg_new:nnn{stex}{error/conflictingmodules}{
1235   Conflicting~imports~for~module~#1
1236 }
```

`\l_stex_current_module_str` The current module:

```
1236 \str_new:N \l_stex_current_module_str
```

(End definition for `\l_stex_current_module_str`. This variable is documented on page 71.)

`\l_stex_all_modules_seq` Stores all available modules

```
1237 \seq_new:N \l_stex_all_modules_seq
```

(End definition for `\l_stex_all_modules_seq`. This variable is documented on page 71.)



```

\stex_if_in_module_p:
\stex_if_in_module:TF
1238 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1239   \str_if_empty:NTF \l_stex_current_module_str
1240   \prg_return_false: \prg_return_true:
1241 }

(End definition for \stex_if_in_module:TF. This function is documented on page 71.)

```

```

\stex_if_module_exists_p:n
\stex_if_module_exists:nTF
1242 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1243   \prop_if_exist:cTF { c_stex_module_#1_prop }
1244   \prg_return_true: \prg_return_false:
1245 }

(End definition for \stex_if_module_exists:nTF. This function is documented on page 71.)

```

```

\stex_add_to_current_module:n
\STEXexport
1246 \cs_new_protected:Nn \stex_execute_in_module:n { \stex_if_in_module:T {
1247   \stex_add_to_current_module:n { #1 }
1248   \stex_do_up_to_module:n { #1 }
1249 }}
1250 \cs_generate_variant:Nn \stex_execute_in_module:n {x}
1251
1252 \cs_new_protected:Nn \stex_add_to_current_module:n {
1253   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1254 }
1255 \cs_generate_variant:Nn \stex_add_to_current_module:n {x}
1256 \cs_new_protected:Npn \STEXexport {
1257   \begingroup
1258   \newlinechar=-1\relax
1259   \endlinechar=-1\relax
1260   %\catcode'\ = 9\relax
1261   \expandafter\endgroup\__stex_modules_export:n
1262 }
1263 \cs_new_protected:Nn \__stex_modules_export:n {
1264   \ignorespaces #1
1265   \stex_add_to_current_module:n { \ignorespaces #1 }
1266   \stex_smsmode_do:
1267 }
1268 \stex_deactivate_macro:Nn \STEXexport {module~environments}

(End definition for \stex_add_to_current_module:n and \STEXexport. These functions are documented
on page 71.)

```

```

\stex_add_constant_to_current_module:n
1269 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1270   \str_set:Nx \l_tmpa_str { #1 }
1271   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1272 }

(End definition for \stex_add_constant_to_current_module:n. This function is documented on page
71.)

```

`\stex_add_import_to_current_module:n`

```

1273 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1274   \str_set:Nx \l_tmpa_str { #1 }
1275   \exp_args:Nno
1276   \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1277     \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1278   }
1279 }

```

(End definition for `\stex_add_import_to_current_module:n`. This function is documented on page 71.)

`\stex_collect_imports:n`

```

1280 \cs_new_protected:Nn \stex_collect_imports:n {
1281   \seq_clear:N \l_stex_collect_imports_seq
1282   \__stex_modules_collect_imports:n {#1}
1283 }
1284 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1285   \seq_map_inline:cn {c_stex_module_#1_imports} {
1286     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1287       \__stex_modules_collect_imports:n { ##1 }
1288     }
1289   }
1290   \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1291     \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1292   }
1293 }

```

(End definition for `\stex_collect_imports:n`. This function is documented on page 71.)

`\stex_do_up_to_module:n`

```

1294 \int_new:N \l__stex_modules_group_depth_int
1295 \cs_new_protected:Nn \stex_do_up_to_module:n {
1296   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1297     #1
1298   }{
1299     #1
1300     \expandafter \tl_gset:Nn
1301     \csname l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1302     \expandafter\expandafter\expandafter\endcsname
1303     \expandafter\expandafter\expandafter { \csname
1304       l__stex_modules_aftergroup_\l_stex_current_module_str _tl\endcsname #1 }
1305     \aftergroup\__stex_modules_aftergroup_do:
1306   }
1307 }
1308 \cs_generate_variant:Nn \stex_do_up_to_module:n {x}
1309 \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1310   \stex_debug:nn{aftergroup}{\cs_meaning:c{
1311     l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1312   }}
1313   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1314     \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1315     \tl_gclear:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1316   }{
1317     \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}

```

```

1318     \aftergroup\__stex_modules_aftergroup_do:
1319   }
1320 }
1321 \cs_new_protected:Nn \stex_reset_up_to_module:n {
1322   \expandafter\let\csname l__stex_modules_aftergroup_#1_tl\endcsname\undefined
1323 }

```

(End definition for `\stex_do_up_to_module:n`. This function is documented on page 71.)

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

```

1324

```

(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.

```

1325 \str_new:N \l_stex_module_ns_str
1326 \str_new:N \l_stex_module_subpath_str
1327 \cs_new_protected:Nn \__stex_modules_compute_namespace:nN {
1328   \seq_set_eq:NN \l_tmpa_seq #2
1329   % split off file extension
1330   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str % <- filename
1331   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1332   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str % <- filename without suffixes
1333   \seq_put_right:No \l_tmpa_seq \l_tmpb_str % <- file path including name without suffixes
1334
1335   \bool_set_true:N \l_tmpa_bool
1336   \bool_while_do:Nn \l_tmpa_bool {
1337     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1338     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1339       {source} { \bool_set_false:N \l_tmpa_bool }
1340     }{}{
1341       \seq_if_empty:NT \l_tmpa_seq {
1342         \bool_set_false:N \l_tmpa_bool
1343       }
1344     }
1345   }
1346
1347   \stex_path_to_string:NN \l_tmpa_seq \l_stex_module_subpath_str
1348   % \l_tmpa_seq <- sub-path relative to archive
1349   \str_if_empty:NTF \l_stex_module_subpath_str {
1350     \str_set:Nx \l_stex_module_ns_str {#1}
1351   }{
1352     \str_set:Nx \l_stex_module_ns_str {
1353       #1/\l_stex_module_subpath_str
1354     }
1355   }
1356 }
1357
1358 \cs_new_protected:Nn \stex_modules_current_namespace: {
1359   \str_clear:N \l_stex_module_subpath_str
1360   \prop_if_exist:NTF \l_stex_current_repository_prop {
1361     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str

```

```

1362     \__stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1363   }{
1364     % split off file extension
1365     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1366     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1367     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1368     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1369     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1370     \str_set:Nx \l_stex_module_ns_str {
1371       file:/\stex_path_to_string:N \l_tmpa_seq
1372     }
1373   }
1374 }

```

(End definition for `\stex_modules_current_namespace:.` This function is documented on page [72](#).)

## 27.1 The smodule environment

smodule arguments:

```

1375 \keys_define:nn { stex / module } {
1376   title      .tl_set:N      = \smodulename ,
1377   type       .str_set_x:N   = \smodulename ,
1378   id         .str_set_x:N   = \smoduleid ,
1379   deprecate  .str_set_x:N   = \l_stex_module_deprecate_str ,
1380   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1381   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1382   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1383   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1384   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1385   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1386   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1387 }
1388
1389 \cs_new_protected:Nn \__stex_modules_args:n {
1390   \str_clear:N \smodulename
1391   \str_clear:N \smodulename
1392   \str_clear:N \smoduleid
1393   \str_clear:N \l_stex_module_ns_str
1394   \str_clear:N \l_stex_module_deprecate_str
1395   \str_clear:N \l_stex_module_lang_str
1396   \str_clear:N \l_stex_module_sig_str
1397   \str_clear:N \l_stex_module_creators_str
1398   \str_clear:N \l_stex_module_contributors_str
1399   \str_clear:N \l_stex_module_meta_str
1400   \str_clear:N \l_stex_module_srccite_str
1401   \keys_set:nn { stex / module } { #1 }
1402 }
1403
1404 % module parameters here? In the body?
1405

```

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

```

1406 \cs_new_protected:Nn \stex_module_setup:nn {

```

```

1407 \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1408 \str_set:Nx \l_stex_module_name_str { #2 }
1409 \__stex_modules_args:n { #1 }

```

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

```

1410 \stex_if_in_module:TF {
1411   % Nested module
1412   \prop_get:cnN {c_stex_module\l_stex_current_module_str _prop}
1413   { ns } \l_stex_module_ns_str
1414   \str_set:Nx \l_stex_module_name_str {
1415     \prop_item:cn {c_stex_module\l_stex_current_module_str _prop}
1416     { name } / \l_stex_module_name_str
1417   }
1418   \str_if_empty:NT \l_stex_module_lang_str {
1419     \str_set:Nx \l_stex_module_lang_str {
1420       \prop_item:cn {c_stex_module\l_stex_current_module_str _prop}
1421       { lang }
1422     }
1423   }
1424 }{
1425   % not nested:
1426   \str_if_empty:NT \l_stex_module_ns_str {
1427     \stex_modules_current_namespace:
1428     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1429       / {\l_stex_module_ns_str}
1430     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1431     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1432       \str_set:Nx \l_stex_module_ns_str {
1433         \stex_path_to_string:N \l_tmpa_seq
1434       }
1435     }
1436   }
1437 }

```

Next, we determine the language of the module:

```

1438 \str_if_empty:NT \l_stex_module_lang_str {
1439   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1440   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1441   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1442   \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
1443     \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
1444       \exp_args:NNNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
1445     }
1446   }
1447   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1448   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
1449     \seq_pop_right:NN \l_tmpa_seq \l_stex_module_lang_str
1450     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1451       inferred~from~file~name}
1452   }
1453 }
1454
1455 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {

```

```

1456     \exp_args:NNo \stex_set_language:Nn \l_tmpa_str \l_stex_module_lang_str
1457   }}

```

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

```

1458   \str_if_empty:NTF \l_stex_module_sig_str {
1459     \exp_args:Nnx \prop_gset_from_keyval:cn {
1460       c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1461     } {
1462       name      = \l_stex_module_name_str ,
1463       ns        = \l_stex_module_ns_str ,
1464       file      = \exp_not:o { \g_stex_currentfile_seq } ,
1465       lang      = \l_stex_module_lang_str ,
1466       sig       = \l_stex_module_sig_str ,
1467       deprecate = \l_stex_module_deprecate_str ,
1468       meta      = \l_stex_module_meta_str
1469     }
1470     \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1471     \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1472     \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _copymodules}
1473     \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1474     \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1475   \str_if_empty:NT \l_stex_module_meta_str {
1476     \str_set:Nx \l_stex_module_meta_str {
1477       \c_stex_metatheory_ns_str ? Metatheory
1478     }
1479   }
1480   \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1481     \bool_set_true:N \l_stex_in_meta_bool
1482     \exp_args:Nx \stex_add_to_current_module:n {
1483       \bool_set_true:N \l_stex_in_meta_bool
1484       \stex_activate_module:n {\l_stex_module_meta_str}
1485       \bool_set_false:N \l_stex_in_meta_bool
1486     }
1487     \stex_activate_module:n {\l_stex_module_meta_str}
1488     \bool_set_false:N \l_stex_in_meta_bool
1489   }
1490 }{
1491   \str_if_empty:NT \l_stex_module_lang_str {
1492     \msg_error:nnxx{stex}{error/siglanguage}{
1493       \l_stex_module_ns_str?\l_stex_module_name_str
1494     }\l_stex_module_sig_str}
1495   }
1496   \stex_debug:nn{modules}{Signature~\l_stex_module_sig_str~for~\l_stex_module_ns_str?\l_st
1497   \stex_if_module_exists:nTF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1498     \stex_debug:nn{modules}{(already exists)}
1499   }{
1500     \stex_debug:nn{modules}{(needs loading)}
1501     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1502     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1503     \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1504     \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex

```

```

1505 \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1506 \str_set:Nx \l_tmpa_str {
1507   \stex_path_to_string:N \l_tmpa_seq /
1508   \l_tmpa_str . \l_stex_module_sig_str .tex
1509 }
1510 \IfFileExists \l_tmpa_str {
1511   \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1512     \str_clear:N \l_stex_current_module_str
1513     \seq_clear:N \l_stex_all_modules_seq
1514     \stex_debug:nn{modules}{Loading~signature}
1515   }
1516   {{
1517     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1518   }
1519 }
1520 \stex_if_smsmode:F {
1521   \stex_activate_module:n {
1522     \l_stex_module_ns_str ? \l_stex_module_name_str
1523   }
1524 }
1525 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1526 }
1527 \str_if_empty:NF \l_stex_module_deprecate_str {
1528   \msg_warning:nnxx{stex}{warning/deprecated}{
1529     Module~\l_stex_current_module_str
1530   }{
1531     \l_stex_module_deprecate_str
1532   }
1533 }
1534 \seq_put_right:Nx \l_stex_all_modules_seq {
1535   \l_stex_module_ns_str ? \l_stex_module_name_str
1536 }
1537 \tl_clear:c{l__stex_modules_aftergroup\l_stex_module_ns_str ? \l_stex_module_name_str _tl}
1538 }

```

(End definition for `\stex_module_setup:nn`. This function is documented on page [72](#).)

**smodule** The module environment.

```

\__stex_modules_begin_module: implements \begin{smodule}
1539 \cs_new_protected:Nn \__stex_modules_begin_module: {
1540   \stex_reactivate_macro:N \STEXexport
1541   \stex_reactivate_macro:N \importmodule
1542   \stex_reactivate_macro:N \symdecl
1543   \stex_reactivate_macro:N \notation
1544   \stex_reactivate_macro:N \symdef
1545
1546   \stex_debug:nn{modules}{
1547     New~module:\\
1548     Namespace:~\l_stex_module_ns_str\\
1549     Name:~\l_stex_module_name_str\\
1550     Language:~\l_stex_module_lang_str\\
1551     Signature:~\l_stex_module_sig_str\\
1552     Metatheory:~\l_stex_module_meta_str\\

```

```

1553   File:~\stex_path_to_string:N \g_stex_currentfile_seq
1554 }
1555
1556 \stex_if_do_html:T{
1557   \begin{stex_annotate_env} {theory} {
1558     \l_stex_module_ns_str ? \l_stex_module_name_str
1559   }
1560
1561   \stex_annotate_invisible:nnn{header}{} {
1562     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1563     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1564     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1565       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1566     }
1567     \str_if_empty:NF \smoduletype {
1568       \stex_annotate:nnn{type}{\smoduletype}{}
1569     }
1570   }
1571 }
1572 % TODO: Inherit metatheory for nested modules?
1573 }
1574 \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}

```

1575 \cs_new_protected:Nn \_stex_modules_end_module: {
1576   \stex_debug:nn{modules}{Closing~module~\prop_item:cn {c_stex_module_\l_stex_current_module_str}
1577   \stex_reset_up_to_module:n \l_stex_current_module_str
1578   \stex_if_smsmode:T {
1579     \stex_persist:x {
1580       \prop_set_from_keyval:cn{c_stex_module_\l_stex_current_module_str _prop}{
1581         \exp_after:wN \prop_to_keyval:N \csname c_stex_module_\l_stex_current_module_str _pr
1582       }
1583       \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _constants}{
1584         \seq_use:cn{c_stex_module_\l_stex_current_module_str _constants},
1585       }
1586       \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _imports}{
1587         \seq_use:cn{c_stex_module_\l_stex_current_module_str _imports},
1588       }
1589       \tl_set:cn {c_stex_module_\l_stex_current_module_str _code}
1590     }
1591     \exp_after:wN \let \exp_after:wN \l_tmpa_tl \csname c_stex_module_\l_stex_current_module_str
1592     \exp_after:wN \stex_persist:n \exp_after:wN { \exp_after:wN { \l_tmpa_tl } }
1593   }
1594 }

```

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

The core environment

```

1595 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1596 \NewDocumentEnvironment { smodule } { 0 } { m } {
1597   \stex_module_setup:nn{#1}{#2}
1598   %\par
1599   \stex_if_smsmode:F{

```



```

1600 \tl_if_empty:NF \smodulename {
1601   \exp_args:No \stex_document_title:n \smodulename
1602 }
1603 \tl_clear:N \l_tmpa_tl
1604 \clist_map_inline:Nn \smodulename {
1605   \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1606     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1607   }
1608 }
1609 \tl_if_empty:NTF \l_tmpa_tl {
1610   \__stex_modules_smodule_start:
1611 }{
1612   \l_tmpa_tl
1613 }
1614 }
1615 \__stex_modules_begin_module:
1616 \str_if_empty:NF \smoduleid {
1617   \stex_ref_new_doc_target:n \smoduleid
1618 }
1619 \stex_smsmode_do:
1620 } {
1621   \__stex_modules_end_module:
1622   \stex_if_smsmode:F {
1623     \end{stex_annotate_env}
1624     \clist_set:No \l_tmpa_clist \smodulename
1625     \tl_clear:N \l_tmpa_tl
1626     \clist_map_inline:Nn \l_tmpa_clist {
1627       \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1628         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1629       }
1630     }
1631     \tl_if_empty:NTF \l_tmpa_tl {
1632       \__stex_modules_smodule_end:
1633     }{
1634       \l_tmpa_tl
1635     }
1636   }
1637 }

```

### **\stexpatchmodule**

```

1638 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1639 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1640
1641 \newcommand\stexpatchmodule[3] [] {
1642   \str_set:Nx \l_tmpa_str{ #1 }
1643   \str_if_empty:NTF \l_tmpa_str {
1644     \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1645     \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1646   }{
1647     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1648     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1649   }
1650 }

```

(End definition for \stexpatchmodule. This function is documented on page 72.)

## 27.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n 1651 \NewDocumentCommand \STEXModule { m } {
1652   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1653   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1654   \tl_set:Nn \l_tmpa_tl {
1655     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1656   }
1657   \seq_map_inline:Nn \l_stex_all_modules_seq {
1658     \str_set:Nn \l_tmpb_str { ##1 }
1659     \str_if_eq:eeT { \l_tmpa_str } {
1660       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1661     } {
1662       \seq_map_break:n {
1663         \tl_set:Nn \l_tmpa_tl {
1664           \stex_invoke_module:n { ##1 }
1665         }
1666       }
1667     }
1668   }
1669   \l_tmpa_tl
1670 }
1671
1672 \cs_new_protected:Nn \stex_invoke_module:n {
1673   \stex_debug:nn{modules}{Invoking~module~#1}
1674   \peek_charcode_remove:NTF ! {
1675     \__stex_modules_invoke_uri:nN { #1 }
1676   } {
1677     \peek_charcode_remove:NTF ? {
1678       \__stex_modules_invoke_symbol:nn { #1 }
1679     } {
1680       \msg_error:nnx{stex}{error/syntax}{
1681         ?~or~!~expected~after~
1682         \c_backslash_str STEXModule{#1}
1683       }
1684     }
1685   }
1686 }
1687
1688 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1689   \str_set:Nn #2 { #1 }
1690 }
1691
1692 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1693   \stex_invoke_symbol:n{#1?#2}
1694 }

```

(End definition for `\STEXModule` and `\stex_invoke_module:n`. These functions are documented on page 72.)

```

\stex_activate_module:n
1695 \bool_new:N \l_stex_in_meta_bool
1696 \bool_set_false:N \l_stex_in_meta_bool

```

```

1697 \cs_new_protected:Nn \stex_activate_module:n {
1698   \stex_debug:nn{modules}{Activating~module~#1}
1699   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1700     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1701     \use:c{ c_stex_module_#1_code }
1702   }
1703 }

```

*(End definition for \stex\_activate\_module:n. This function is documented on page 73.)*

```

1704 \endpackage

```

## Chapter 28

# STEX -Module Inheritance Implementation

```
1705 <*package>
1706
1707 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1708
```

### 28.1 SMS Mode

```
1709 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1710 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1711 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1712 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1713
1714 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1715   \makeatletter
1716   \makeatother
1717   \ExplSyntaxOn
1718   \ExplSyntaxOff
1719   \rustexBREAK
1720 }
1721
1722 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1723   \symdef
1724   \importmodule
1725   \notation
1726   \symdecl
1727   \STEXexport
1728   \inlineass
1729   \inlinedef
1730   \inlineex
1731   \endinput
1732   \setnotation
```

```

1733 \copynotation
1734 \assign
1735 \renamedekl
1736 \donotcopy
1737 \instantiate
1738 }
1739
1740 \exp_args:Nn \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1741   \tl_to_str:n {
1742     smodule,
1743     copymodule,
1744     interpretmodule,
1745     sdefinition,
1746     sexample,
1747     sassertion,
1748     sparagraph,
1749     mathstructure
1750   }
1751 }

```

(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 74.)

`\stex_if_smsmode_p:`  
`\stex_if_smsmode:TF`

```

1752 \bool_new:N \g__stex_smsmode_bool
1753 \bool_set_false:N \g__stex_smsmode_bool
1754 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1755   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1756 }

```

(End definition for `\stex_if_smsmode:TF`. This function is documented on page 74.)

`\_stex_smsmode_in_smsmode:nn`

```

1757 \cs_new_protected:Nn \_stex_smsmode_in_smsmode:nn { \stex_suppress_html:n {
1758   \vbox_set:Nn \l_tmpa_box {
1759     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1760     \bool_gset_true:N \g__stex_smsmode_bool
1761     #2
1762     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1763   }
1764   \box_clear:N \l_tmpa_box
1765 } }

```

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

`\stex_file_in_smsmode:nn`

```

1766 \quark_new:N \q__stex_smsmode_break
1767
1768 \NewDocumentCommand \_stex_smsmode_importmodule: { 0{} m } {
1769   \seq_gput_right:Nn \l__stex_smsmode_importmodules_seq { {#1}{#2} }
1770   \stex_smsmode_do:
1771 }
1772
1773 \cs_new_protected:Nn \_stex_smsmode_module:nn {
1774   \_stex_modules_args:n{#1}

```

```

1775 \stex_if_in_module:F {
1776   \str_if_empty:NF \l_stex_module_sig_str {
1777     \stex_modules_current_namespace:
1778     \str_set:Nx \l_stex_module_name_str { #2 }
1779     \stex_if_module_exists:nF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1780       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1781       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1782       \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1783       \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1784       \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1785       \str_set:Nx \l_tmpa_str {
1786         \stex_path_to_string:N \l_tmpa_seq /
1787         \l_tmpa_str . \l_stex_module_sig_str .tex
1788       }
1789       \IfFileExists \l_tmpa_str {
1790         \exp_args:NNx \seq_gput_right:Nn \l__stex_smsmode_sigmodules_seq \l_tmpa_str
1791       }{
1792         \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1793       }
1794     }
1795   }
1796 }
1797 }
1798
1799 \prg_new_conditional:Nnn \__stex_smsmode_check_import_pair:nn {T,F,TF} {
1800   %\stex_debug:nn{import-pair}{\detokenize{#{1}}~{#{2}}}}
1801   \tl_if_empty:nTF{#{1}}{
1802     \prop_if_exist:NTF \l_stex_current_repository_prop
1803     {
1804       %\stex_debug:nn{import-pair}{in repository \prop_item:Nn \l_stex_current_repository_
1805       \prg_return_true:
1806     } {
1807       \seq_set_split:Nnn \l_tmpa_seq ? {#{2}}
1808       \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
1809       \tl_if_empty:NT \l_tmpa_tl {
1810         \seq_pop_left:NN \l_tmpa_seq \l_tmpa_tl
1811       }
1812       %\stex_debug:nn{import-pair}{\seq_use:Nn \l_tmpa_seq,~of~length~\seq_count:N \l_tmpa
1813       \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1
1814         \prg_return_true: \prg_return_false:
1815     }
1816   }\prg_return_true:
1817 }
1818
1819 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1820   \stex_filestack_push:n{#{1}}
1821   \seq_gclear:N \l__stex_smsmode_importmodules_seq
1822   \seq_gclear:N \l__stex_smsmode_sigmodules_seq
1823   % ----- new -----
1824   \__stex_smsmode_in_smsmode:nn{#{1}}{
1825     \let\importmodule\__stex_smsmode_importmodule:
1826     \let\stex_module_setup:nn\__stex_smsmode_module:nn
1827     \let\__stex_modules_begin_module:\relax
1828     \let\__stex_modules_end_module:\relax

```

```

1829 \seq_clear:N \g_stex_smsmode_allowedenvs_seq
1830 \exp_args:NNx \seq_put_right:Nn \g_stex_smsmode_allowedenvs_seq {\tl_to_str:n{smodule}}
1831 \tl_clear:N \g_stex_smsmode_allowedmacros_tl
1832 \tl_clear:N \g_stex_smsmode_allowedmacros_escape_tl
1833 \tl_put_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\importmodule}
1834 \everyeof{\q__stex_smsmode_break\noexpand}
1835 \expandafter\expandafter\expandafter
1836 \stex_smsmode_do:
1837 \csname @ @ input\endcsname "#1"\relax
1838
1839 \seq_map_inline:Nn \l__stex_smsmode_sigmodules_seq {
1840   \stex_filestack_push:n{##1}
1841   \expandafter\expandafter\expandafter
1842   \stex_smsmode_do:
1843   \csname @ @ input\endcsname "##1"\relax
1844   \stex_filestack_pop:
1845 }
1846 }
1847 % ----- new -----
1848 \__stex_smsmode_in_smsmode:nn{#1} {
1849   #2
1850   % ----- new -----
1851   \begingroup
1852   %\stex_debug:nn{smsmode}{Here:~\seq_use:Nn\l__stex_smsmode_importmodules_seq, }
1853   \seq_map_inline:Nn \l__stex_smsmode_importmodules_seq {
1854     \__stex_smsmode_check_import_pair:nnT ##1 { \begingroup
1855       \stex_import_module_uri:nn ##1
1856       \stex_import_require_module:nnnn
1857       \l_stex_import_ns_str
1858       \l_stex_import_archive_str
1859       \l_stex_import_path_str
1860       \l_stex_import_name_str \endgroup
1861     }
1862   }
1863   \endgroup
1864   \stex_debug:nn{smsmode}{Actually~loading~file~#1}
1865   % ----- new -----
1866   \everyeof{\q__stex_smsmode_break\noexpand}
1867   \expandafter\expandafter\expandafter
1868   \stex_smsmode_do:
1869   \csname @ @ input\endcsname "#1"\relax
1870 }
1871 \stex_filestack_pop:
1872 }

```

(End definition for `\stex_file_in_smsmode:nn`. This function is documented on page 75.)

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

```

1873 \cs_new_protected:Npn \stex_smsmode_do: {
1874   \stex_if_smsmode:T {
1875     \__stex_smsmode_do:w
1876   }
1877 }

```

```

1878 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1879   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
1880     \expandafter\if\expandafter\relax\noexpand#1
1881     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1882     \else\expandafter\__stex_smsmode_do:w\fi
1883   }{
1884     \__stex_smsmode_do:w %#1
1885   }
1886 }
1887 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1888   \cs_if_eq:NNTF #1 \q__stex_smsmode_break {
1889     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1890       #1\__stex_smsmode_do:w
1891     }{
1892       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1893         #1
1894       }{
1895         \cs_if_eq:NNTF \begin #1 {
1896           \__stex_smsmode_check_begin:n
1897         }{
1898           \cs_if_eq:NNTF \end #1 {
1899             \__stex_smsmode_check_end:n
1900           }{
1901             \__stex_smsmode_do:w
1902           }
1903         }
1904       }
1905     }
1906   }
1907 }
1908
1909 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1910   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1911     \begin{#1}
1912   }{
1913     \__stex_smsmode_do:w
1914   }
1915 }
1916 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1917   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1918     \end{#1}\__stex_smsmode_do:w
1919   }{
1920     \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1921   }
1922 }

```

(End definition for `\stex_smsmode_do:.` This function is documented on page 75.)

## 28.2 Inheritance

```

1923 <@@=stex_importmodule>

```

`\stex_import_module_uri:nn`

```

1924 \cs_new_protected:Nn \stex_import_module_uri:nn {

```



```

1925 \str_set:Nx \l_stex_import_archive_str { #1 }
1926 \str_set:Nn \l_stex_import_path_str { #2 }
1927
1928 \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1929 \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1930 \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1931
1932 \stex_modules_current_namespace:
1933 \bool_lazy_all:nTF {
1934   {\str_if_empty_p:N \l_stex_import_archive_str}
1935   {\str_if_empty_p:N \l_stex_import_path_str}
1936   {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1937 }{
1938   \str_set_eq:NN \l_stex_import_path_str \l_stex_module_subpath_str
1939   \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1940 }{
1941   \str_if_empty:NT \l_stex_import_archive_str {
1942     \prop_if_exist:NT \l_stex_current_repository_prop {
1943       \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1944     }
1945   }
1946   \str_if_empty:NTF \l_stex_import_archive_str {
1947     \str_if_empty:NF \l_stex_import_path_str {
1948       \stex_path_from_string:Nn \l_tmpb_seq {
1949         \l_stex_module_ns_str / .. / \l_stex_import_path_str
1950       }
1951       \str_set:Nx \l_stex_import_ns_str {\stex_path_to_string:N \l_tmpb_seq}
1952       \str_replace_once:Nnn \l_stex_import_ns_str {file:/{ } {file://}
1953     }
1954   }{
1955     \stex_require_repository:n \l_stex_import_archive_str
1956     \prop_get:cnN { c_stex_mathhub\_l_stex_import_archive_str_manifest_prop } { ns }
1957     \l_stex_import_ns_str
1958     \str_if_empty:NF \l_stex_import_path_str {
1959       \str_set:Nx \l_stex_import_ns_str {
1960         \l_stex_import_ns_str / \l_stex_import_path_str
1961       }
1962     }
1963   }
1964 }
1965 }

```

(End definition for `\stex_import_module_uri:nn`. This function is documented on page 76.)

```

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

```

(End definition for `\l_stex_import_name_str` and others. These variables are documented on page 76.)

```

\stex_import_require_module:nnnn {\<ns>} {\<archive-ID>} {\<path>} {\<name>}
1970 \cs_new_protected:Nn \stex_import_require_module:nnnn {

```

```

1971 \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1972
1973   \stex_debug:nn{requiremodule}{Here:\~1:~#1\~2:~#2\~3:~#3\~4:~#4}
1974
1975   \exp_args:NNxx \seq_set_split:Nnn \l_tmpa_seq {\tl_to_str:n{/}} {#4}
1976   \seq_get_left:NN \l_tmpa_seq \l_tmpc_str
1977
1978   %\stex_debug:nn{requiremodule}{Top-module:\l_tmpc_str}
1979
1980   % archive
1981   \str_set:Nx \l_tmpa_str { #2 }
1982   \str_if_empty:NTF \l_tmpa_str {
1983     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1984     \seq_put_right:Nn \l_tmpa_seq {..}
1985   } {
1986     \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1987     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1988     \seq_put_right:Nn \l_tmpa_seq { source }
1989   }
1990
1991   % path
1992   \str_set:Nx \l_tmpb_str { #3 }
1993   \str_if_empty:NTF \l_tmpb_str {
1994     \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / \l_tmpc_str }
1995
1996     \ltx@ifpackageloaded{babel} {
1997       \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1998         { \language } \l_tmpb_str {
1999         \msg_error:nnx{stex}{error/unknownlanguage}{\language}
2000       }
2001     } {
2002       \str_clear:N \l_tmpb_str
2003     }
2004
2005     \stex_debug:nn{modules}{Checking~a1~\l_tmpa_str.\l_tmpb_str.tex}
2006     \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2007       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2008     }{
2009       \stex_debug:nn{modules}{Checking~a2~\l_tmpa_str.tex}
2010       \IfFileExists{ \l_tmpa_str.tex }{
2011         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2012       }{
2013         % try english as default
2014         \stex_debug:nn{modules}{Checking~a3~\l_tmpa_str.en.tex}
2015         \IfFileExists{ \l_tmpa_str.en.tex }{
2016           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2017         }{
2018           \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2019         }
2020       }
2021     }
2022
2023   } {
2024     \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str

```

```

2025 \seq_concat:NNN \l_tmpb_seq \l_tmpa_seq \l_tmpb_seq
2026
2027 \ltx@ifpackageloaded{babel} {
2028   \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
2029     { \language } \l_tmpb_str {
2030       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
2031     }
2032   } {
2033     \str_clear:N \l_tmpb_str
2034   }
2035
2036 \stex_path_canonicalize:N \l_tmpb_seq
2037 \stex_path_to_string:NN \l_tmpb_seq \l_tmpa_str
2038
2039 \stex_debug:nn{modules}{Checking~b1~\l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex}
2040 \IfFileExists{ \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }{
2041   \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }
2042 }{
2043   \stex_debug:nn{modules}{Checking~b2~\l_tmpa_str/\l_tmpc_str.tex}
2044   \IfFileExists{ \l_tmpa_str/\l_tmpc_str.tex }{
2045     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.tex }
2046   }{
2047     % try english as default
2048     \stex_debug:nn{modules}{Checking~b3~\l_tmpa_str/\l_tmpc_str.en.tex}
2049     \IfFileExists{ \l_tmpa_str/\l_tmpc_str.en.tex }{
2050       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.en.tex }
2051     }{
2052       \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.\l_tmpb_str.tex}
2053       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2054         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2055       }{
2056         \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.tex}
2057         \IfFileExists{ \l_tmpa_str.tex }{
2058           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2059         }{
2060           % try english as default
2061           \stex_debug:nn{modules}{Checking~b5~\l_tmpa_str.en.tex}
2062           \IfFileExists{ \l_tmpa_str.en.tex }{
2063             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2064           }{
2065             \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2066           }
2067         }
2068       }
2069     }
2070   }
2071 }
2072
2073
2074 \str_if_eq:eeF{\g__stex_importmodule_file_str}{\seq_use:Nn \g_stex_currentfile_seq /}{
2075   \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
2076     \seq_clear:N \l_stex_all_modules_seq
2077     \str_clear:N \l_stex_current_module_str
2078     \str_set:Nx \l_tmpb_str { #2 }

```

```

2079     \str_if_empty:NF \l_tmpb_str {
2080       \stex_set_current_repository:n { #2 }
2081     }
2082     \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
2083   }
2084
2085   \stex_if_module_exists:nF { #1 ? #4 } {
2086     \msg_error:nnx{stex}{error/unknownmodule}{
2087       #1?#4~(in~file~\g__stex_importmodule_file_str)
2088     }
2089   }
2090 }
2091
2092 }
2093 \stex_activate_module:n { #1 ? #4 }
2094 }

```

(End definition for `\stex_import_require_module:nnnn`. This function is documented on page 76.)

## `\importmodule`

```

2095 \NewDocumentCommand \importmodule { 0{} m } {
2096   \stex_import_module_uri:nn { #1 } { #2 }
2097   \stex_debug:nn{modules}{Importing~module:~
2098     \l_stex_import_ns_str ? \l_stex_import_name_str
2099   }
2100   \stex_import_require_module:nnnn
2101   { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2102   { \l_stex_import_path_str } { \l_stex_import_name_str }
2103   \stex_if_smsmode:F {
2104     \stex_annotate_invisible:nnn
2105     {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
2106   }
2107   \exp_args:Nx \stex_add_to_current_module:n {
2108     \stex_import_require_module:nnnn
2109     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2110     { \l_stex_import_path_str } { \l_stex_import_name_str }
2111   }
2112   \exp_args:Nx \stex_add_import_to_current_module:n {
2113     \l_stex_import_ns_str ? \l_stex_import_name_str
2114   }
2115   \stex_smsmode_do:
2116   \ignorespacesandpars
2117 }
2118 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

(End definition for `\importmodule`. This function is documented on page 75.)

## `\usemodule`

```

2119 \NewDocumentCommand \usemodule { 0{} m } {
2120   \stex_if_smsmode:F {
2121     \stex_import_module_uri:nn { #1 } { #2 }
2122     \stex_import_require_module:nnnn
2123     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2124     { \l_stex_import_path_str } { \l_stex_import_name_str }
2125     \stex_annotate_invisible:nnn

```

```

2126     {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
2127   }
2128   \stex_smsmode_do:
2129   \ignorespacesandpars
2130 }

```

*(End definition for \usemodule. This function is documented on page 75.)*

```

2131 \cs_new_protected:Nn \stex_csl_to_imports:Nn {
2132   \tl_if_empty:nF{#2}{
2133     \clist_set:Nn \l_tmpa_clist {#2}
2134     \clist_map_inline:Nn \l_tmpa_clist {
2135       \tl_if_head_eq_charcode:nNTF {##1} [{
2136         #1 ##1
2137       } {
2138         #1{##1}
2139       }
2140     }
2141   }
2142 }
2143 \cs_generate_variant:Nn \stex_csl_to_imports:Nn {No}
2144
2145
2146 \</package>

```

## Chapter 29

# STEX -Symbols Implementation

```
2147 <*package>
2148
2149 %%%%%%%%%% symbols.dtx %%%%%%%%%%
2150
2151 Warnings and error messages
2152 \msg_new:nnn{stex}{error/wrongargs}{
2153   args~value~in~symbol~declaration~for~#1~
2154   needs~to~be~i,~a,~b~or~B,~but~#2~given
2155 }
2156 \msg_new:nnn{stex}{error/unknownsymbol}{
2157   No~symbol~#1~found!
2158 }
2159 \msg_new:nnn{stex}{error/seqlength}{
2160   Expected~#1~arguments;~got~#2!
2161 }
2162 \msg_new:nnn{stex}{error/unknownnotation}{
2163   Unknown~notation~#1~for~#2!
2164 }
```

### 29.1 Symbol Declarations

```
2164 <@@=stex_symdecl>
\stex_all_symbols:n Map over all available symbols
2165 \cs_new_protected:Nn \stex_all_symbols:n {
2166   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
2167   \seq_map_inline:Nn \l_stex_all_modules_seq {
2168     \seq_map_inline:cn{c_stex_module_##1_constants}{
2169       \__stex_symdecl_all_symbols_cs{##1?####1}
2170     }
2171   }
2172 }
```

(End definition for `\stex_all_symbols:n`. This function is documented on page 78.)

## `\STEXsymbol`

```
2173 \NewDocumentCommand \STEXsymbol { m } {  
2174   \stex_get_symbol:n { #1 }  
2175   \exp_args:No  
2176   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }  
2177 }
```

(End definition for `\STEXsymbol`. This function is documented on page 79.)

`symdecl` arguments:

```
2178 \keys_define:nn { stex / symdecl } {  
2179   name      .str_set_x:N = \l_stex_symdecl_name_str ,  
2180   local     .bool_set:N = \l_stex_symdecl_local_bool ,  
2181   args      .str_set_x:N = \l_stex_symdecl_args_str ,  
2182   type      .tl_set:N = \l_stex_symdecl_type_tl ,  
2183   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,  
2184   align     .str_set:N = \l_stex_symdecl_align_str , % TODO(?)  
2185   gfc       .str_set:N = \l_stex_symdecl_gfc_str , % TODO(?)  
2186   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)  
2187   def       .tl_set:N = \l_stex_symdecl_definiens_tl ,  
2188   reorder   .str_set_x:N = \l_stex_symdecl_reorder_str ,  
2189   assoc     .choices:nn =  
2190     {bin,binl,binr,pre,conj,pwconj}  
2191     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}  
2192 }  
2193  
2194 \bool_new:N \l_stex_symdecl_make_macro_bool  
2195  
2196 \cs_new_protected:Nn \__stex_symdecl_args:n {  
2197   \str_clear:N \l_stex_symdecl_name_str  
2198   \str_clear:N \l_stex_symdecl_args_str  
2199   \str_clear:N \l_stex_symdecl_deprecate_str  
2200   \str_clear:N \l_stex_symdecl_reorder_str  
2201   \str_clear:N \l_stex_symdecl_assoctype_str  
2202   \bool_set_false:N \l_stex_symdecl_local_bool  
2203   \tl_clear:N \l_stex_symdecl_type_tl  
2204   \tl_clear:N \l_stex_symdecl_definiens_tl  
2205  
2206   \keys_set:nn { stex / symdecl } { #1 }  
2207 }
```

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

```
2208  
2209 \NewDocumentCommand \symdecl { s m O{} } {  
2210   \__stex_symdecl_args:n { #3 }  
2211   \IfBooleanTF #1 {  
2212     \bool_set_false:N \l_stex_symdecl_make_macro_bool  
2213   } {  
2214     \bool_set_true:N \l_stex_symdecl_make_macro_bool  
2215   }  
2216   \stex_symdecl_do:n { #2 }  
2217   \stex_smsmode_do:  
2218 }
```

```

2219
2220 \cs_new_protected:Nn \stex_symdecl_do:nn {
2221   \__stex_symdecl_args:n{#1}
2222   \bool_set_false:N \l_stex_symdecl_make_macro_bool
2223   \stex_symdecl_do:n{#2}
2224 }
2225
2226 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

(End definition for \symdecl. This function is documented on page 77.)

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

```

2227 \cs_new_protected:Nn \stex_symdecl_do:n {
2228   \stex_if_in_module:F {
2229     % TODO throw error? some default namespace?
2230   }
2231
2232   \str_if_empty:NT \l_stex_symdecl_name_str {
2233     \str_set:Nx \l_stex_symdecl_name_str { #1 }
2234   }
2235
2236   \prop_if_exist:cT { l_stex_symdecl_
2237     \l_stex_current_module_str ?
2238     \l_stex_symdecl_name_str
2239     _prop
2240   }{
2241     % TODO throw error (beware of circular dependencies)
2242   }
2243
2244   \prop_clear:N \l_tmpa_prop
2245   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
2246   \seq_clear:N \l_tmpa_seq
2247   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
2248   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
2249
2250   \str_if_empty:NT \l_stex_symdecl_deprecate_str {
2251     \str_if_empty:NF \l_stex_module_deprecate_str {
2252       \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
2253     }
2254   }
2255   \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
2256
2257   \exp_args:No \stex_add_constant_to_current_module:n {
2258     \l_stex_symdecl_name_str
2259   }
2260
2261   % arity/args
2262   \int_zero:N \l_tmpb_int
2263
2264   \bool_set_true:N \l_tmpa_bool
2265   \str_map_inline:Nn \l_stex_symdecl_args_str {
2266     \token_case_meaning:NnF ##1 {
2267       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2268       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }

```



```

2269     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2270     {\tl_to_str:n a} {
2271         \bool_set_false:N \l_tmpa_bool
2272         \int_incr:N \l_tmpb_int
2273     }
2274     {\tl_to_str:n B} {
2275         \bool_set_false:N \l_tmpa_bool
2276         \int_incr:N \l_tmpb_int
2277     }
2278 }{
2279     \msg_error:nnxx{stex}{error/wrongargs}{
2280         \l_stex_current_module_str ?
2281         \l_stex_symdecl_name_str
2282     }{##1}
2283 }
2284 }
2285 \bool_if:NTF \l_tmpa_bool {
2286     % possibly numeric
2287     \str_if_empty:NTF \l_stex_symdecl_args_str {
2288         \prop_put:Nnn \l_tmpa_prop { args } {}
2289         \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2290     }{
2291         \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
2292         \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2293         \str_clear:N \l_tmpa_str
2294         \int_step_inline:nn \l_tmpa_int {
2295             \str_put_right:Nn \l_tmpa_str i
2296         }
2297         \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
2298     }
2299 } {
2300     \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2301     \prop_put:Nnx \l_tmpa_prop { arity }
2302     { \str_count:N \l_stex_symdecl_args_str }
2303 }
2304 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2305
2306 \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
2307     \prop_put:Nnx \l_tmpa_prop { defined }{ false }
2308 }{
2309     \prop_put:Nnx \l_tmpa_prop { defined }{ true }
2310 }
2311
2312 % semantic macro
2313
2314 \bool_if:NT \l_stex_symdecl_make_macro_bool {
2315     \exp_args:Nx \stex_do_up_to_module:n {
2316         \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2317             \l_stex_current_module_str ? \l_stex_symdecl_name_str
2318         }}
2319     }
2320 }
2321
2322 \stex_debug:nn{symbols}{New~symbol:~

```

```

2323 \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2324 Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2325 Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
2326 Definiens:~\exp_not:o {\l_stex_symdecl_definiens_tl}
2327 }
2328
2329 % circular dependencies require this:
2330 \stex_if_do_html:T {
2331   \stex_annotate_invisible:nnn {symdecl} {
2332     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2333   } {
2334     \tl_if_empty:NF \l_stex_symdecl_type_tl {
2335       \stex_annotate_invisible:nnn{type}{\l_stex_symdecl_type_tl$}
2336     }
2337     \stex_annotate_invisible:nnn{args}{\l_stex_symdecl_args_tl$}
2338     \prop_item:Nn \l_tmpa_prop { args }
2339   }
2340   \stex_annotate_invisible:nnn{macroname}{\l_stex_symdecl_macroname_tl$}
2341   \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2342     \stex_annotate_invisible:nnn{definiens}{\l_stex_symdecl_definiens_tl$}
2343   }
2344   \str_if_empty:NF \l_stex_symdecl_assoc_type_str {
2345     \stex_annotate_invisible:nnn{assoc_type}{\l_stex_symdecl_assoc_type_str$}
2346   }
2347   \str_if_empty:NF \l_stex_symdecl_reorder_str {
2348     \stex_annotate_invisible:nnn{reorder_args}{\l_stex_symdecl_reorder_str$}
2349   }
2350 }
2351 }
2352 }
2353 \prop_if_exist:cF {
2354   \l_stex_symdecl_
2355   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2356   _prop
2357 } {
2358   \bool_if:NTF \l_stex_symdecl_local_bool \stex_do_up_to_module:x \stex_execute_in_module:
2359   \__stex_symdecl_restore_symbol:nnnnnnn
2360     {\l_stex_symdecl_name_str}
2361     { \prop_item:Nn \l_tmpa_prop {args} }
2362     { \prop_item:Nn \l_tmpa_prop {arity} }
2363     { \prop_item:Nn \l_tmpa_prop {assocs} }
2364     { \prop_item:Nn \l_tmpa_prop {defined} }
2365     {\bool_if:NT \l_stex_symdecl_make_macro_bool {#1} }
2366     {\l_stex_current_module_str}
2367 }
2368 }
2369 }
2370 \cs_new_protected:Nn \__stex_symdecl_restore_symbol:nnnnnnn {
2371   \prop_clear:N \l_tmpa_prop
2372   \prop_put:Nnn \l_tmpa_prop { module } { #7 }
2373   \prop_put:Nnn \l_tmpa_prop { name } { #1 }
2374   \prop_put:Nnn \l_tmpa_prop { args } { #2 }
2375   \prop_put:Nnn \l_tmpa_prop { arity } { #3 }
2376   \prop_put:Nnn \l_tmpa_prop { assocs } { #4 }

```

```

2377 \prop_put:Nnn \l_tmpa_prop { defined } { #5 }
2378 \tl_if_empty:nF{#6}{
2379   \tl_set:cx{#6}{\stex_invoke_symbol:n{\detokenize{#7 ? #1}}}
2380 }
2381 \prop_set_eq:cN{l_stex_symdecl_ \detokenize{#7 ? #1} _prop}\l_tmpa_prop
2382 \seq_clear:c{l_stex_symdecl_ \detokenize{#7 ? #1} _notations}
2383 }

```

(End definition for `\stex_symdecl_do:n`. This function is documented on page 78.)

## `\stex_get_symbol:n`

```

2384 \str_new:N \l_stex_get_symbol_uri_str
2385
2386 \cs_new_protected:Nn \stex_get_symbol:n {
2387   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2388     \tl_set:Nn \l_tmpa_tl { #1 }
2389     \__stex_symdecl_get_symbol_from_cs:
2390   }{
2391     % argument is a string
2392     % is it a command name?
2393     \cs_if_exist:cTF { #1 }{
2394       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2395       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2396       \str_if_empty:NTF \l_tmpa_str {
2397         \exp_args:Nx \cs_if_eq:NNTF {
2398           \tl_head:N \l_tmpa_tl
2399         } \stex_invoke_symbol:n {
2400           \__stex_symdecl_get_symbol_from_cs:
2401         }{
2402           \__stex_symdecl_get_symbol_from_string:n { #1 }
2403         }
2404       } {
2405         \__stex_symdecl_get_symbol_from_string:n { #1 }
2406       }
2407     }{
2408       % argument is not a command name
2409       \__stex_symdecl_get_symbol_from_string:n { #1 }
2410       % \l_stex_all_symbols_seq
2411     }
2412   }
2413   \str_if_eq:eeF {
2414     \prop_item:cn {
2415       l_stex_symdecl_\l_stex_get_symbol_uri_str _prop
2416     }{ deprecate }
2417   }{
2418     \msg_warning:nxxx{stex}{warning/deprecated}{
2419       Symbol~\l_stex_get_symbol_uri_str
2420     }{
2421       \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{ deprecate }
2422     }
2423   }
2424 }
2425
2426 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {

```

```

2427 \tl_set:Nn \l_tmpa_tl {
2428   \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2429 }
2430 \str_set:Nn \l_tmpa_str { #1 }
2431
2432 %\int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2433
2434 \str_if_in:NnTF \l_tmpa_str ? {
2435   \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
2436   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
2437   \str_set:Nx \l_tmpb_str {\seq_use:Nn \l_tmpa_seq ?}
2438 }{
2439   \str_clear:N \l_tmpb_str
2440 }
2441 \str_if_empty:NNTF \l_tmpb_str {
2442   \seq_map_inline:Nn \l_stex_all_modules_seq {
2443     \seq_map_inline:cn{c_stex_module_###1_constants}{
2444       \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2445         \seq_map_break:n{\seq_map_break:n{
2446           \tl_set:Nn \l_tmpa_tl {
2447             \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2448           }
2449         }}
2450       }
2451     }
2452   }
2453 }{
2454   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpb_str }
2455   \seq_map_inline:Nn \l_stex_all_modules_seq {
2456     \str_if_eq:eeT{ \l_tmpb_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2457       \seq_map_inline:cn{c_stex_module_###1_constants}{
2458         \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2459           \seq_map_break:n{\seq_map_break:n{
2460             \tl_set:Nn \l_tmpa_tl {
2461               \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2462             }
2463           }}
2464         }
2465       }
2466     }
2467   }
2468 }
2469
2470 \l_tmpa_tl
2471 }
2472
2473 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
2474   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2475     { \tl_tail:N \l_tmpa_tl }
2476   \tl_if_single:NNTF \l_tmpa_tl {
2477     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2478       \exp_after:wN \str_set:Nn \exp_after:wN
2479         \l_stex_get_symbol_uri_str \l_tmpa_tl
2480     }{

```

```

2481     % TODO
2482     % tail is not a single group
2483   }
2484 }{
2485   % TODO
2486   % tail is not a single group
2487 }
2488 }

```

(End definition for `\stex_get_symbol:n`. This function is documented on page 78.)

## 29.2 Notations

```

2489 <@@=stex_notation>

notation arguments:
2490 \keys_define:nn { stex / notation } {
2491   % lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2492   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2493   prec     .str_set_x:N = \l__stex_notation_prec_str ,
2494   op       .tl_set:N = \l__stex_notation_op_tl ,
2495   primary .bool_set:N = \l__stex_notation_primary_bool ,
2496   primary .default:n = {true} ,
2497   unknown .code:n = \str_set:Nx
2498     \l__stex_notation_variant_str \l_keys_key_str
2499 }
2500
2501 \cs_new_protected:Nn \_stex_notation_args:n {
2502   % \str_clear:N \l__stex_notation_lang_str
2503   \str_clear:N \l__stex_notation_variant_str
2504   \str_clear:N \l__stex_notation_prec_str
2505   \tl_clear:N \l__stex_notation_op_tl
2506   \bool_set_false:N \l__stex_notation_primary_bool
2507
2508   \keys_set:nn { stex / notation } { #1 }
2509 }

\notation

2510 \NewDocumentCommand \notation { s m 0{}} {
2511   \_stex_notation_args:n { #3 }
2512   \tl_clear:N \l_stex_symdecl_definiens_tl
2513   \stex_get_symbol:n { #2 }
2514   \tl_set:Nn \l_stex_notation_after_do_tl {
2515     \__stex_notation_final:
2516     \IfBooleanTF#1{
2517       \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2518     }{}
2519     \stex_smsmode_do:\ignorespacesandpars
2520   }
2521   \stex_notation_do:nnnnn
2522   { \prop_item:cn {\l_stex_symdecl\_l_stex_get_symbol_uri_str _prop } { args } }
2523   { \prop_item:cn { \l_stex_symdecl\_l_stex_get_symbol_uri_str _prop } { arity } }
2524   { \l__stex_notation_variant_str }
2525   { \l__stex_notation_prec_str }

```

```

2526 }
2527 \stex_deactivate_macro:Nn \notation {module-environments}

```

(End definition for \notation. This function is documented on page 78.)

\stex\_notation\_do:nnnnn

```

2528 \seq_new:N \l__stex_notation_precedences_seq
2529 \tl_new:N \l__stex_notation_opprec_tl
2530 \int_new:N \l__stex_notation_currarg_int
2531 \tl_new:N \stex_symbol_after_invokation_tl
2532
2533 \cs_new_protected:Nn \stex_notation_do:nnnnn {
2534   \let\l_stex_current_symbol_str\relax
2535   \seq_clear:N \l__stex_notation_precedences_seq
2536   \tl_clear:N \l__stex_notation_opprec_tl
2537   \str_set:Nx \l__stex_notation_args_str { #1 }
2538   \str_set:Nx \l__stex_notation_arity_str { #2 }
2539   \str_set:Nx \l__stex_notation_suffix_str { #3 }
2540   \str_set:Nx \l__stex_notation_prec_str { #4 }
2541
2542   % precedences
2543   \str_if_empty:NTF \l__stex_notation_prec_str {
2544     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2545       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2546     }{
2547       \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2548     }
2549   } {
2550     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2551       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2552       \int_step_inline:nn { \l__stex_notation_arity_str } {
2553         \exp_args:NNo
2554         \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2555       }
2556     }{
2557       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2558       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2559         \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2560         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2561           \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2562             \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2563           \seq_map_inline:Nn \l_tmpa_seq {
2564             \seq_put_right:Nn \l_tmpb_seq { ##1 }
2565           }
2566         }
2567       }{
2568         \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2569           \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2570         }{
2571           \tl_set:No \l__stex_notation_opprec_tl { 0 }
2572         }
2573       }
2574     }
2575   }

```

```

2576 \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2577 \int_step_inline:nn { \l__stex_notation_arity_str } {
2578   \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
2579     \exp_args:NNo
2580     \seq_put_right:No \l__stex_notation_precedences_seq {
2581       \l__stex_notation_opprec_tl
2582     }
2583   }
2584 }
2585 }
2586 \tl_clear:N \l_stex_notation_dummyargs_tl
2587
2588 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2589   \exp_args:NNe
2590   \cs_set:Npn \l_stex_notation_macrocode_cs {
2591     \stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2592     { \l__stex_notation_suffix_str }
2593     { \l__stex_notation_opprec_tl }
2594     { \exp_not:n { #5 } }
2595   }
2596   \l_stex_notation_after_do_tl
2597 }{
2598   \str_if_in:NnTF \l__stex_notation_args_str b {
2599     \exp_args:Nne \use:nn
2600     {
2601       \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2602       \cs_set:Npn \l__stex_notation_arity_str } { {
2603         \stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2604         { \l__stex_notation_suffix_str }
2605         { \l__stex_notation_opprec_tl }
2606         { \exp_not:n { #5 } }
2607       }}
2608   }{
2609     \str_if_in:NnTF \l__stex_notation_args_str B {
2610       \exp_args:Nne \use:nn
2611       {
2612         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2613         \cs_set:Npn \l__stex_notation_arity_str } { {
2614           \stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2615           { \l__stex_notation_suffix_str }
2616           { \l__stex_notation_opprec_tl }
2617           { \exp_not:n { #5 } }
2618         } }
2619     }{
2620       \exp_args:Nne \use:nn
2621       {
2622         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2623         \cs_set:Npn \l__stex_notation_arity_str } { {
2624           \stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2625           { \l__stex_notation_suffix_str }
2626           { \l__stex_notation_opprec_tl }
2627           { \exp_not:n { #5 } }
2628         } }
2629     }

```

```

2630     }
2631
2632     \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2633     \int_zero:N \l__stex_notation_currarg_int
2634     \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2635     \__stex_notation_arguments:
2636   }
2637 }

```

(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

```

2638 \cs_new_protected:Nn \__stex_notation_arguments: {
2639   \int_incr:N \l__stex_notation_currarg_int
2640   \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2641     \l_stex_notation_after_do_tl
2642   }{
2643     \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2644     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2645     \str_if_eq:VnTF \l_tmpa_str a {
2646       \__stex_notation_argument_assoc:nn{a}
2647     }{
2648       \str_if_eq:VnTF \l_tmpa_str B {
2649         \__stex_notation_argument_assoc:nn{B}
2650       }{
2651         \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpb_str
2652         \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2653           { \stex_term_math_arg:nnn
2654             { \l_tmpa_str\int_use:N \l__stex_notation_currarg_int }
2655             { \l_tmpb_str }
2656             { ###\int_use:N \l__stex_notation_currarg_int }
2657           }
2658         }
2659         \__stex_notation_arguments:
2660       }
2661     }
2662   }
2663 }

```

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

\\_\_stex\_notation\_argument\_assoc:nn

```

2664 \cs_new_protected:Nn \__stex_notation_argument_assoc:nn {
2665
2666   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2667     {\l__stex_notation_arity_str}{
2668     #2
2669   }
2670   \int_zero:N \l_tmpa_int
2671   \tl_clear:N \l_tmpa_tl
2672   \str_map_inline:Nn \l__stex_notation_args_str {
2673     \int_incr:N \l_tmpa_int
2674     \tl_put_right:Nx \l_tmpa_tl {
2675       \str_if_eq:nnTF {##1}{a}{ { } }{ }

```



```

2676 \str_if_eq:nnTF {##1}{B}{ } }{
2677   {\_stex_term_arg:nn{##1\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa
2678   }
2679 }
2680 }
2681 }
2682 \exp_after:wN\exp_after:wN\exp_after:wN \def
2683 \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2684 \exp_after:wN\exp_after:wN\exp_after:wN ##
2685 \exp_after:wN\exp_after:wN\exp_after:wN 1
2686 \exp_after:wN\exp_after:wN\exp_after:wN ##
2687 \exp_after:wN\exp_after:wN\exp_after:wN 2
2688 \exp_after:wN\exp_after:wN\exp_after:wN {
2689   \exp_after:wN \exp_after:wN \exp_after:wN
2690   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2691     \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2692   }
2693 }
2694
2695 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2696 \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
2697   \_stex_term_math_assoc_arg:nnnn
2698   { #1\int_use:N \l__stex_notation_currarg_int }
2699   { \l_tmpa_str }
2700   { #####\int_use:N \l__stex_notation_currarg_int }
2701   { \l_tmpa_cs {####1} {####2} }
2702 } }
2703 \__stex_notation_arguments:
2704 }

```

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

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

```

2705 \cs_new_protected:Nn \_stex_notation_restore_notation:nnnnn {
2706   \cs_generate_from_arg_count:cNnn{stex_notation_\detokenize{#1} \c_hash_str \detokenize{#2}}
2707   \cs_set_nopar:Npn {#3}{#4}
2708   \tl_if_empty:nF {#5}{
2709     \tl_set:cn{stex_op_notation_\detokenize{#1} \c_hash_str \detokenize{#2}_cs}{\comp{ #5 }
2710   }
2711   \seq_if_exist:cT { l_stex_symdecl_\detokenize{#1} _notations }{
2712     \seq_put_right:cx { l_stex_symdecl_\detokenize{#1} _notations } { \detokenize{#2} }
2713   }
2714 }
2715
2716 \cs_new_protected:Nn \_stex_notation_final: {
2717
2718   \stex_execute_in_module:x {
2719     \_stex_notation_restore_notation:nnnnn
2720     {\l_stex_get_symbol_uri_str}
2721     {\l_stex_notation_suffix_str}
2722     {\l__stex_notation_arity_str}
2723     {
2724       \exp_after:wN \exp_after:wN \exp_after:wN
2725       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN

```

```

2726     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2727   }
2728   {\exp_args:No \exp_not:n \l__stex_notation_op_tl }
2729 }
2730
2731 \stex_debug:nn{symbols}{
2732   Notation~\l__stex_notation_suffix_str
2733   ~for~\l_stex_get_symbol_uri_str^^J
2734   Operator~precedence:~\l__stex_notation_opprec_tl^^J
2735   Argument~precedences:~
2736     \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2737   Notation: \cs_meaning:c {
2738     stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2739     \l__stex_notation_suffix_str
2740     _cs
2741   }
2742 }
2743 % HTML annotations
2744 \stex_if_do_html:T {
2745   \stex_annotate_invisible:nnn { notation }
2746   { \l_stex_get_symbol_uri_str } {
2747     \stex_annotate_invisible:nnn { notationfragment }
2748     { \l__stex_notation_suffix_str }{}
2749     \stex_annotate_invisible:nnn { precedence }
2750     { \l__stex_notation_prec_str }{}
2751
2752     \int_zero:N \l_tmpa_int
2753     \str_set_eq:NN \l__stex_notation_remaining_args_str \l_stex_notation_args_str
2754     \tl_clear:N \l_tmpa_tl
2755     \int_step_inline:nn { \l__stex_notation_arity_str }{
2756       \int_incr:N \l_tmpa_int
2757       \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2758       \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_rema
2759       \str_if_eq:VnTF \l_tmpb_str a {
2760         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2761           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2762           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2763         } }
2764       }{
2765         \str_if_eq:VnTF \l_tmpb_str B {
2766           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2767             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2768             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2769           } }
2770         }{
2771           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2772             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
2773           } }
2774       }
2775     }
2776   }
2777   \stex_annotate_invisible:nnn { notationcomp }{}{
2778     \str_set:Nx \l_stex_current_symbol_str {\l_stex_get_symbol_uri_str }
2779     $ \exp_args:Nno \use:nn { \use:c {

```

```

2780         stex_notation_ \l_stex_current_symbol_str
2781         \c_hash_str \l__stex_notation_suffix_str _cs
2782     } } { \l_tmpa_tl } $
2783   }
2784 }
2785 }
2786 }

```

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

## \setnotation

```

2787 \keys_define:nn { stex / setnotation } {
2788   % lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2789   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2790   unknown .code:n      = \str_set:Nx
2791     \l__stex_notation_variant_str \l_keys_key_str
2792 }
2793
2794 \cs_new_protected:Nn \stex_setnotation_args:n {
2795   % \str_clear:N \l__stex_notation_lang_str
2796   \str_clear:N \l__stex_notation_variant_str
2797   \keys_set:nn { stex / setnotation } { #1 }
2798 }
2799
2800 \cs_new_protected:Nn \__stex_notation_setnotation:nn {
2801   \seq_if_exist:cT{l_stex_symdecl_#1_notations}{
2802     \seq_remove_all:cn { l_stex_symdecl_#1_notations }{ #2 }
2803     \seq_put_left:cn { l_stex_symdecl_#1_notations }{ #2 }
2804   }
2805 }
2806
2807 \cs_new_protected:Nn \stex_setnotation:n {
2808   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1_notations }
2809     { \l__stex_notation_variant_str }{
2810     \stex_execute_in_module:x{ \__stex_notation_setnotation:nn {#1}{\l__stex_notation_vari
2811     \stex_debug:nn {notations}{
2812       Setting~default~notation~
2813       {\l__stex_notation_variant_str }~for~
2814       #1 \\
2815       \expandafter\meaning\csname
2816       l_stex_symdecl_#1_notations\endcsname
2817     }
2818   }{
2819     \msg_error:nnxx{stex}{unknownnotation}{\l__stex_notation_variant_str}{#1}
2820   }
2821 }
2822
2823 \NewDocumentCommand \setnotation {m m} {
2824   \stex_get_symbol:n { #1 }
2825   \stex_setnotation_args:n { #2 }
2826   \stex_setnotation:n{\l_stex_get_symbol_uri_str}
2827   \stex_smsmode_do:\ignorespacesandpars
2828 }
2829

```

```

2830 \cs_new_protected:Nn \stex_copy_notations:nn {
2831   \stex_debug:nn {notations}{
2832     Copying~notations~from~#2~to~#1\\
2833     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
2834   }
2835   \tl_clear:N \l_tmpa_tl
2836   \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2837     \tl_put_right:Nn \l_tmpa_tl { {##### #1} }
2838   }
2839   \seq_map_inline:cn {l_stex_symdecl_#2_notations}{
2840     \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2841     \edef \l_tmpa_tl {
2842       \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2843       \exp_after:wN\exp_after:wN\exp_after:wN {
2844         \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2845       }
2846     }
2847
2848     \exp_after:wN \def \exp_after:wN \l_tmpa_tl
2849     \exp_after:wN #####\exp_after:wN 1 \exp_after:wN #####\exp_after:wN 2
2850     \exp_after:wN { \l_tmpa_tl }
2851
2852     \edef \l_tmpa_tl {
2853       \exp_after:wN \exp_not:n \exp_after:wN {
2854         \l_tmpa_tl {##### 1}{##### 2}
2855       }
2856     }
2857
2858     \stex_execute_in_module:x {
2859       \__stex_notation_restore_notation:nnnnn
2860       {#1}{##1}
2861       { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }
2862       { \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl} }
2863       {
2864         \cs_if_exist:cT{stex_op_notation_ #2\c_hash_str ##1 _cs}{
2865           \exp_args:NNo\exp_args:No\exp_not:n{\csname stex_op_notation_ #2\c_hash_str ##1
2866             }
2867         }
2868       }
2869     }
2870   }
2871
2872   \NewDocumentCommand \copynotation {m m} {
2873     \stex_get_symbol:n { #1 }
2874     \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2875     \stex_get_symbol:n { #2 }
2876     \exp_args:Noo
2877     \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2878     \stex_smsmode_do:\ignorespacesandpars
2879   }
2880

```

(End definition for \setnotation. This function is documented on page 19.)

**\symdef**

```
2881 \keys_define:nn { stex / symdef } {
2882   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2883   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2884   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2885   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
2886   def       .tl_set:N    = \l_stex_symdecl_definiens_tl ,
2887   reorder   .str_set_x:N = \l_stex_symdecl_reorder_str ,
2888   op        .tl_set:N    = \l__stex_notation_op_tl ,
2889   % lang     .str_set_x:N = \l__stex_notation_lang_str ,
2890   variant   .str_set_x:N = \l__stex_notation_variant_str ,
2891   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2892   assoc     .choices:nn =
2893     {bin,binl,binr,pre,conj,pwconj}
2894     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
2895   unknown   .code:n      = \str_set:Nx
2896     \l__stex_notation_variant_str \l_keys_key_str
2897 }
2898
2899 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2900   \str_clear:N \l_stex_symdecl_name_str
2901   \str_clear:N \l_stex_symdecl_args_str
2902   \str_clear:N \l_stex_symdecl_assoctype_str
2903   \str_clear:N \l_stex_symdecl_reorder_str
2904   \bool_set_false:N \l_stex_symdecl_local_bool
2905   \tl_clear:N \l_stex_symdecl_type_tl
2906   \tl_clear:N \l_stex_symdecl_definiens_tl
2907   % \str_clear:N \l__stex_notation_lang_str
2908   \str_clear:N \l__stex_notation_variant_str
2909   \str_clear:N \l__stex_notation_prec_str
2910   \tl_clear:N \l__stex_notation_op_tl
2911
2912   \keys_set:nn { stex / symdef } { #1 }
2913 }
2914
2915 \NewDocumentCommand \symdef { m O{} } {
2916   \__stex_notation_symdef_args:n { #2 }
2917   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2918   \stex_symdecl_do:n { #1 }
2919   \tl_set:Nn \l_stex_notation_after_do_tl {
2920     \__stex_notation_final:
2921     \stex_smsmode_do:\ignorespacesandpars
2922   }
2923   \str_set:Nx \l_stex_get_symbol_uri_str {
2924     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2925   }
2926   \exp_args:Nx \stex_notation_do:nnnnn
2927     { \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str _prop } { args } }
2928     { \prop_item:cn { l_stex_symdecl\l_stex_get_symbol_uri_str _prop } { arity } }
2929     { \l__stex_notation_variant_str }
2930     { \l__stex_notation_prec_str }
2931 }
2932 \stex_deactivate_macro:Nn \symdef {module~environments}
```

(End definition for `\symdef`. This function is documented on page 78.)

## 29.3 Variables

```

2933 <@@=stex_variables>
2934
2935 \keys_define:nn { stex / vardef } {
2936   name      .str_set_x:N = \l__stex_variables_name_str ,
2937   args      .str_set_x:N = \l__stex_variables_args_str ,
2938   type      .tl_set:N    = \l__stex_variables_type_tl ,
2939   def       .tl_set:N    = \l__stex_variables_def_tl ,
2940   op        .tl_set:N    = \l__stex_variables_op_tl ,
2941   prec      .str_set_x:N = \l__stex_variables_prec_str ,
2942   assoc     .choices:nn =
2943     {bin,binl,binr,pre,conj,pwconj}
2944     {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
2945   bind      .choices:nn =
2946     {forall,exists}
2947     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2948 }
2949
2950 \cs_new_protected:Nn \__stex_variables_args:n {
2951   \str_clear:N \l__stex_variables_name_str
2952   \str_clear:N \l__stex_variables_args_str
2953   \str_clear:N \l__stex_variables_prec_str
2954   \str_clear:N \l__stex_variables_assoctype_str
2955   \str_clear:N \l__stex_variables_bind_str
2956   \tl_clear:N \l__stex_variables_type_tl
2957   \tl_clear:N \l__stex_variables_def_tl
2958   \tl_clear:N \l__stex_variables_op_tl
2959
2960   \keys_set:nn { stex / vardef } { #1 }
2961 }
2962
2963 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{} } {
2964   \__stex_variables_args:n {#2}
2965   \str_if_empty:NT \l__stex_variables_name_str {
2966     \str_set:Nx \l__stex_variables_name_str { #1 }
2967   }
2968   \prop_clear:N \l_tmpa_prop
2969   \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
2970
2971   \int_zero:N \l_tmpb_int
2972   \bool_set_true:N \l_tmpa_bool
2973   \str_map_inline:Nn \l__stex_variables_args_str {
2974     \token_case_meaning:NnF ##1 {
2975       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2976       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2977       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2978       {\tl_to_str:n a} {
2979         \bool_set_false:N \l_tmpa_bool
2980         \int_incr:N \l_tmpb_int
2981       }

```

```

2982     {\tl_to_str:n B} {
2983       \bool_set_false:N \l_tmpa_bool
2984       \int_incr:N \l_tmpb_int
2985     }
2986   }{
2987     \msg_error:nxxx{stex}{error/wrongargs}{
2988       variable~\l__stex_variables_name_str
2989     }{##1}
2990   }
2991 }
2992 \bool_if:NTF \l_tmpa_bool {
2993   % possibly numeric
2994   \str_if_empty:NTF \l__stex_variables_args_str {
2995     \prop_put:Nnn \l_tmpa_prop { args } {}
2996     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2997   }{
2998     \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
2999     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
3000     \str_clear:N \l_tmpa_str
3001     \int_step_inline:nn \l_tmpa_int {
3002       \str_put_right:Nn \l_tmpa_str i
3003     }
3004     \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
3005     \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3006   }
3007 } {
3008   \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3009   \prop_put:Nnx \l_tmpa_prop { arity }
3010     { \str_count:N \l__stex_variables_args_str }
3011 }
3012 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
3013 \tl_set:cx { #1 }{ \stex_invoke_variable:n { \l__stex_variables_name_str } }
3014
3015 \prop_set_eq:cN { l_stex_variable_\l__stex_variables_name_str _prop } \l_tmpa_prop
3016
3017 \tl_if_empty:NF \l__stex_variables_op_tl {
3018   \cs_set:cpx {
3019     stex_var_op_notation_\l__stex_variables_name_str _cs
3020   } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
3021 }
3022
3023 \tl_set:Nn \l_stex_notation_after_do_tl {
3024   \exp_args:Nne \use:nn {
3025     \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str _cs }
3026     \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
3027   } {{
3028     \exp_after:wN \exp_after:wN \exp_after:wN
3029     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
3030     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_symbol
3031   }}
3032 \stex_if_do_html:T {
3033   \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
3034     \stex_annotate_invisible:nnn { precedence }
3035     { \l__stex_variables_prec_str }{}

```

```

3036 \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{}}{\$ \l
3037 \stex_annotate_invisible:nnn{args}{}}{\l__stex_variables_args_str }
3038 \stex_annotate_invisible:nnn{macroname}{#1}{}
3039 \tl_if_empty:NF \l__stex_variables_def_tl {
3040 \stex_annotate_invisible:nnn{definiens}{}
3041 {\$ \l__stex_variables_def_tl$}
3042 }
3043 \str_if_empty:NF \l__stex_variables_assoctype_str {
3044 \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{}
3045 }
3046 \str_if_empty:NF \l__stex_variables_bind_str {
3047 \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{}
3048 }
3049 \int_zero:N \l_tmpa_int
3050 \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
3051 \tl_clear:N \l_tmpa_tl
3052 \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
3053 \int_incr:N \l_tmpa_int
3054 \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
3055 \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables
3056 \str_if_eq:VnTF \l_tmpb_str a {
3057 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3058 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3059 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3060 } }
3061 }{
3062 \str_if_eq:VnTF \l_tmpb_str B {
3063 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3064 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3065 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3066 } }
3067 }{
3068 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3069 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
3070 } }
3071 }
3072 }
3073 }
3074 \stex_annotate_invisible:nnn { notationcomp }{ }{
3075 \str_set:Nx \l_stex_current_symbol_str {var://\l__stex_variables_name_str }
3076 $ \exp_args:Nno \use:nn { \use:c {
3077 stex_var_notation_\l__stex_variables_name_str_cs
3078 } } { \l_tmpa_tl } $
3079 }
3080 }
3081 }\ignorespacesandpars
3082 }
3083
3084 \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
3085 }
3086
3087 \cs_new:Nn \_stex_reset:N {
3088 \tl_if_exist:NTF #1 {
3089 \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }

```



```

3090 }{
3091   \let \exp_not:N #1 \exp_not:N \undefined
3092 }
3093 }
3094
3095 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
3096   \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
3097   \exp_args:Nnx \use:nn {
3098     % TODO
3099     \stex_annotate_invisible:nnn {vardecl}{\clist_use:Nn\l__stex_variables_names,}{
3100       #2
3101     }
3102   }{
3103     \_stex_reset:N \varnot
3104     \_stex_reset:N \vartype
3105     \_stex_reset:N \vardefi
3106   }
3107 }
3108
3109 \NewDocumentCommand \vardef { s } {
3110   \IfBooleanTF#1 {
3111     \__stex_variables_do_complex:nn
3112   }{
3113     \__stex_variables_do_simple:nnn
3114   }
3115 }
3116
3117 \NewDocumentCommand \svar { 0{} m }{
3118   \tl_if_empty:nTF {#1}{
3119     \str_set:Nn \l_tmpa_str { #2 }
3120   }{
3121     \str_set:Nn \l_tmpa_str { #1 }
3122   }
3123   \_stex_term_omv:nn {
3124     var://\l_tmpa_str
3125   }{
3126     \exp_args:Nnx \use:nn {
3127       \def\comp{\_varcomp}
3128       \str_set:Nx \l_stex_current_symbol_str { var://\l_tmpa_str }
3129       \comp{ #2 }
3130     }{
3131       \_stex_reset:N \comp
3132       \_stex_reset:N \l_stex_current_symbol_str
3133     }
3134   }
3135 }
3136
3137
3138
3139 \keys_define:nn { stex / varseq } {
3140   name .str_set_x:N = \l__stex_variables_name_str ,
3141   args .int_set:N   = \l__stex_variables_args_int ,
3142   type .tl_set:N    = \l__stex_variables_type_tl ,
3143   mid .tl_set:N     = \l__stex_variables_mid_tl ,

```

```

3144   bind      .choices:nn      =
3145           {forall,exists}
3146           {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
3147   }
3148
3149   \cs_new_protected:Nn \__stex_variables_seq_args:n {
3150     \str_clear:N \l__stex_variables_name_str
3151     \int_set:Nn \l__stex_variables_args_int 1
3152     \tl_clear:N \l__stex_variables_type_tl
3153     \str_clear:N \l__stex_variables_bind_str
3154
3155     \keys_set:nn { stex / varseq } { #1 }
3156   }
3157
3158   \NewDocumentCommand \varseq {m O{}} m m m m){
3159     \__stex_variables_seq_args:n { #2 }
3160     \str_if_empty:NT \l__stex_variables_name_str {
3161       \str_set:Nx \l__stex_variables_name_str { #1 }
3162     }
3163     \prop_clear:N \l_tmpa_prop
3164     \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
3165
3166     \seq_set_from_clist:Nn \l_tmpa_seq {#3}
3167     \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
3168       \msg_error:nnxx{stex}{error/seqlength}
3169       {\int_use:N \l__stex_variables_args_int}
3170       {\seq_count:N \l_tmpa_seq}
3171     }
3172     \seq_set_from_clist:Nn \l_tmpb_seq {#4}
3173     \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
3174       \msg_error:nnxx{stex}{error/seqlength}
3175       {\int_use:N \l__stex_variables_args_int}
3176       {\seq_count:N \l_tmpb_seq}
3177     }
3178     \prop_put:Nnn \l_tmpa_prop {starts} {#3}
3179     \prop_put:Nnn \l_tmpa_prop {ends} {#4}
3180
3181     \cs_generate_from_arg_count:cNnn {stex_varseq\l__stex_variables_name_str _cs}
3182     \cs_set:Npn { \int_use:N \l__stex_variables_args_int } { #5 }
3183
3184     \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}
3185     \int_step_inline:nn \l__stex_variables_args_int {
3186       \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
3187     }
3188     \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
3189     \tl_put_right:Nn \l_tmpa_tl {\,\ellipses,}
3190     \tl_if_empty:NF \l__stex_variables_mid_tl {
3191       \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
3192       \tl_put_right:Nn \l_tmpa_tl {\,\ellipses,}
3193     }
3194     \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}
3195     \int_step_inline:nn \l__stex_variables_args_int {
3196       \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
3197     }

```

```

3198 \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
3199 \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
3200
3201
3202 \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl
3203
3204 \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}
3205
3206 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}
3207
3208 \int_step_inline:nn \l__stex_variables_args_int {
3209   \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
3210     \stex_term_math_arg:nnn{i##1}{0}{\exp_not:n{####}##1}
3211   }}
3212 }
3213
3214 \tl_set:Nx \l_tmpa_tl {
3215   \stex_term_math_oma:nnnn { varseq://\l__stex_variables_name_str}{0}{
3216     \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
3217   }
3218 }
3219
3220 \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \stex_symbol_after_invokation_tl} }
3221
3222 \exp_args:Nno \use:nn {
3223   \cs_generate_from_arg_count:cNnn {stex_varseq\l__stex_variables_name_str _cs}
3224   \cs_set:Npn {\int_use:N \l__stex_variables_args_int}{\l_tmpa_tl}
3225
3226   \stex_debug:nn{sequences}{New~Sequence:~
3227     \expandafter\meaning\csname stex_varseq\l__stex_variables_name_str _cs\endcsname\\~\\
3228     \prop_to_keyval:N \l_tmpa_prop
3229   }
3230   \stex_if_do_html:T{\stex_annotate_invisible:nnn{varseq}{\l__stex_variables_name_str}{
3231     \tl_if_empty:NF \l__stex_variables_type_tl {
3232       \stex_annotate:nnn {type}{\}{\seqtype\l__stex_variables_type_tl$}
3233     }
3234     \stex_annotate:nnn {args}{\int_use:N \l__stex_variables_args_int}{\}
3235     \str_if_empty:NF \l__stex_variables_bind_str {
3236       \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{\}
3237     }
3238   }}
3239
3240   \prop_set_eq:cN {stex_varseq\l__stex_variables_name_str _prop}\l_tmpa_prop
3241   \ignorespacesandpars
3242 }
3243
3244 </package>

```

## Chapter 30

# STEX -Terms Implementation

```
3245 <*package>
3246
3247 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
3248
3249 <@@=stex_terms>
3250
3251 Warnings and error messages
3252 \msg_new:nnn{stex}{error/nonotation}{
3253   Symbol~#1~invoked,~but~has~no~notation#2!
3254 }
3255 \msg_new:nnn{stex}{error/notationarg}{
3256   Error~in~parsing~notation~#1
3257 }
3258 \msg_new:nnn{stex}{error/noop}{
3259   Symbol~#1~has~no~operator~notation~for~notation~#2
3260 }
3261 \msg_new:nnn{stex}{error/notallowed}{
3262   Symbol~invocation~#1~not~allowed~in~notation~component~of~#2
3263 }
3264 \msg_new:nnn{stex}{error/doubleargument}{
3265   Argument~#1~of~symbol~#2~already~assigned
3266 }
3267 \msg_new:nnn{stex}{error/overarity}{
3268   Argument~#1~invalid~for~symbol~#2~with~arity~#3
3269 }
3270
```

### 30.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
3269
3270
3271 \bool_new:N \l_stex_allow_semantic_bool
3272 \bool_set_true:N \l_stex_allow_semantic_bool
3273
```

```

3274 \cs_new_protected:Nn \stex_invoke_symbol:n {
3275   \bool_if:NTF \l_stex_allow_semantic_bool {
3276     \str_if_eq:eeF {
3277       \prop_item:cn {
3278         l_stex_symdecl_#1_prop
3279       }{ deprecate }
3280     }{}{
3281       \msg_warning:nxxx{stex}{warning/deprecated}{
3282         Symbol~#1
3283       }{
3284         \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
3285       }
3286     }
3287     \if_mode_math:
3288       \exp_after:wN \__stex_terms_invoke_math:n
3289     \else:
3290       \exp_after:wN \__stex_terms_invoke_text:n
3291     \fi: { #1 }
3292   }{
3293     \msg_error:nxxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
3294   }
3295 }
3296
3297 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
3298   \peek_charcode_remove:NTF ! {
3299     \__stex_terms_invoke_op_custom:nn {#1}
3300   }{
3301     \__stex_terms_invoke_custom:nn {#1}
3302   }
3303 }
3304
3305 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
3306   \peek_charcode_remove:NTF ! {
3307     % operator
3308     \peek_charcode_remove:NTF * {
3309       % custom op
3310       \__stex_terms_invoke_op_custom:nn {#1}
3311     }{
3312       % op notation
3313       \peek_charcode:NTF [ {
3314         \__stex_terms_invoke_op_notation:nw {#1}
3315       }{
3316         \__stex_terms_invoke_op_notation:nw {#1}[]
3317       }
3318     }
3319   }{
3320     \peek_charcode_remove:NTF * {
3321       \__stex_terms_invoke_custom:nn {#1}
3322       % custom
3323     }{
3324       % normal
3325       \peek_charcode:NTF [ {
3326         \__stex_terms_invoke_notation:nw {#1}
3327       }{

```

```

3328     \_stex_terms_invoke_notation:nw {#1}[]
3329   }
3330 }
3331 }
3332 }
3333
3334
3335 \cs_new_protected:Nn \_stex_terms_invoke_op_custom:nn {
3336   \exp_args:Nnx \use:nn {
3337     \def\comp{\_comp}
3338     \str_set:Nn \l_stex_current_symbol_str { #1 }
3339     \bool_set_false:N \l_stex_allow_semantic_bool
3340     \_stex_term_oms:nnn {#1}{#1 \c_hash_str CUSTOM-}{
3341       \comp{ #2 }
3342     }
3343   }{
3344     \_stex_reset:N \comp
3345     \_stex_reset:N \l_stex_current_symbol_str
3346     \bool_set_true:N \l_stex_allow_semantic_bool
3347   }
3348 }
3349
3350 \keys_define:nn { stex / terms } {
3351   % lang .tl_set_x:N = \l_stex_notation_lang_str ,
3352   variant .tl_set_x:N = \l_stex_notation_variant_str ,
3353   unknown .code:n = \str_set:Nx
3354     \l_stex_notation_variant_str \l_keys_key_str
3355 }
3356
3357 \cs_new_protected:Nn \_stex_terms_args:n {
3358   % \str_clear:N \l_stex_notation_lang_str
3359   \str_clear:N \l_stex_notation_variant_str
3360
3361   \keys_set:nn { stex / terms } { #1 }
3362 }
3363
3364 \cs_new_protected:Nn \stex_find_notation:nn {
3365   \_stex_terms_args:n { #2 }
3366   \seq_if_empty:cTF {
3367     l_stex_symdecl_ #1 _notations
3368   } {
3369     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
3370   } {
3371     \str_if_empty:NTF \l_stex_notation_variant_str {
3372       \seq_get_left:cN {l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
3373     }{
3374       \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
3375         \l_stex_notation_variant_str
3376       }{
3377         % \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3378       }{
3379         \msg_error:nnxx{stex}{error/nonotation}{#1}{
3380           ~\l_stex_notation_variant_str
3381         }
3382       }

```

```

3382     }
3383   }
3384 }
3385 }
3386
3387 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3388   \exp_args:Nnx \use:nn {
3389     \def\comp{\_comp}
3390     \str_set:Nn \l_stex_current_symbol_str { #1 }
3391     \stex_find_notation:nn { #1 }{ #2 }
3392     \bool_set_false:N \l_stex_allow_semantic_bool
3393     \cs_if_exist:cTF {
3394       stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3395     }{
3396       \_stex_term_oms:nnn { #1 }{
3397         #1 \c_hash_str \l_stex_notation_variant_str
3398       }{
3399         \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3400       }
3401     }{
3402       \int_compare:nNnTF {\prop_item:cn {\l_stex_symdecl_#1_prop}{arity}} = 0{
3403         \cs_if_exist:cTF {
3404           stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3405         }{
3406           \tl_set:Nx \stex_symbol_after_invokation_tl {
3407             \_stex_reset:N \comp
3408             \_stex_reset:N \stex_symbol_after_invokation_tl
3409             \_stex_reset:N \l_stex_current_symbol_str
3410             \bool_set_true:N \l_stex_allow_semantic_bool
3411           }
3412           \def\comp{\_comp}
3413           \str_set:Nn \l_stex_current_symbol_str { #1 }
3414           \bool_set_false:N \l_stex_allow_semantic_bool
3415           \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3416         }{
3417           \msg_error:nnxx{stex}{error/nonotation}{#1}{
3418             ~\l_stex_notation_variant_str
3419           }
3420         }
3421       }{
3422         \msg_error:nnxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3423       }
3424     }
3425   }{
3426     \_stex_reset:N \comp
3427     \_stex_reset:N \l_stex_current_symbol_str
3428     \bool_set_true:N \l_stex_allow_semantic_bool
3429   }
3430 }
3431
3432 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {
3433   \stex_find_notation:nn { #1 }{ #2 }
3434   \cs_if_exist:cTF {
3435     stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs

```

```

3436 }{
3437   \tl_set:Nx \stex_symbol_after_invokation_tl {
3438     \_stex_reset:N \comp
3439     \_stex_reset:N \stex_symbol_after_invokation_tl
3440     \_stex_reset:N \l_stex_current_symbol_str
3441     \bool_set_true:N \l_stex_allow_semantic_bool
3442   }
3443   \def\comp{\_comp}
3444   \str_set:Nn \l_stex_current_symbol_str { #1 }
3445   \bool_set_false:N \l_stex_allow_semantic_bool
3446   \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3447 }{
3448   \msg_error:nnxx{stex}{error/nonotation}{#1}{
3449     ~\l_stex_notation_variant_str
3450   }
3451 }
3452 }
3453
3454 \prop_new:N \l__stex_terms_custom_args_prop
3455
3456 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3457   \exp_args:Nnx \use:nn {
3458     \bool_set_false:N \l_stex_allow_semantic_bool
3459     \def\comp{\_comp}
3460     \str_set:Nn \l_stex_current_symbol_str { #1 }
3461     \prop_clear:N \l__stex_terms_custom_args_prop
3462     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3463     \prop_get:cnN {
3464       l_stex_symdecl_#1 _prop
3465     }{ args } \l_tmpa_str
3466     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3467     \tl_set:Nn \arg { \__stex_terms_arg: }
3468     \str_if_empty:NTF \l_tmpa_str {
3469       \_stex_term oms:nnn {#1}{#1\c_hash_str CUSTOM-}{#2}
3470     }{
3471       \str_if_in:NnTF \l_tmpa_str b {
3472         \_stex_term ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3473       }{
3474         \str_if_in:NnTF \l_tmpa_str B {
3475           \_stex_term ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3476         }{
3477           \_stex_term oma:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3478         }
3479       }
3480     }
3481     % TODO check that all arguments exist
3482   }{
3483     \_stex_reset:N \l_stex_current_symbol_str
3484     \_stex_reset:N \arg
3485     \_stex_reset:N \comp
3486     \_stex_reset:N \l__stex_terms_custom_args_prop
3487     \bool_set_true:N \l_stex_allow_semantic_bool
3488   }
3489 }

```



```

3490 \NewDocumentCommand \__stex_terms_arg: { s O{} m}{
3491   \tl_if_empty:nTF {#2}{
3492     \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
3493     \bool_set_true:N \l_tmpa_bool
3494     \bool_do_while:Nn \l_tmpa_bool {
3495       \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int
3496         \int_incr:N \l_tmpa_int
3497       }{
3498         \bool_set_false:N \l_tmpa_bool
3499       }
3500     }
3501   }{
3502     \int_set:Nn \l_tmpa_int { #2 }
3503   }
3504   \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
3505   \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3506     \msg_error:nnxxx{stex}{error/overarity}
3507     {\int_use:N \l_tmpa_int}
3508     {\l_stex_current_symbol_str}
3509     {\str_count:N \l_tmpa_str}
3510   }
3511   \str_set:Nx \l_tmpa_str {\str_item:Nn \l_tmpa_str \l_tmpa_int}
3512   \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
3513     \bool_lazy_any:nF {
3514       {\str_if_eq_p:Vn \l_tmpa_str {a}}
3515       {\str_if_eq_p:Vn \l_tmpa_str {B}}
3516     }{
3517       \msg_error:nnxx{stex}{error/doubleargument}
3518       {\int_use:N \l_tmpa_int}
3519       {\l_stex_current_symbol_str}
3520     }
3521   }
3522 }
3523 \exp_args:NNx \prop_put:Nnn \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {#3}
3524 \bool_set_true:N \l_stex_allow_semantic_bool
3525 \IfBooleanTF#1{
3526   \stex_annotate_invisible:n { %TODO
3527     \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{#3}
3528   }
3529 }{ %TODO
3530   \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{#3}
3531 }
3532 \bool_set_false:N \l_stex_allow_semantic_bool
3533 }
3534
3535
3536 \cs_new_protected:Nn \_stex_term_arg:nn {
3537   \bool_set_true:N \l_stex_allow_semantic_bool
3538   \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3539   \bool_set_false:N \l_stex_allow_semantic_bool
3540 }
3541
3542 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
3543   \exp_args:Nnx \use:nn

```

```

3544 { \int_set:Nn \l__stex_terms_downprec { #2 }
3545   \stex_term_arg:nn { #1 }{ #3 }
3546 }
3547 { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3548 }

```

(End definition for `\stex_invoke_symbol:n`. This function is documented on page 79.)

`\stex_term_math_assoc_arg:nnnn`

```

3549 \cs_new_protected:Nn \stex_term_math_assoc_arg:nnnn {
3550   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3551   \tl_set:Nn \l_tmpb_tl {\stex_term_math_arg:nnn{#1}{#2}}
3552   \tl_if_empty:NTF { #3 }{
3553     \stex_term_math_arg:nnn{#1}{#2}{#3}
3554   }{
3555     \exp_args:Nx \tl_if_empty:NTF { \tl_tail:n{ #3 } }{
3556       \expandafter\if\expandafter\relax\noexpand#3
3557         \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_maybe_sequence:Nn#3{#1}}
3558       \else
3559         \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_simple:nn{#1}{#3}}
3560       \fi
3561       \l_tmpa_tl
3562     }{
3563       \__stex_terms_math_assoc_arg_simple:nn{#1}{#3}
3564     }
3565   }
3566 }
3567
3568 \cs_new_protected:Nn \__stex_terms_math_assoc_arg_maybe_sequence:Nn {
3569   \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
3570   \str_if_empty:NTF \l_tmpa_str {
3571     \exp_args:Nx \cs_if_eq:NNTF {
3572       \tl_head:N #1
3573     } \stex_invoke_sequence:n {
3574       \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
3575       \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
3576       \tl_set:Nx \l_tmpa_tl {\prop_item:cn {stex_varseq\l_tmpa_str _prop}{notation}}
3577       \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
3578       \tl_set:Nx \l_tmpa_tl {\exp_not:N \exp_not:n{
3579         \exp_not:n{\exp_args:Nnx \use:nn} {
3580           \exp_not:n {
3581             \def\comp{\_varcomp}
3582             \str_set:Nn \l_stex_current_symbol_str
3583             } {varseq://\l_tmpa_str}
3584           \exp_not:n{ ##1 }
3585         }{
3586           \exp_not:n {
3587             \stex_reset:N \comp
3588             \stex_reset:N \l_stex_current_symbol_str
3589           }
3590         }
3591       }}}
3592   \exp_args:Nno \use:nn {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
3593   \seq_reverse:N \l_tmpa_seq

```

```

3594 \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
3595 \seq_map_inline:Nn \l_tmpa_seq {
3596   \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3597     \exp_args:Nno
3598     \l_tmpa_cs { ##1 } \l_tmpa_tl
3599   }
3600 }
3601 \tl_set:Nx \l_tmpa_tl {
3602   \stex_term_omv:nn {varseq://\l_tmpa_str}{
3603     \exp_args:No \exp_not:n \l_tmpa_tl
3604   }
3605 }
3606 \exp_args:No\l_tmpb_tl\l_tmpa_tl
3607 }{
3608   \stex_terms_math_assoc_arg_simple:nn{#2} { #1 }
3609 }
3610 } {
3611   \stex_terms_math_assoc_arg_simple:nn{#2} { #1 }
3612 }
3613 }
3614 }
3615
3616 \cs_new_protected:Nn \stex_terms_math_assoc_arg_simple:nn {
3617   \clist_set:Nn \l_tmpa_clist{ #2 }
3618   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3619     \tl_set:Nn \l_tmpa_tl { \stex_term_arg:nn{A#1}{ #2 } }
3620   }{
3621     \clist_reverse:N \l_tmpa_clist
3622     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3623     \tl_set:Nx \l_tmpa_tl { \stex_term_arg:nn{A#1}{
3624       \exp_args:No \exp_not:n \l_tmpa_tl
3625     }}
3626     \clist_map_inline:Nn \l_tmpa_clist {
3627       \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3628         \exp_args:Nno
3629         \l_tmpa_cs { \stex_term_arg:nn{A#1}{##1} } \l_tmpa_tl
3630       }
3631     }
3632   }
3633   \exp_args:No\l_tmpb_tl\l_tmpa_tl
3634 }

```

(End definition for `\stex_term_math_assoc_arg:nnnn`. This function is documented on page 79.)

## 30.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
3635 \tl_const:Nx \infprec {\int_use:N \c_max_int}
3636 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
3637 \int_new:N \l__stex_terms_downprec
3638 \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 80.)

Bracketing:

```
\l__stex_terms_left_bracket_str
\l__stex_terms_right_bracket_str
```

```
3639 \tl_set:Nn \l__stex_terms_left_bracket_str (
3640 \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

```
3641 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
3642   \bool_if:NTF \l__stex_terms_brackets_done_bool {
3643     \bool_set_false:N \l__stex_terms_brackets_done_bool
3644     #2
3645   } {
3646     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
3647       \bool_if:NTF \l__stex_inarray_bool { #2 } {
3648         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
3649         \dobrackets { #2 }
3650       }
3651     }{ #2 }
3652   }
3653 }
```

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

**\dobrackets**

```
3654 \bool_new:N \l__stex_terms_brackets_done_bool
3655 %\RequirePackage{scalerel}
3656 \cs_new_protected:Npn \dobrackets #1 {
3657   %\ThisStyle{\if D\m@switch
3658   %   \exp_args:Nnx \use:nn
3659   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
3660   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
3661   %   \else
3662   \exp_args:Nnx \use:nn
3663   {
3664     \bool_set_true:N \l__stex_terms_brackets_done_bool
3665     \int_set:Nn \l__stex_terms_downprec \infprec
3666     \l__stex_terms_left_bracket_str
3667     #1
3668   }
3669   {
3670     \bool_set_false:N \l__stex_terms_brackets_done_bool
3671     \l__stex_terms_right_bracket_str
3672     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3673   }
3674   %\fi}
3675 }
```

(End definition for `\dobrackets`. This function is documented on page 80.)

**\withbrackets**

```
3676 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
3677   \exp_args:Nnx \use:nn
3678   {
3679     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
3680     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
3681     #3
3682   }
3683   {
3684     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
3685     {\l__stex_terms_left_bracket_str}
3686     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
3687     {\l__stex_terms_right_bracket_str}
3688   }
3689 }
```

(End definition for \withbrackets. This function is documented on page 80.)

**\STEXinvisible**

```
3690 \cs_new_protected:Npn \STEXinvisible #1 {
3691   \stex_annotate_invisible:n { #1 }
3692 }
```

(End definition for \STEXinvisible. This function is documented on page 80.)

OMDoc terms:

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

```
3693 \cs_new_protected:Nn \_stex_term_oms:nnn {
3694   \stex_annotate:nnn{ OMID }{ #2 }{
3695     #3
3696   }
3697 }
3698
3699 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
3700   \__stex_terms_maybe_brackets:nn { #3 }{
3701     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3702   }
3703 }
```

(End definition for \\_stex\_term\_math\_oms:nnnn. This function is documented on page 79.)

**\\_stex\_term\_math\_omv:nn**

```
3704 \cs_new_protected:Nn \_stex_term_omv:nn {
3705   \stex_annotate:nnn{ OMV }{ #1 }{
3706     #2
3707   }
3708 }
```

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

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

```
3709 \cs_new_protected:Nn \_stex_term_oma:nnn {
3710   \stex_annotate:nnn{ OMA }{ #2 }{
3711     #3
3712   }
```

```

3713 }
3714
3715 \cs_new_protected:Nn \stex_term_math_oma:nnnn {
3716   \__stex_terms_maybe_brackets:nn { #3 }{
3717     \stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3718   }
3719 }

```

(End definition for `\stex_term_math_oma:nnnn`. This function is documented on page 79.)

`\stex_term_math_omb:nnnn`

```

3720 \cs_new_protected:Nn \stex_term_ombind:nnn {
3721   \stex_annotate:nnn{ OMBIND }{ #2 }{
3722     #3
3723   }
3724 }
3725
3726 \cs_new_protected:Nn \stex_term_math_omb:nnnn {
3727   \__stex_terms_maybe_brackets:nn { #3 }{
3728     \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3729   }
3730 }

```

(End definition for `\stex_term_math_omb:nnnn`. This function is documented on page 79.)

`\symref`

`\symname`

```

3731 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3732
3733 \keys_define:nn { stex / symname } {
3734   pre      .tl_set_x:N      = \l__stex_terms_pre_tl ,
3735   post     .tl_set_x:N      = \l__stex_terms_post_tl ,
3736   root     .tl_set_x:N      = \l__stex_terms_root_tl
3737 }
3738
3739 \cs_new_protected:Nn \stex_symname_args:n {
3740   \tl_clear:N \l__stex_terms_post_tl
3741   \tl_clear:N \l__stex_terms_pre_tl
3742   \tl_clear:N \l__stex_terms_root_str
3743   \keys_set:nn { stex / symname } { #1 }
3744 }
3745
3746 \NewDocumentCommand \symref { m m }{
3747   \let\compemph_uri_prev:\compemph@uri
3748   \let\compemph@uri\symrefemph@uri
3749   \STEXsymbol{#1}!\{ #2 }
3750   \let\compemph@uri\compemph_uri_prev:
3751 }
3752
3753 \NewDocumentCommand \synonym { O{} m m }{
3754   \stex_symname_args:n { #1 }
3755   \let\compemph_uri_prev:\compemph@uri
3756   \let\compemph@uri\symrefemph@uri
3757   % TODO
3758   \STEXsymbol{#2}!\{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
3759   \let\compemph@uri\compemph_uri_prev:

```

```

3760 }
3761
3762 \NewDocumentCommand \symname { 0{} m }{
3763   \stex_symname_args:n { #1 }
3764   \stex_get_symbol:n { #2 }
3765   \str_set:Nx \l_tmpa_str {
3766     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3767   }
3768   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3769
3770   \let\compemph_uri_prev:\compemph@uri
3771   \let\compemph@uri\symrefemph@uri
3772   \exp_args:NNx \use:nn
3773   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmmode*\fi{
3774     \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
3775   } }
3776   \let\compemph@uri\compemph_uri_prev:
3777 }
3778
3779 \NewDocumentCommand \Symname { 0{} m }{
3780   \stex_symname_args:n { #1 }
3781   \stex_get_symbol:n { #2 }
3782   \str_set:Nx \l_tmpa_str {
3783     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3784   }
3785   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3786   \let\compemph_uri_prev:\compemph@uri
3787   \let\compemph@uri\symrefemph@uri
3788   \exp_args:NNx \use:nn
3789   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmmode*\fi{
3790     \exp_after:wN \stex_capitalize:n \l_tmpa_str
3791     \l__stex_terms_post_tl
3792   } }
3793   \let\compemph@uri\compemph_uri_prev:
3794 }

```

(End definition for `\symref` and `\symname`. These functions are documented on page 79.)

### 30.3 Notation Components

```

3795 <@@=stex_notationcomps>

\comp
\compemph@uri
\compemph
\defemph
\defemph@uri
\symrefemph
\symrefemph@uri
\varemp
\varemp@uri

3796 \cs_new_protected:Npn \_comp #1 {
3797   \str_if_empty:NF \l_stex_current_symbol_str {
3798     \stex_html_backend:TF {
3799       \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
3800     }{
3801       \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3802     }
3803   }
3804 }
3805
3806 \cs_new_protected:Npn \_varcomp #1 {

```

```

3807 \str_if_empty:NF \l_stex_current_symbol_str {
3808   \stex_html_backend:TF {
3809     \stex_annotate:nnn { varcomp }{ \l_stex_current_symbol_str }{ #1 }
3810   }{
3811     \exp_args:Nnx \varemp@uri { #1 } { \l_stex_current_symbol_str }
3812   }
3813 }
3814 }
3815
3816 \def\comp{\_comp}
3817
3818 \cs_new_protected:Npn \compemph@uri #1 #2 {
3819   \compemph{ #1 }
3820 }
3821
3822
3823 \cs_new_protected:Npn \compemph #1 {
3824   #1
3825 }
3826
3827 \cs_new_protected:Npn \defemph@uri #1 #2 {
3828   \defemph{#1}
3829 }
3830
3831 \cs_new_protected:Npn \defemph #1 {
3832   \textbf{#1}
3833 }
3834
3835 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3836   \symrefemph{#1}
3837 }
3838
3839 \cs_new_protected:Npn \symrefemph #1 {
3840   \emph{#1}
3841 }
3842
3843 \cs_new_protected:Npn \varemp@uri #1 #2 {
3844   \varemp{#1}
3845 }
3846
3847 \cs_new_protected:Npn \varemp #1 {
3848   #1
3849 }

```

(End definition for `\comp` and others. These functions are documented on page 80.)

## **\ellipses**

```

3850 \NewDocumentCommand \ellipses {} { \ldots }

```

(End definition for `\ellipses`. This function is documented on page 80.)

```

\parray
\prmatrix 3851 \bool_new:N \l_stex_inparray_bool
\parrayline 3852 \bool_set_false:N \l_stex_inparray_bool
\parraylineh 3853 \NewDocumentCommand \parray { m m } {
\parraycell

```



```

3854 \begingroup
3855 \bool_set_true:N \l_stex_inarray_bool
3856 \begin{array}{#1}
3857 #2
3858 \end{array}
3859 \endgroup
3860 }
3861
3862 \NewDocumentCommand \prmatrix { m } {
3863 \begingroup
3864 \bool_set_true:N \l_stex_inarray_bool
3865 \begin{matrix}
3866 #1
3867 \end{matrix}
3868 \endgroup
3869 }
3870
3871 \def \maybepline {
3872 \bool_if:NT \l_stex_inarray_bool {\hline}
3873 }
3874
3875 \def \parrayline #1 #2 {
3876 #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3877 }
3878
3879 \def \pmrow #1 { \parrayline{}{ #1 } }
3880
3881 \def \parraylineh #1 #2 {
3882 #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3883 }
3884
3885 \def \parraycell #1 {
3886 #1 \bool_if:NT \l_stex_inarray_bool {&}
3887 }

```

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

## 30.4 Variables

```

3888 <@@=stex_variables>

```

`\stex_invoke_variable:n` Invokes a variable

```

3889 \cs_new_protected:Nn \stex_invoke_variable:n {
3890 \if_mode_math:
3891 \exp_after:wN \__stex_variables_invoke_math:n
3892 \else:
3893 \exp_after:wN \__stex_variables_invoke_text:n
3894 \fi: {#1}
3895 }
3896
3897 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
3898 %TODO
3899 }
3900

```

```

3901
3902 \cs_new_protected:Nn \__stex_variables_invoke_math:n {
3903   \peek_charcode_remove:NTF ! {
3904     \peek_charcode_remove:NTF ! {
3905       \peek_charcode:NTF [ {
3906         \__stex_variables_invoke_op_custom:nw
3907       }{
3908         % TODO throw error
3909       }
3910     }{
3911       \__stex_variables_invoke_op:n { #1 }
3912     }
3913   }{
3914     \peek_charcode_remove:NTF * {
3915       \__stex_variables_invoke_text:n { #1 }
3916     }{
3917       \__stex_variables_invoke_math_ii:n { #1 }
3918     }
3919   }
3920 }
3921
3922 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
3923   \cs_if_exist:cTF {
3924     stex_var_op_notation_ #1 _cs
3925   }{
3926     \exp_args:Nnx \use:nn {
3927       \def\comp{\_varcomp}
3928       \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3929       \_stex_term_omv:nn { var://#1 }{
3930         \use:c{stex_var_op_notation_ #1 _cs }
3931       }
3932     }{
3933       \_stex_reset:N \comp
3934       \_stex_reset:N \l_stex_current_symbol_str
3935     }
3936   }{
3937     \int_compare:nNnTF {\prop_item:cn {l_stex_variable_#1_prop}{arity}} = 0{
3938       \__stex_variables_invoke_math_ii:n {#1}
3939     }{
3940       \msg_error:nnxx{stex}{error/noop}{variable~#1}{}
3941     }
3942   }
3943 }
3944
3945 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n #1 {
3946   \cs_if_exist:cTF {
3947     stex_var_notation_#1_cs
3948   }{
3949     \tl_set:Nx \stex_symbol_after_invokation_tl {
3950       \_stex_reset:N \comp
3951       \_stex_reset:N \stex_symbol_after_invokation_tl
3952       \_stex_reset:N \l_stex_current_symbol_str
3953       \bool_set_true:N \l_stex_allow_semantic_bool
3954     }

```

```

3955 \def\comp{\_varcomp}
3956 \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3957 \bool_set_false:N \l_stex_allow_semantic_bool
3958 \use:c{stex_var_notation_#1_cs}
3959 }{
3960 \msg_error:nnxx{stex}{error/nonotation}{variable-#1}{s}
3961 }
3962 }

```

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

## 30.5 Sequences

```

3963 <@@=stex_sequences>
3964
3965 \cs_new_protected:Nn \stex_invoke_sequence:n {
3966 \peek_charcode_remove:NTF ! {
3967 \_stex_term_omv:nn {varseq://#1}{
3968 \exp_args:Nnx \use:nn {
3969 \def\comp{\_varcomp}
3970 \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3971 \prop_item:cn{stex_varseq_#1_prop}{notation}
3972 }{
3973 \_stex_reset:N \comp
3974 \_stex_reset:N \l_stex_current_symbol_str
3975 }
3976 }
3977 }{
3978 \bool_set_false:N \l_stex_allow_semantic_bool
3979 \def\comp{\_varcomp}
3980 \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3981 \tl_set:Nx \stex_symbol_after_invokation_tl {
3982 \_stex_reset:N \comp
3983 \_stex_reset:N \stex_symbol_after_invokation_tl
3984 \_stex_reset:N \l_stex_current_symbol_str
3985 \bool_set_true:N \l_stex_allow_semantic_bool
3986 }
3987 \use:c { stex_varseq_#1_cs }
3988 }
3989 }
3990 </package>

```

## Chapter 31

# STEX -Structural Features Implementation

```
3991 <*package>
3992
3993 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3994
    Warnings and error messages
3995 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3996   Symbol~#1~can~not~be~assigned~in~copymodule~#2
3997 }
3998 \msg_new:nnn{stex}{error/interpretmodule/noddefinens}{
3999   Symbol~#1~not~assigned~in~interpretmodule~#2
4000 }
4001
4002 \msg_new:nnn{stex}{error/unknownstructure}{
4003   No~structure~#1~found!
4004 }
4005
4006 \msg_new:nnn{stex}{error/unknownfield}{
4007   No~field~#1~in~instance~#2~found!\#3
4008 }
4009
4010 \msg_new:nnn{stex}{error/keyval}{
4011   Invalid~key=value~pair~#1
4012 }
4013 \msg_new:nnn{stex}{error/instantiate/missing}{
4014   Assignments~missing~in~instantiate:~#1
4015 }
4016 \msg_new:nnn{stex}{error/incompatible}{
4017   Incompatible~signature:~#1~(#2)~and~#3~(#4)
4018 }
4019
```

## 31.1 Imports with modification

```

4020 <@@=stex_copymodule>
4021 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
4022   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4023     \tl_set:Nn \l_tmpa_tl { #1 }
4024     \__stex_copymodule_get_symbol_from_cs:
4025   }{
4026     % argument is a string
4027     % is it a command name?
4028     \cs_if_exist:cTF { #1 }{
4029       \cs_set_eq:Nc \l_tmpa_tl { #1 }
4030       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4031       \str_if_empty:NTF \l_tmpa_str {
4032         \exp_args:Nx \cs_if_eq:NNTF {
4033           \tl_head:N \l_tmpa_tl
4034         } \stex_invoke_symbol:n {
4035           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
4036         }{
4037           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4038         }
4039       } {
4040         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4041       }
4042     }{
4043       % argument is not a command name
4044       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4045       % \l_stex_all_symbols_seq
4046     }
4047   }
4048 }
4049
4050 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
4051   \str_set:Nn \l_tmpa_str { #1 }
4052   \bool_set_false:N \l_tmpa_bool
4053   \bool_if:NF \l_tmpa_bool {
4054     \tl_set:Nn \l_tmpa_tl {
4055       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
4056     }
4057     \str_set:Nn \l_tmpa_str { #1 }
4058     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4059     \seq_map_inline:Nn #2 {
4060       \str_set:Nn \l_tmpb_str { ##1 }
4061       \str_if_eq:eeT { \l_tmpa_str } {
4062         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4063       } {
4064         \seq_map_break:n {
4065           \tl_set:Nn \l_tmpa_tl {
4066             \str_set:Nn \l_stex_get_symbol_uri_str {
4067               ##1
4068             }
4069           }
4070         }
4071       }

```

```

4072     }
4073     \l_tmpa_tl
4074   }
4075 }
4076
4077 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
4078   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4079     { \tl_tail:N \l_tmpa_tl }
4080   \tl_if_single:NTF \l_tmpa_tl {
4081     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
4082       \exp_after:wN \str_set:Nn \exp_after:wN
4083         \l_stex_get_symbol_uri_str \l_tmpa_tl
4084       \__stex_copymodule_get_symbol_check:n { #1 }
4085     }{
4086       % TODO
4087       % tail is not a single group
4088     }
4089   }{
4090     % TODO
4091     % tail is not a single group
4092   }
4093 }
4094
4095 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
4096   \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
4097     \msg_error:nxxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
4098       :~\seq_use:Nn #1 {,~}
4099     }
4100   }
4101 }
4102
4103 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
4104   % import module
4105   \stex_import_module_uri:nn { #1 } { #2 }
4106   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
4107   \stex_import_require_module:nnnn
4108     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4109     { \l_stex_import_path_str } { \l_stex_import_name_str }
4110
4111   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
4112   \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
4113
4114   % fields
4115   \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
4116   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4117     \seq_map_inline:cn {c_stex_module_##1_constants}{
4118       \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
4119         ##1 ? #####1
4120       }
4121     }
4122   }
4123
4124   % setup prop
4125   \seq_clear:N \l_tmpa_seq

```

```

4126 \exp_args:Nn \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
4127   name      = \l_stex_current_copymodule_name_str ,
4128   module    = \l_stex_current_module_str ,
4129   from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
4130   includes  = \l_tmpa_seq %,
4131 % fields    = \l_tmpa_seq
4132 }
4133 \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
4134   as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
4135 \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_seq {,
4136 \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
4137
4138 \stex_if_do_html:T {
4139   \begin{stex_annotate_env} {#4} {
4140     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4141   }
4142   \stex_annotate_invisible:nnn{domain}{\l_stex_import_ns_str ?\l_stex_import_name_str}{}
4143 }
4144 }
4145
4146 \cs_new_protected:Nn \stex_copymodule_end:n {
4147   % apply to every field
4148   \def \l_tmpa_cs ##1 ##2 {#1}
4149
4150   \tl_clear:N \__stex_copymodule_module_tl
4151   \tl_clear:N \__stex_copymodule_exec_tl
4152
4153   %\prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
4154   \seq_clear:N \__stex_copymodule_fields_seq
4155
4156   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4157     \seq_map_inline:cn {c_stex_module_##1_constants}{
4158
4159       \tl_clear:N \__stex_copymodule_curr_symbol_tl % <- wrap in current symbol html
4160       \l_tmpa_cs{##1}{####1}
4161
4162       \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
4163         \str_set_eq:Nc \__stex_copymodule_curr_name_str {l__stex_copymodule_copymodule_##1?####1_name_str}
4164         \stex_if_do_html:T {
4165           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4166             \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?####1_name_str}}
4167           }
4168         }
4169       }{
4170         \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str /
4171       }
4172
4173       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1_prop}
4174       \prop_put:Nnx \l_tmpa_prop { name } \__stex_copymodule_curr_name_str
4175       \prop_put:Nnx \l_tmpa_prop { module } \l_stex_current_module_str
4176
4177       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4178         \stex_if_do_html:T {
4179           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {

```

```

4180         $\stex_annotate_invisible:nnn{definiens}{\exp_after:wN \exp_not:N\csname l__st
4181     }
4182 }
4183 \prop_put:Nnn \l_tmpa_prop { defined } { true }
4184 }
4185
4186 \stex_add_constant_to_current_module:n \__stex_copymodule_curr_name_str
4187 \tl_put_right:Nx \__stex_copymodule_module_tl {
4188     \seq_clear:c {l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_n
4189     \prop_set_from_keyval:cn {
4190         l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_name_str _prop
4191     }{
4192         \prop_to_keyval:N \l_tmpa_prop
4193     }
4194 }
4195
4196 \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
4197     \stex_if_do_html:T {
4198         \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4199             \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1
4200         }
4201     }
4202     \tl_put_right:Nx \__stex_copymodule_module_tl {
4203         \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
4204             \stex_invoke_symbol:n {
4205                 \l_stex_current_module_str ? \__stex_copymodule_curr_name_str
4206             }
4207         }
4208     }
4209 }
4210
4211 \seq_put_right:Nx \__stex_copymodule_fields_seq {\l_stex_current_module_str ? \__stex_
4212
4213 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4214     \stex_copy_notations:nn {\l_stex_current_module_str ? \__stex_copymodule_curr_name_s
4215 }
4216
4217 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4218     \stex_if_do_html:TF{
4219         \stex_annotate_invisible:nnn{assignment} {##1?####1} { \exp_after:wN \exp_not:n \e
4220     }{
4221         \exp_after:wN \exp_not:n \exp_after:wN {\__stex_copymodule_curr_symbol_tl}
4222     }
4223 }
4224 }
4225 }
4226
4227
4228 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \__stex_copymodule_fields_seq
4229 \tl_put_left:Nx \__stex_copymodule_module_tl {
4230     \prop_set_from_keyval:cn {
4231         l_stex_copymodule_ \l_stex_current_module_str? \l_stex_current_copymodule_name_str _pro
4232     }{
4233         \prop_to_keyval:N \l_stex_current_copymodule_prop

```



```

4234     }
4235 }
4236
4237 \seq_gput_right:cx{c_stex_module_\l_stex_current_module_str _copymodules}{
4238   \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4239 }
4240
4241 \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4242 \stex_debug:nn{copymodule}{result:\meaning \__stex_copymodule_module_tl}
4243 \stex_debug:nn{copymodule}{output:\meaning \__stex_copymodule_exec_tl}
4244
4245 \__stex_copymodule_exec_tl
4246 \stex_if_do_html:T {
4247   \end{stex_annotate_env}
4248 }
4249 }
4250
4251 \NewDocumentEnvironment {copymodule} { 0{} m m}{
4252   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ copymodule }
4253   \stex_deactivate_macro:Nn \symdecl {module~environments}
4254   \stex_deactivate_macro:Nn \symdef {module~environments}
4255   \stex_deactivate_macro:Nn \notation {module~environments}
4256   \stex_reactivate_macro:N \assign
4257   \stex_reactivate_macro:N \renamedekl
4258   \stex_reactivate_macro:N \donotcopy
4259   \stex_smsmode_do:
4260 }{
4261   \stex_copymodule_end:n {}
4262 }
4263
4264 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
4265   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ interpretmodule }
4266   \stex_deactivate_macro:Nn \symdecl {module~environments}
4267   \stex_deactivate_macro:Nn \symdef {module~environments}
4268   \stex_deactivate_macro:Nn \notation {module~environments}
4269   \stex_reactivate_macro:N \assign
4270   \stex_reactivate_macro:N \renamedekl
4271   \stex_reactivate_macro:N \donotcopy
4272   \stex_smsmode_do:
4273 }{
4274   \stex_copymodule_end:n {
4275     \tl_if_exist:cF {
4276       l__stex_copymodule_copymodule_##1?##2_def_tl
4277     }{
4278       \str_if_eq:eeF {
4279         \prop_item:cn{
4280           l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
4281         }{ true }{
4282           \msg_error:nxxx{stex}{error/interpretmodule/nodéfiniens}{
4283             ##1?##2
4284           }{\l_stex_current_copymodule_name_str}
4285         }
4286       }
4287     }

```

```

4288 }
4289
4290 \iffalse \begin{stex_annotate_env} \fi
4291 \NewDocumentEnvironment {realization} { 0 } { m } {
4292   \stex_copymodule_start:nnnn { #1 } { #2 } { #2 } { realize }
4293   \stex_deactivate_macro:Nn \symdecl {module~environments}
4294   \stex_deactivate_macro:Nn \symdef {module~environments}
4295   \stex_deactivate_macro:Nn \notation {module~environments}
4296   \stex_reactivate_macro:N \donotcopy
4297   \stex_reactivate_macro:N \assign
4298   \stex_smsmode_do:
4299 } {
4300   \stex_import_module_uri:nn { #1 } { #2 }
4301   \tl_clear:N \__stex_copymodule_exec_tl
4302   \tl_set:Nx \__stex_copymodule_module_tl {
4303     \stex_import_require_module:nnnn
4304     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4305     { \l_stex_import_path_str } { \l_stex_import_name_str }
4306   }
4307
4308   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4309     \seq_map_inline:cn {c_stex_module_##1_constants}{
4310       \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str / #1 }
4311       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?###1_def_tl}{
4312         \stex_if_do_html:T {
4313           \tl_put_right:Nx \__stex_copymodule_exec_tl {
4314             \stex_annotate_invisible:nnn{assignment} {##1?###1} {
4315               $\stex_annotate_invisible:nnn{definien}{}{\exp_after:wN \exp_not:N\csname l__stex_copymodule_##1_def_tl\endcsname}
4316             }
4317           }
4318         }
4319         \tl_put_right:Nx \__stex_copymodule_module_tl {
4320           \prop_put:cnn {l_stex_symdecl_##1?###1_prop}{ defined }{ true }
4321         }
4322       }
4323     }
4324   }
4325   \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4326
4327   \__stex_copymodule_exec_tl
4328   \stex_if_do_html:T {\end{stex_annotate_env}}
4329 }
4330
4331 \NewDocumentCommand \donotcopy { m } {
4332   \str_clear:N \l_stex_import_name_str
4333   \str_set:Nn \l_tmpa_str { #1 }
4334   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4335   \seq_map_inline:Nn \l_stex_all_modules_seq {
4336     \str_set:Nn \l_tmpb_str { ##1 }
4337     \str_if_eq:eeT { \l_tmpa_str } {
4338       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4339     } {
4340       \seq_map_break:n {
4341         \stex_if_do_html:T {

```

```

4342         \stex_if_smsmode:F {
4343             \stex_annotate_invisible:nnn{donotcopy}{##1}{
4344                 \stex_annotate:nnn{domain}{##1}{}}
4345         }
4346     }
4347 }
4348 \str_set_eq:NN \l_stex_import_name_str \l_tmpb_str
4349 }
4350 }
4351 \seq_map_inline:cn {c_stex_module_###1_copymodules}{
4352     \str_set:Nn \l_tmpb_str { #####1 }
4353     \str_if_eq:eeT { \l_tmpa_str } {
4354         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4355     } {
4356         \seq_map_break:n {\seq_map_break:n {
4357             \stex_if_do_html:T {
4358                 \stex_if_smsmode:F {
4359                     \stex_annotate_invisible:nnn{donotcopy}{#####1}{
4360                         \stex_annotate:nnn{domain}{
4361                             \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4362                         }{}
4363                     }
4364                 }
4365             }
4366             \str_set:Nx \l_stex_import_name_str {
4367                 \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4368             }
4369         }}
4370     }
4371 }
4372 }
4373 \str_if_empty:NTF \l_stex_import_name_str {
4374     % TODO throw error
4375 }{
4376     \stex_collect_imports:n {\l_stex_import_name_str }
4377     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4378         \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
4379         \seq_map_inline:cn {c_stex_module_###1_constants}{
4380             \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? #####1 }
4381             \bool_lazy_any:nT {
4382                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_name_str}}
4383                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_macroname_str}}
4384                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_def_tl}}
4385             }{
4386                 % TODO throw error
4387             }
4388         }
4389     }
4390     \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
4391     \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_name_str }
4392     \prop_put:Nno \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
4393 }
4394 \stex_smsmode_do:
4395 }

```

```

4396
4397 \NewDocumentCommand \assign { m m }{
4398   \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
4399   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
4400   \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_def_tl}{#2}
4401   \stex_smsmode_do:
4402 }
4403
4404 \keys_define:nn { stex / renamedekl } {
4405   name          .str_set_x:N = \l_stex_renamedekl_name_str
4406 }
4407 \cs_new_protected:Nn \__stex_copymodule_renamedekl_args:n {
4408   \str_clear:N \l_stex_renamedekl_name_str
4409   \keys_set:nn { stex / renamedekl } { #1 }
4410 }
4411
4412 \NewDocumentCommand \renamedekl { O{} m m }{
4413   \__stex_copymodule_renamedekl_args:n { #1 }
4414   \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
4415   \stex_debug:nn{renamedekl}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
4416   \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_macroname_str}{#3}
4417   \str_if_empty:NTF \l_stex_renamedekl_name_str {
4418     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4419       \l_stex_get_symbol_uri_str
4420     } }
4421   } {
4422     \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_name_str}{\l_stex
4423       \stex_debug:nn{renamedekl}{@~\l_stex_current_module_str ? \l_stex_renamedekl_name_str}
4424       \prop_set_eq:cc {l_stex_symdecl_
4425         \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4426         _prop
4427       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str_prop}
4428       \seq_set_eq:cc {l_stex_symdecl_
4429         \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4430         _notations
4431       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str_notations}
4432       \prop_put:cnx {l_stex_symdecl_
4433         \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4434         _prop
4435       }{ name }{ \l_stex_renamedekl_name_str }
4436       \prop_put:cnx {l_stex_symdecl_
4437         \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4438         _prop
4439       }{ module }{ \l_stex_current_module_str }
4440       \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
4441         \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4442       }
4443       \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4444         \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4445       } }
4446   }
4447   \stex_smsmode_do:
4448 }
4449

```

```

4450 \stex_deactivate_macro:Nn \assign {copymodules}
4451 \stex_deactivate_macro:Nn \renamedekl {copymodules}
4452 \stex_deactivate_macro:Nn \donotcopy {copymodules}
4453
4454

```

## 31.2 The feature environment

structural@feature

```

4455 <@@=stex_features>
4456
4457 \NewDocumentEnvironment{structural_feature_module}{ m m m }{
4458   \stex_if_in_module:F {
4459     \msg_set:nnn{stex}{error/nomodule}{
4460       Structural~Feature~has~to~occur~in~a~module:\\
4461       Feature~#2~of~type~#1\\
4462       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
4463     }
4464     \msg_error:nn{stex}{error/nomodule}
4465   }
4466
4467   \str_set_eq:NN \l_stex_feature_parent_str \l_stex_current_module_str
4468
4469   \stex_module_setup:nn{meta=NONE}{#2 - #1}
4470
4471   \stex_if_do_html:T {
4472     \begin{stex_annotate_env}{ feature:#1 }{\l_stex_feature_parent_str ? #2 - #1}
4473     \stex_annotate_invisible:nnn{header}{\{ #3 }
4474   }
4475   }{
4476     \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
4477     \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
4478     \stex_debug:nn{features}{
4479       Feature: \l_stex_last_feature_str
4480     }
4481     \stex_if_do_html:T {
4482       \end{stex_annotate_env}
4483     }
4484   }

```

## 31.3 Structure

structure

```

4485 <@@=stex_structures>
4486 \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
4487   \prop_if_exist:cF {c_stex_module_ \l_stex_current_module_str _structures}{
4488     \prop_new:c {c_stex_module_ \l_stex_current_module_str _structures}
4489   }
4490   \prop_gput:cxn{c_stex_module_ \l_stex_current_module_str _structures}
4491   {#1}{#2}
4492 }
4493

```

```

4494 \keys_define:nn { stex / features / structure } {
4495   name          .str_set_x:N = \l__stex_structures_name_str ,
4496 }
4497
4498 \cs_new_protected:Nn \__stex_structures_structure_args:n {
4499   \str_clear:N \l__stex_structures_name_str
4500   \keys_set:nn { stex / features / structure } { #1 }
4501 }
4502
4503 \NewDocumentEnvironment{mathstructure}{m O{}}{
4504   \__stex_structures_structure_args:n { #2 }
4505   \str_if_empty:NT \l__stex_structures_name_str {
4506     \str_set:Nx \l__stex_structures_name_str { #1 }
4507   }
4508   \stex_suppress_html:n {
4509     \exp_args:Nx \stex_symdecl_do:nn {
4510       name = \l__stex_structures_name_str ,
4511       def = {\STEXsymbol{module-type}}{
4512         \stex_term_math_oms:nnnn {
4513           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4514             { ns } ?
4515           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4516             { name } / \l__stex_structures_name_str - structure
4517         }{}{0}{}
4518       }}
4519     }{ #1 }
4520   }
4521   \exp_args:Nnnx
4522   \begin{structural_feature_module}{ structure }
4523     { \l__stex_structures_name_str }{}
4524   \stex_smsmode_do:
4525 }{
4526   \end{structural_feature_module}
4527   \stex_reset_up_to_module:n \l_stex_last_feature_str
4528   \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
4529   \seq_clear:N \l_tmpa_seq
4530   \seq_map_inline:Nn \l_stex_collect_imports_seq {
4531     \seq_map_inline:cn{c_stex_module_##1_constants}{
4532       \seq_put_right:Nn \l_tmpa_seq { ##1 ? ###1 }
4533     }
4534   }
4535   \exp_args:Nnno
4536   \prop_gput:cnn {c_stex_module_\l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
4537   \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
4538   \stex_add_structure_to_current_module:nn
4539     \l__stex_structures_name_str
4540     \l_stex_last_feature_str
4541
4542   \stex_execute_in_module:x {
4543     \tl_set:cn { #1 }{
4544       \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures
4545     }
4546   }
4547 }

```

```

4548
4549 \cs_new:Nn \stex_invoke_structure:nn {
4550   \stex_invoke_symbol:n { #1?#2 }
4551 }
4552
4553 \cs_new_protected:Nn \stex_get_structure:n {
4554   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4555     \tl_set:Nn \l_tmpa_tl { #1 }
4556     \__stex_structures_get_from_cs:
4557   }{
4558     \cs_if_exist:cTF { #1 }{
4559       \cs_set_eq:Nc \l_tmpa_cs { #1 }
4560       \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
4561       \str_if_empty:NNTF \l_tmpa_str {
4562         \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs } \stex_invoke_structure:nn {
4563           \__stex_structures_get_from_cs:
4564         }{
4565           \__stex_structures_get_from_string:n { #1 }
4566         }
4567       }{
4568         \__stex_structures_get_from_string:n { #1 }
4569       }
4570     }{
4571       \__stex_structures_get_from_string:n { #1 }
4572     }
4573   }
4574 }
4575
4576 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
4577   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4578     { \tl_tail:N \l_tmpa_tl }
4579   \str_set:Nx \l_tmpa_str {
4580     \exp_after:wN \use_i:nn \l_tmpa_tl
4581   }
4582   \str_set:Nx \l_tmpb_str {
4583     \exp_after:wN \use_ii:nn \l_tmpa_tl
4584   }
4585   \str_set:Nx \l_stex_get_structure_str {
4586     \l_tmpa_str ? \l_tmpb_str
4587   }
4588   \str_set:Nx \l_stex_get_structure_module_str {
4589     \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
4590   }
4591 }
4592
4593 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
4594   \tl_set:Nn \l_tmpa_tl {
4595     \msg_error:nnn{stex}{error/unknownstructure}{#1}
4596   }
4597   \str_set:Nn \l_tmpa_str { #1 }
4598   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4599
4600   \seq_map_inline:Nn \l_stex_all_modules_seq {
4601     \prop_if_exist:cT {c_stex_module_##1_structures} {

```

```

4602 \prop_map_inline:cn {c_stex_module_##1_structures} {
4603   \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
4604     \prop_map_break:n{\seq_map_break:n{
4605       \tl_set:Nn \l_tmpa_tl {
4606         \str_set:Nn \l_stex_get_structure_str {##1?####1}
4607         \str_set:Nn \l_stex_get_structure_module_str {####2}
4608       }
4609     }}
4610   }
4611 }
4612 }
4613 }
4614 \l_tmpa_tl
4615 }

```

**\instantiate**

```

4616
4617 \keys_define:nn { stex / instantiate } {
4618   name          .str_set_x:N = \l__stex_structures_name_str
4619 }
4620 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
4621   \str_clear:N \l__stex_structures_name_str
4622   \keys_set:nn { stex / instantiate } { #1 }
4623 }
4624
4625 \NewDocumentCommand \instantiate {m O{} m m O{}}{
4626   \begingroup
4627     \stex_get_structure:n {#3}
4628     \__stex_structures_instantiate_args:n { #2 }
4629     \str_if_empty:NT \l__stex_structures_name_str {
4630       \str_set:Nn \l__stex_structures_name_str { #1 }
4631     }
4632     \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4633     \seq_clear:N \l__stex_structures_fields_seq
4634     \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4635     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4636       \seq_map_inline:cn {c_stex_module_##1_constants}{
4637         \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4638       }
4639     }
4640
4641     \tl_if_empty:nF{#5}{
4642       \seq_set_split:Nnn \l_tmpa_seq , {#5}
4643       \prop_clear:N \l_tmpa_prop
4644       \seq_map_inline:Nn \l_tmpa_seq {
4645         \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4646         \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4647           \msg_error:nnn{stex}{error/keyval}{##1}
4648         }
4649         \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
4650         \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4651         \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
4652         \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}
4653         \exp_args:Nxx \str_if_eq:nnF

```



```

4654         {\prop_item:cn{l_stex_symdecl\_l\_stex_structures_dom_str\_prop}{args}}
4655         {\prop_item:cn{l_stex_symdecl\_l\_stex_get_symbol_uri_str\_prop}{args}}{
4656         \msg_error:nnxxxx{stex}{error/incompatible}
4657         {l\_stex_structures_dom_str}
4658         {\prop_item:cn{l_stex_symdecl\_l\_stex_structures_dom_str\_prop}{args}}
4659         {\l_stex_get_symbol_uri_str}
4660         {\prop_item:cn{l_stex_symdecl\_l\_stex_get_symbol_uri_str\_prop}{args}}
4661     }
4662     \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
4663 }
4664 }
4665
4666 \seq_map_inline:Nn \l\_stex_structures_fields_seq {
4667     \str_set:Nx \l_tmpa_str {field:l\_stex_structures_name_str . \prop_item:cn {l_stex_sy
4668     \stex_debug:nn{instantiate}{Field~\l_tmpa_str :~##1}
4669
4670     \stex_add_constant_to_current_module:n {\l_tmpa_str}
4671     \stex_execute_in_module:x {
4672         \prop_set_from_keyval:cn { l_stex_symdecl\_l\_stex_current_module_str?\l_tmpa_str_p
4673         name = \l_tmpa_str ,
4674         args = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4675         arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4676         assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4677     }
4678     \seq_clear:c {l_stex_symdecl\_l\_stex_current_module_str?\l_tmpa_str\_notations}
4679 }
4680
4681 \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4682     \stex_find_notation:nn{##1}{}
4683     \stex_execute_in_module:x {
4684         \seq_put_right:cn {l_stex_symdecl\_l\_stex_current_module_str?\l_tmpa_str\_notation
4685     }
4686
4687     \stex_copy_control_sequence_ii:ccN
4688     {stex_notation\_l\_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_notation_
4689     {stex_notation_##1\c_hash_str \l_stex_notation_variant_str_cs}
4690     \l_tmpa_tl
4691     \exp_args:No \stex_execute_in_module:n \l_tmpa_tl
4692
4693
4694     \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str_cs}{
4695         \tl_set_eq:Nc \l_tmpa_cs {stex_op_notation_##1\c_hash_str \l_stex_notation_variant
4696         \stex_execute_in_module:x {
4697             \tl_set:cn
4698             {stex_op_notation\_l\_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_not
4699             { \exp_args:No \exp_not:n \l_tmpa_cs}
4700         }
4701     }
4702
4703 }
4704
4705 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\l_stex_cur
4706 }
4707

```

```

4708 \stex_execute_in_module:x {
4709   \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
4710   domain = \l_stex_get_structure_module_str ,
4711   \prop_to_keyval:N \l_tmpa_prop
4712 }
4713 \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
4714 }
4715 \stex_debug:nn{instantiate}{
4716   Instance~\l_stex_current_module_str?\l__stex_structures_name_str \
4717   \prop_to_keyval:N \l_tmpa_prop
4718 }
4719 \exp_args:Nxx \stex_symdecl_do:nn {
4720   type={\STEXsymbol{module-type}}{
4721     \stex_term_math_oms:nnnn {
4722       \l_stex_get_structure_module_str
4723     }{}{0}{}
4724   }}
4725 }{\l__stex_structures_name_str}
4726 % {
4727   \str_set:Nx \l_stex_get_symbol_uri_str {\l_stex_current_module_str?\l__stex_structures
4728   \tl_set:Nn \l_stex_notation_after_do_tl {\__stex_notation_final:}
4729   \stex_notation_do:nnnnn{}{}{}{}{\comp{#4}}
4730 % }
4731 %\exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}
4732 \endgroup
4733 \stex_smsmode_do:\ignorespacesandpars
4734 }
4735
4736 \cs_new_protected:Nn \stex_symbol_or_var:n {
4737   \cs_if_exist:cTF{#1}{
4738     \cs_set_eq:Nc \l_tmpa_tl { #1 }
4739     \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4740     \str_if_empty:NTF \l_tmpa_str {
4741       \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
4742       \stex_invoke_variable:n {
4743         \bool_set_true:N \l_stex_symbol_or_var_bool
4744         \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4745         \str_set:Nx \l_stex_get_symbol_uri_str {
4746           \exp_after:wN \use:n \l_tmpa_tl
4747         }
4748       }{
4749         \bool_set_false:N \l_stex_symbol_or_var_bool
4750         \stex_get_symbol:n{#1}
4751       }
4752     }{
4753       \__stex_structures_symbolorvar_from_string:n{ #1 }
4754     }
4755   }{
4756     \__stex_structures_symbolorvar_from_string:n{ #1 }
4757   }
4758 }
4759
4760 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
4761   \prop_if_exist:cTF {l_stex_variable_#1 _prop}{

```

```

4762     \bool_set_true:N \l_stex_symbol_or_var_bool
4763     \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
4764   }{
4765     \bool_set_false:N \l_stex_symbol_or_var_bool
4766     \stex_get_symbol:n{#1}
4767   }
4768 }
4769
4770 \keys_define:nn { stex / varinstantiate } {
4771   name      .str_set_x:N = \l__stex_structures_name_str,
4772   bind      .choices:nn =
4773     {forall,exists}
4774     {\str_set:Nx \l__stex_structures_bind_str {\l_keys_choice_tl}}
4775 }
4776
4777 \cs_new_protected:Nn \__stex_structures_varinstantiate_args:n {
4778   \str_clear:N \l__stex_structures_name_str
4779   \str_clear:N \l__stex_structures_bind_str
4780   \keys_set:nn { stex / varinstantiate } { #1 }
4781 }
4782
4783 \NewDocumentCommand \varinstantiate {m O{} m m O{}}{
4784   \beginingroup
4785     \stex_get_structure:n {#3}
4786     \__stex_structures_varinstantiate_args:n { #2 }
4787     \str_if_empty:NT \l__stex_structures_name_str {
4788       \str_set:Nn \l__stex_structures_name_str { #1 }
4789     }
4790     \stex_if_do_html:TF{
4791       \stex_annotate:nnn{varinstance}{\l__stex_structures_name_str}
4792     }{\use:n}
4793     {
4794       \stex_if_do_html:T{
4795         \stex_annotate_invisible:nnn{domain}{\l_stex_get_structure_module_str}{}}
4796       }
4797       \seq_clear:N \l__stex_structures_fields_seq
4798       \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4799       \seq_map_inline:Nn \l_stex_collect_imports_seq {
4800         \seq_map_inline:cn {c_stex_module_##1_constants}{
4801           \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4802         }
4803       }
4804       \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4805       \prop_clear:N \l_tmpa_prop
4806       \tl_if_empty:nF {#5} {
4807         \seq_set_split:Nnn \l_tmpa_seq , {#5}
4808         \seq_map_inline:Nn \l_tmpa_seq {
4809           \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4810           \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4811             \msg_error:nnn{stex}{error/keyval}{##1}
4812           }
4813           \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_stru
4814           \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4815           \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol

```

```

4816 \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
4817 \stex_if_do_html:T{
4818   \stex_annotate:nnn{assign}{\l__stex_structures_dom_str,\l_stex_get_symbol_uri_str}
4819 }
4820 \bool_if:NTF \l_stex_symbol_or_var_bool {
4821   \exp_args:Nxx \str_if_eq:nnF
4822     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4823     {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}{
4824     \msg_error:nnxxxx{stex}{error/incompatible}
4825     {\l__stex_structures_dom_str}
4826     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4827     {\l_stex_get_symbol_uri_str}
4828     {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}
4829   }
4830   \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n}
4831 }{
4832   \exp_args:Nxx \str_if_eq:nnF
4833     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4834     {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
4835     \msg_error:nnxxxx{stex}{error/incompatible}
4836     {\l__stex_structures_dom_str}
4837     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4838     {\l_stex_get_symbol_uri_str}
4839     {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
4840   }
4841   \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n}
4842 }
4843 }
4844 }
4845 \tl_gclear:N \g__stex_structures_aftergroup_tl
4846 \seq_map_inline:Nn \l__stex_structures_fields_seq {
4847   \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdecl_\l__stex_structures_fields_seq}{args}}
4848   \stex_debug:nn{varinstantiate}{Field~\l_tmpa_str :~##1}
4849   \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4850     \stex_find_notation:nn{##1}{
4851       \cs_gset_eq:cc{g__stex_structures_tmpa_\l_tmpa_str _cs}
4852         {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4853       \stex_debug:nn{varinstantiate}{Notation:~\cs_meaning:c{g__stex_structures_tmpa_\l_tmpa_str _cs}}
4854       \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
4855         \cs_gset_eq:cc {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
4856           {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4857         \stex_debug:nn{varinstantiate}{Operator~Notation:~\cs_meaning:c{g__stex_structures_tmpa_op_\l_tmpa_str _cs}}
4858       }
4859     }
4860   }
4861   \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4862     \prop_set_from_keyval:cn { l_stex_variable_ \l_tmpa_str _prop}{
4863       name = \l_tmpa_str ,
4864       args = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4865       arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4866       assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4867     }
4868     \cs_set_eq:cc {stex_var_notation_\l_tmpa_str _cs}
4869     {g__stex_structures_tmpa_\l_tmpa_str _cs}

```

```

4870         \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str_cs}
4871         {g__stex_structures_tmpa_op_\l_tmpa_str_cs}
4872     }
4873     \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_inv
4874 }
4875 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4876     \prop_set_from_keyval:cn {l_stex_varinstance_\l__stex_structures_name_str_prop }{
4877         domain = \l_stex_get_structure_module_str ,
4878         \prop_to_keyval:N \l_tmpa_prop
4879     }
4880     \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
4881     \tl_set:cn {l_stex_varinstance_\l__stex_structures_name_str_op_tl}{
4882         \exp_args:Nnx \exp_not:N \use:nn {
4883             \str_set:Nn \exp_not:N \l_stex_current_symbol_str {var://\l__stex_structures_nam
4884             \_stex_term_omv:nn {var://\l__stex_structures_name_str}{
4885                 \exp_not:n{
4886                     \_varcomp{#4}
4887                 }
4888             }
4889             }{
4890                 \exp_not:n{\_stex_reset:N \l_stex_current_symbol_str}
4891             }
4892         }
4893     }
4894 }
4895 \stex_debug:nn{varinstantiate}{\expandafter\detokenize\expandafter{\g__stex_structures_a
4896 \aftergroup\g__stex_structures_aftergroup_tl
4897 \endgroup
4898 \stex_smsmode_do:\ignorespacesandpars
4899 }
4900
4901 \cs_new_protected:Nn \stex_invoke_instance:n {
4902     \peek_charcode_remove:NTF ! {
4903         \stex_invoke_symbol:n{#1}
4904     }{
4905         \_stex_invoke_instance:nn {#1}
4906     }
4907 }
4908
4909
4910 \cs_new_protected:Nn \stex_invoke_varinstance:n {
4911     \peek_charcode_remove:NTF ! {
4912         \exp_args:Nnx \use:nn {
4913             \def\comp{\_varcomp}
4914             \use:c{l_stex_varinstance_#1_op_tl}
4915         }{
4916             \_stex_reset:N \comp
4917         }
4918     }{
4919         \_stex_invoke_varinstance:nn {#1}
4920     }
4921 }
4922
4923 \cs_new_protected:Nn \stex_invoke_instance:nn {

```

```

4924 \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
4925   \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
4926 }{
4927   \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
4928   \msg_error:nnxxx{stex}{error/unknownfield}{#2}{#1}{
4929     \prop_to_keyval:N \l_tmpa_prop
4930   }
4931 }
4932 }
4933
4934 \cs_new_protected:Nn \stex_invoke_varinstance:nn {
4935   \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
4936     \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
4937     \l_tmpa_tl
4938   }{
4939     \msg_error:nnnnn{stex}{error/unknownfield}{#2}{#1}{}
4940   }
4941 }

```

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

`\stex_invoke_structure:nnn`

```

4942 % #1: URI of the instance
4943 % #2: URI of the instantiated module
4944 \cs_new_protected:Nn \stex_invoke_structure:nnn {
4945   \tl_if_empty:nTF{ #3 }{
4946     \prop_set_eq:Nc \l__stex_structures_structure_prop {
4947       c_stex_feature_ #2 _prop
4948     }
4949     \tl_clear:N \l_tmpa_tl
4950     \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
4951     \seq_map_inline:Nn \l_tmpa_seq {
4952       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
4953       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
4954       \cs_if_exist:cT {
4955         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
4956       }{
4957         \tl_if_empty:NF \l_tmpa_tl {
4958           \tl_put_right:Nn \l_tmpa_tl {,}
4959         }
4960         \tl_put_right:Nx \l_tmpa_tl {
4961           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
4962         }
4963       }
4964     }
4965     \exp_args:No \mathstruct \l_tmpa_tl
4966   }{
4967     \stex_invoke_symbol:n{#1/#3}
4968   }
4969 }

```

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

```

4970 \</package>

```

## Chapter 32

# STEX -Statements Implementation

```
4971 <*package>
4972
4973 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4974
4975 <@@=stex_statements>
    Warnings and error messages
4976
\titleemph
4977 \def\titleemph#1{\textbf{#1}}
    (End definition for \titleemph. This function is documented on page ??.)
```

### 32.1 Definitions

#### definiendum

```
4978 \keys_define:nn {stex / definiendum }{
4979   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
4980   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
4981   root     .str_set_x:N    = \l__stex_statements_definiendum_root_str,
4982   gfa      .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
4983 }
4984 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
4985   \str_clear:N \l__stex_statements_definiendum_root_str
4986   \tl_clear:N \l__stex_statements_definiendum_post_tl
4987   \str_clear:N \l__stex_statements_definiendum_gfa_str
4988   \keys_set:nn { stex / definiendum }{ #1 }
4989 }
4990 \NewDocumentCommand \definiendum { O{} m m } {
4991   \__stex_statements_definiendum_args:n { #1 }
4992   \stex_get_symbol:n { #2 }
4993   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4994   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
4995     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```

4996     \tl_set:Nn \l_tmpa_tl { #3 }
4997   } {
4998     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
4999     \tl_set:Nn \l_tmpa_tl {
5000       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
5001     }
5002   }
5003 } {
5004   \tl_set:Nn \l_tmpa_tl { #3 }
5005 }
5006
5007 % TODO root
5008 \stex_html_backend:TF {
5009   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
5010 } {
5011   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
5012 }
5013 }
5014 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

(End definition for definiendum. This function is documented on page 41.)

#### definame

```

5015
5016 \NewDocumentCommand \definame { 0{ } m } {
5017   \__stex_statements_definiendum_args:n { #1 }
5018   % TODO: root
5019   \stex_get_symbol:n { #2 }
5020   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5021   \str_set:Nx \l_tmpa_str {
5022     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5023   }
5024   \str_replace_all:Nnn \l_tmpa_str {-} {~}
5025   \stex_html_backend:TF {
5026     \stex_if_do_html:T {
5027       \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5028         \l_tmpa_str\l__stex_statements_definiendum_post_tl
5029       }
5030     }
5031   } {
5032     \exp_args:Nnx \defemph@uri {
5033       \l_tmpa_str\l__stex_statements_definiendum_post_tl
5034     } { \l_stex_get_symbol_uri_str }
5035   }
5036 }
5037 \stex_deactivate_macro:Nn \definame {definition~environments}
5038
5039 \NewDocumentCommand \Definame { 0{ } m } {
5040   \__stex_statements_definiendum_args:n { #1 }
5041   \stex_get_symbol:n { #2 }
5042   \str_set:Nx \l_tmpa_str {
5043     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5044   }
5045   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}

```



```

5046 \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5047 \stex_html_backend:TF {
5048   \stex_if_do_html:T {
5049     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5050       \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5051     }
5052   }
5053 } {
5054   \exp_args:Nnx \defemph@uri {
5055     \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5056   } { \l_stex_get_symbol_uri_str }
5057 }
5058 }
5059 \stex_deactivate_macro:Nn \Definame {definition-environments}
5060
5061 \NewDocumentCommand \premise { m }{
5062   \stex_annotate:nnn{ premise }{}{ #1 }
5063 }
5064 \NewDocumentCommand \conclusion { m }{
5065   \stex_annotate:nnn{ conclusion }{}{ #1 }
5066 }
5067 \NewDocumentCommand \definiens { 0{} m }{
5068   \str_clear:N \l_stex_get_symbol_uri_str
5069   \tl_if_empty:nF {#1} {
5070     \stex_get_symbol:n { #1 }
5071   }
5072   \str_if_empty:NT \l_stex_get_symbol_uri_str {
5073     \int_compare:nNnTF {\clist_count:N \l__stex_statements_sdefinition_for_clist} = 1 {
5074       \str_set:Nx \l_stex_get_symbol_uri_str {\clist_item:Nn \l__stex_statements_sdefinition
5075     }{
5076       % TODO throw error
5077     }
5078   }
5079   \str_if_eq:eeT {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{module}}
5080   {\l_stex_current_module_str}{
5081     \str_if_eq:eeF {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defin
5082   }{true}{
5083     \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5084     \exp_args:Nx \stex_add_to_current_module:n {
5085       \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5086     }
5087   }
5088 }
5089 \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
5090 }
5091
5092 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
5093 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
5094 \stex_deactivate_macro:Nn \definiens {definition~environments}
5095

```

(End definition for `definame`. This function is documented on page 41.)

`sdefinition`

```

5096
5097 \keys_define:nn {stex / sdefinition }{
5098   type      .str_set_x:N = \sdefinitiontype,
5099   id        .str_set_x:N = \sdefinitionid,
5100   name      .str_set_x:N = \sdefinitionname,
5101   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5102   title     .tl_set:N     = \sdefinitiontitle
5103 }
5104 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
5105   \str_clear:N \sdefinitiontype
5106   \str_clear:N \sdefinitionid
5107   \str_clear:N \sdefinitionname
5108   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5109   \tl_clear:N \sdefinitiontitle
5110   \keys_set:nn { stex / sdefinition }{ #1 }
5111 }
5112
5113 \NewDocumentEnvironment{sdefinition}{0{}}{
5114   \__stex_statements_sdefinition_args:n{ #1 }
5115   \stex_reactivate_macro:N \definiendum
5116   \stex_reactivate_macro:N \definame
5117   \stex_reactivate_macro:N \Definame
5118   \stex_reactivate_macro:N \premise
5119   \stex_reactivate_macro:N \definiens
5120   \stex_if_smsmode:F{
5121     \seq_clear:N \l_tmpb_seq
5122     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5123       \tl_if_empty:nF{ ##1 }{
5124         \stex_get_symbol:n { ##1 }
5125         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5126           \l_stex_get_symbol_uri_str
5127         }
5128       }
5129     }
5130     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5131     \exp_args:Nnnx
5132     \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
5133     \str_if_empty:NF \sdefinitiontype {
5134       \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5135     }
5136     \str_if_empty:NF \sdefinitionname {
5137       \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5138     }
5139     \clist_set:Nn \l_tmpa_clist \sdefinitiontype
5140     \tl_clear:N \l_tmpa_tl
5141     \clist_map_inline:Nn \l_tmpa_clist {
5142       \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
5143         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
5144       }
5145     }
5146     \tl_if_empty:NTF \l_tmpa_tl {
5147       \__stex_statements_sdefinition_start:
5148     }{
5149       \l_tmpa_tl

```

```

5150     }
5151   }
5152   \stex_ref_new_doc_target:n \sdefinitionid
5153   \stex_smsmode_do:
5154 }{
5155   \stex_suppress_html:n {
5156     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5157   }
5158   \stex_if_smsmode:F {
5159     \clist_set:No \l_tmpa_clist \sdefinitiontype
5160     \tl_clear:N \l_tmpa_tl
5161     \clist_map_inline:Nn \l_tmpa_clist {
5162       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
5163         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
5164       }
5165     }
5166     \tl_if_empty:NTF \l_tmpa_tl {
5167       \__stex_statements_sdefinition_end:
5168     }{
5169       \l_tmpa_tl
5170     }
5171     \end{stex_annotate_env}
5172   }
5173 }

```

### **\stexpatchdefinition**

```

5174 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
5175   \stex_par:\noindent\titllemph{Definition\tl_if_empty:NF \sdefinitiontitle {
5176     ~(\sdefinitiontitle)
5177   }~}
5178 }
5179 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\stex_par:\medskip}
5180
5181 \newcommand\stexpatchdefinition[3] [] {
5182   \str_set:Nx \l_tmpa_str{ #1 }
5183   \str_if_empty:NTF \l_tmpa_str {
5184     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
5185     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
5186   }{
5187     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
5188     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
5189   }
5190 }

```

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

### **\inlinedef inline:**

```

5191 \keys_define:nn {stex / inlinedef }{
5192   type      .str_set_x:N = \sdefinitiontype,
5193   id        .str_set_x:N = \sdefinitionid,
5194   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5195   name      .str_set_x:N = \sdefinitionname
5196 }
5197 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {

```

```

5198 \str_clear:N \sdefinitiontype
5199 \str_clear:N \sdefinitionid
5200 \str_clear:N \sdefinitionname
5201 \clist_clear:N \l__stex_statements_sdefinition_for_clist
5202 \keys_set:nn { stex / inlinedef }{ #1 }
5203 }
5204 \NewDocumentCommand \inlinedef { 0{} m } {
5205   \begingroup
5206   \__stex_statements_inlinedef_args:n{ #1 }
5207   \stex_reactivate_macro:N \definiendum
5208   \stex_reactivate_macro:N \definame
5209   \stex_reactivate_macro:N \Definame
5210   \stex_reactivate_macro:N \premise
5211   \stex_reactivate_macro:N \definiens
5212   \stex_ref_new_doc_target:n \sdefinitionid
5213   \stex_if_smsmode:TF{\stex_suppress_html:n {
5214     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5215   }}{
5216     \seq_clear:N \l_tmpb_seq
5217     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5218       \tl_if_empty:nF{ ##1 }{
5219         \stex_get_symbol:n { ##1 }
5220         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5221           \l_stex_get_symbol_uri_str
5222         }
5223       }
5224     }
5225     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5226     \exp_args:Nnx
5227     \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpb_seq {,}}{
5228       \str_if_empty:NF \sdefinitiontype {
5229         \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5230       }
5231       #2
5232       \str_if_empty:NF \sdefinitionname {
5233         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sdefinitionname}}
5234         \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5235       }
5236     }
5237   }
5238   \endgroup
5239   \stex_smsmode_do:
5240 }

```

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

## 32.2 Assertions

**sassertion**

```

5241
5242 \keys_define:nn {stex / sassertion }{
5243   type      .str_set_x:N = \sassertiontype,
5244   id        .str_set_x:N = \sassertionid,

```

```

5245 title .tl_set:N = \sassertiontitle ,
5246 for .clist_set:N = \l__stex_statements_sassertion_for_clist ,
5247 name .str_set_x:N = \sassertionname
5248 }
5249 \cs_new_protected:Nn \l__stex_statements_sassertion_args:n {
5250 \str_clear:N \sassertiontype
5251 \str_clear:N \sassertionid
5252 \str_clear:N \sassertionname
5253 \clist_clear:N \l__stex_statements_sassertion_for_clist
5254 \tl_clear:N \sassertiontitle
5255 \keys_set:nn { stex / sassertion }{ #1 }
5256 }
5257
5258 %\tl_new:N \g__stex_statements_aftergroup_tl
5259
5260 \NewDocumentEnvironment{sassertion}{0{}}{
5261 \l__stex_statements_sassertion_args:n{ #1 }
5262 \stex_reactivate_macro:N \premise
5263 \stex_reactivate_macro:N \conclusion
5264 \stex_if_smsmode:F {
5265 \seq_clear:N \l_tmpb_seq
5266 \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5267 \tl_if_empty:nF{ ##1 }{
5268 \stex_get_symbol:n { ##1 }
5269 \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5270 \l_stex_get_symbol_uri_str
5271 }
5272 }
5273 }
5274 \exp_args:Nnnx
5275 \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
5276 \str_if_empty:NF \sassertiontype {
5277 \stex_annotate_invisible:nnn{type}{\sassertiontype}{ }
5278 }
5279 \str_if_empty:NF \sassertionname {
5280 \stex_annotate_invisible:nnn{statementname}{\sassertionname}{ }
5281 }
5282 \clist_set:Nn \l_tmpa_clist \sassertiontype
5283 \tl_clear:N \l_tmpa_tl
5284 \clist_map_inline:Nn \l_tmpa_clist {
5285 \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
5286 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
5287 }
5288 }
5289 \tl_if_empty:NTF \l_tmpa_tl {
5290 \l__stex_statements_sassertion_start:
5291 }{
5292 \l_tmpa_tl
5293 }
5294 }
5295 \str_if_empty:NTF \sassertionid {
5296 \str_if_empty:NF \sassertionname {
5297 \stex_ref_new_doc_target:n { }
5298 }

```

```

5299 } {
5300   \stex_ref_new_doc_target:n \sassertionid
5301 }
5302 \stex_smsmode_do:
5303 }{
5304   \str_if_empty:NF \sassertionname {
5305     \stex_suppress_html:n{\stex_symdecl_do:nn{ }\sassertionname}}
5306     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5307   }
5308   \stex_if_smsmode:F {
5309     \clist_set:Nn \l_tmpa_clist \sassertiontype
5310     \tl_clear:N \l_tmpa_tl
5311     \clist_map_inline:Nn \l_tmpa_clist {
5312       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
5313         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
5314       }
5315     }
5316     \tl_if_empty:NTF \l_tmpa_tl {
5317       __stex_statements_sassertion_end:
5318     }{
5319       \l_tmpa_tl
5320     }
5321     \end{stex_annotate_env}
5322   }
5323 }

```

**\stexpatchassertion**

```

5324
5325 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
5326   \stex_par:\noindent\titllemph{Assertion~\tl_if_empty:NF \sassertiontitle {
5327     (\sassertiontitle)
5328   }~}
5329 }
5330 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\stex_par:\medskip}
5331
5332 \newcommand\stexpatchassertion[3] [] {
5333   \str_set:Nx \l_tmpa_str{ #1 }
5334   \str_if_empty:NTF \l_tmpa_str {
5335     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
5336     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
5337   }{
5338     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2 }
5339     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
5340   }
5341 }

```

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

**\inlineass** inline:

```

5342 \keys_define:nn {stex / inlineass }{
5343   type      .str_set_x:N = \sassertiontype,
5344   id        .str_set_x:N = \sassertionid,
5345   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
5346   name      .str_set_x:N = \sassertionname

```

```

5347 }
5348 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
5349   \str_clear:N \sassertiontype
5350   \str_clear:N \sassertionid
5351   \str_clear:N \sassertionname
5352   \clist_clear:N \l__stex_statements_sassertion_for_clist
5353   \keys_set:nn { stex / inlineass }{ #1 }
5354 }
5355 \NewDocumentCommand \inlineass { 0{} m } {
5356   \begingroup
5357   \stex_reactivate_macro:N \premise
5358   \stex_reactivate_macro:N \conclusion
5359   \__stex_statements_inlineass_args:n{ #1 }
5360   \str_if_empty:NTF \sassertionid {
5361     \str_if_empty:NF \sassertionname {
5362       \stex_ref_new_doc_target:n {}
5363     }
5364   } {
5365     \stex_ref_new_doc_target:n \sassertionid
5366   }
5367
5368   \stex_if_smsmode:TF{
5369     \str_if_empty:NF \sassertionname {
5370       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5371       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5372     }
5373   }{
5374     \seq_clear:N \l_tmpb_seq
5375     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5376       \tl_if_empty:nF{ ##1 }{
5377         \stex_get_symbol:n { ##1 }
5378         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5379           \l_stex_get_symbol_uri_str
5380         }
5381       }
5382     }
5383     \exp_args:Nnx
5384     \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpb_seq {,}}{
5385       \str_if_empty:NF \sassertiontype {
5386         \stex_annotate_invisible:nnn{typestrings}{\sassertiontype}{}
5387       }
5388       #2
5389       \str_if_empty:NF \sassertionname {
5390         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5391         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5392         \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
5393       }
5394     }
5395   }
5396   \endgroup
5397   \stex_smsmode_do:
5398 }

```

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

## 32.3 Examples

sexample

```

5399
5400 \keys_define:nn {stex / sexample }{
5401   type      .str_set_x:N = \exampletype,
5402   id        .str_set_x:N = \sexampleid,
5403   title     .tl_set:N     = \sexampletile,
5404   name      .str_set_x:N = \sexamplename ,
5405   for       .clist_set:N = \l__stex_statements_sexample_for_clist,
5406 }
5407 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
5408   \str_clear:N \sexampletype
5409   \str_clear:N \sexampleid
5410   \str_clear:N \sexamplename
5411   \tl_clear:N \sexampletile
5412   \clist_clear:N \l__stex_statements_sexample_for_clist
5413   \keys_set:nn { stex / sexample }{ #1 }
5414 }
5415
5416 \NewDocumentEnvironment{sexample}{0{}}{
5417   \__stex_statements_sexample_args:n{ #1 }
5418   \stex_reactivate_macro:N \premise
5419   \stex_reactivate_macro:N \conclusion
5420   \stex_if_smsmode:F {
5421     \seq_clear:N \l_tmpb_seq
5422     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5423       \tl_if_empty:NF{ ##1 }{
5424         \stex_get_symbol:n { ##1 }
5425         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5426           \l_stex_get_symbol_uri_str
5427         }
5428       }
5429     }
5430     \exp_args:Nnnx
5431     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpb_seq {,}}
5432     \str_if_empty:NF \sexampletype {
5433       \stex_annotate_invisible:nnn{typestrings}{\sexampletype}{}
5434     }
5435     \str_if_empty:NF \sexamplename {
5436       \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
5437     }
5438     \clist_set:Nn \l_tmpa_clist \sexampletype
5439     \tl_clear:N \l_tmpa_tl
5440     \clist_map_inline:Nn \l_tmpa_clist {
5441       \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
5442         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
5443       }
5444     }
5445     \tl_if_empty:NTF \l_tmpa_tl {
5446       \__stex_statements_sexample_start:
5447     }{
5448       \l_tmpa_tl
5449     }

```



```

5450 }
5451 \str_if_empty:NF \sexampleid {
5452   \stex_ref_new_doc_target:n \sexampleid
5453 }
5454 \stex_smsmode_do:
5455 ){
5456   \str_if_empty:NF \sexamplename {
5457     \stex_suppress_html:n{\stex_symdecl_do:nn}{\sexamplename}}
5458   }
5459   \stex_if_smsmode:F {
5460     \clist_set:Nn \l_tmpa_clist \sexamplotype
5461     \tl_clear:N \l_tmpa_tl
5462     \clist_map_inline:Nn \l_tmpa_clist {
5463       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
5464         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
5465       }
5466     }
5467     \tl_if_empty:NTF \l_tmpa_tl {
5468       \__stex_statements_sexample_end:
5469     }{
5470       \l_tmpa_tl
5471     }
5472     \end{stex_annotate_env}
5473   }
5474 }

```

**\stexpatchexample**

```

5475
5476 \cs_new_protected:Nn \__stex_statements_sexample_start: {
5477   \stex_par:\noindent\titllemph{Example~\tl_if_empty:NF \sexampltitle {
5478     (\sexampltitle)
5479   }~}
5480 }
5481 \cs_new_protected:Nn \__stex_statements_sexample_end: {\stex_par:\medskip}
5482
5483 \newcommand\stexpatchexample[3]{} {
5484   \str_set:Nx \l_tmpa_str{ #1 }
5485   \str_if_empty:NTF \l_tmpa_str {
5486     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
5487     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
5488   }{
5489     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
5490     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
5491   }
5492 }

```

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

**\inlineex** inline:

```

5493 \keys_define:nn {stex / inlineex }{
5494   type      .str_set_x:N = \sexamplotype,
5495   id        .str_set_x:N = \sexampleid,
5496   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
5497   name      .str_set_x:N = \sexamplename

```

```

5498 }
5499 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
5500   \str_clear:N \sexamplotype
5501   \str_clear:N \sexampleid
5502   \str_clear:N \sexamplename
5503   \clist_clear:N \l__stex_statements_sexample_for_clist
5504   \keys_set:nn { stex / inlineex }{ #1 }
5505 }
5506 \NewDocumentCommand \inlineex { 0{ } m } {
5507   \beginngroup
5508   \stex_reactivate_macro:N \premise
5509   \stex_reactivate_macro:N \conclusion
5510   \__stex_statements_inlineex_args:n{ #1 }
5511   \str_if_empty:NF \sexampleid {
5512     \stex_ref_new_doc_target:n \sexampleid
5513   }
5514   \stex_if_smsmode:TF{
5515     \str_if_empty:NF \sexamplename {
5516       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5517     }
5518   }{
5519     \seq_clear:N \l_tmpb_seq
5520     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5521       \tl_if_empty:nF{ ##1 }{
5522         \stex_get_symbol:n { ##1 }
5523         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5524           \l_stex_get_symbol_uri_str
5525         }
5526       }
5527     }
5528     \exp_args:Nnx
5529     \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpb_seq {},}}{
5530       \str_if_empty:NF \sexamplotype {
5531         \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{ }
5532       }
5533       #2
5534       \str_if_empty:NF \sexamplename {
5535         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5536         \stex_annotate_invisible:nnn{statementname}{\sexamplename}{ }
5537       }
5538     }
5539   }
5540   \endgroup
5541   \stex_smsmode_do:
5542 }

```

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

## 32.4 Logical Paragraphs

sparagraph

```

5543 \keys_define:nn { stex / sparagraph } {
5544   id          .str_set_x:N    = \sparagraphid ,

```

```

5545 title .tl_set:N = \l_stex_sparagraph_title_tl ,
5546 type .str_set_x:N = \sparagraphtype ,
5547 for .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5548 from .tl_set:N = \sparagraphfrom ,
5549 to .tl_set:N = \sparagraphto ,
5550 start .tl_set:N = \l_stex_sparagraph_start_tl ,
5551 name .str_set:N = \sparagraphname ,
5552 imports .tl_set:N = \l__stex_statements_sparagraph_imports_tl
5553 }
5554
5555 \cs_new_protected:Nn \stex_sparagraph_args:n {
5556 \tl_clear:N \l_stex_sparagraph_title_tl
5557 \tl_clear:N \sparagraphfrom
5558 \tl_clear:N \sparagraphto
5559 \tl_clear:N \l_stex_sparagraph_start_tl
5560 \tl_clear:N \l__stex_statements_sparagraph_imports_tl
5561 \str_clear:N \sparagraphid
5562 \str_clear:N \sparagraphtype
5563 \clist_clear:N \l__stex_statements_sparagraph_for_clist
5564 \str_clear:N \sparagraphname
5565 \keys_set:nn { stex / sparagraph }{ #1 }
5566 }
5567 \newif\if@in@omtext\@in@omtextfalse
5568
5569 \NewDocumentEnvironment {sparagraph} { 0{} } {
5570 \stex_sparagraph_args:n { #1 }
5571 \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5572 \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
5573 }{
5574 \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
5575 }
5576 \@in@omtexttrue
5577 \stex_if_smsmode:F {
5578 \seq_clear:N \l_tmpb_seq
5579 \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5580 \tl_if_empty:NF{ ##1 }{
5581 \stex_get_symbol:n { ##1 }
5582 \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5583 \l_stex_get_symbol_uri_str
5584 }
5585 }
5586 }
5587 \exp_args:Nnnx
5588 \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}
5589 \str_if_empty:NF \sparagraphtype {
5590 \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}}
5591 }
5592 \str_if_empty:NF \sparagraphfrom {
5593 \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}}
5594 }
5595 \str_if_empty:NF \sparagraphto {
5596 \stex_annotate_invisible:nnn{to}{\sparagraphto}{}}
5597 }
5598 \str_if_empty:NF \sparagraphname {

```

```

5599     \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{ }
5600   }
5601   \clist_set:No \l_tmpa_clist \sparagraphtype
5602   \tl_clear:N \l_tmpa_tl
5603   \clist_map_inline:Nn \sparagraphtype {
5604     \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
5605       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
5606     }
5607   }
5608   \stex_csl_to_imports:No \usemodule \l__stex_statements_sparagraph_imports_tl
5609   \tl_if_empty:NTF \l_tmpa_tl {
5610     \__stex_statements_sparagraph_start:
5611   }{
5612     \l_tmpa_tl
5613   }
5614 }
5615 \clist_set:No \l_tmpa_clist \sparagraphtype
5616 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5617 {
5618   \stex_reactivate_macro:N \definiendum
5619   \stex_reactivate_macro:N \definame
5620   \stex_reactivate_macro:N \Definame
5621   \stex_reactivate_macro:N \premise
5622   \stex_reactivate_macro:N \definiens
5623 }
5624 \str_if_empty:NTF \sparagraphid {
5625   \str_if_empty:NTF \sparagraphname {
5626     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5627       \stex_ref_new_doc_target:n {}
5628     }
5629   } {
5630     \stex_ref_new_doc_target:n {}
5631   }
5632 } {
5633   \stex_ref_new_doc_target:n \sparagraphid
5634 }
5635 \exp_args:NNx
5636 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5637   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5638     \tl_if_empty:nF{ ##1 }{
5639       \stex_get_symbol:n { ##1 }
5640       \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5641     }
5642   }
5643 }
5644 \stex_smsmode_do:
5645 \ignorespacesandpars
5646 }{
5647   \str_if_empty:NF \sparagraphname {
5648     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5649     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5650   }
5651   \stex_if_smsmode:F {
5652     \clist_set:No \l_tmpa_clist \sparagraphtype

```

```

5653 \tl_clear:N \l_tmpa_tl
5654 \clist_map_inline:Nn \l_tmpa_clist {
5655   \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
5656     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
5657   }
5658 }
5659 \tl_if_empty:NTF \l_tmpa_tl {
5660   \__stex_statements_sparagraph_end:
5661 }{
5662   \l_tmpa_tl
5663 }
5664 \end{stex_annotate_env}
5665 }
5666 }

```

### **\stexpatchparagraph**

```

5667
5668 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
5669   \stex_par:\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5670     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
5671       \titleemph{\l_stex_sparagraph_title_tl}:~
5672     }
5673   }{
5674     \titleemph{\l_stex_sparagraph_start_tl}~
5675   }
5676 }
5677 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\stex_par:\medskip}
5678
5679 \newcommand\stexpatchparagraph[3] [] {
5680   \str_set:Nx \l_tmpa_str{ #1 }
5681   \str_if_empty:NTF \l_tmpa_str {
5682     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
5683     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
5684   }{
5685     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
5686     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
5687   }
5688 }
5689
5690 \keys_define:nn { stex / inlinepara } {
5691   id      .str_set_x:N = \sparagraphid ,
5692   type    .str_set_x:N = \sparagraphtype ,
5693   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5694   from    .tl_set:N    = \sparagraphfrom ,
5695   to      .tl_set:N    = \sparagraphto ,
5696   name    .str_set:N   = \sparagraphname
5697 }
5698 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
5699   \tl_clear:N \sparagraphfrom
5700   \tl_clear:N \sparagraphto
5701   \str_clear:N \sparagraphid
5702   \str_clear:N \sparagraphtype
5703   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5704   \str_clear:N \sparagraphname

```

```

5705 \keys_set:nn { stex / inlinepara }{ #1 }
5706 }
5707 \NewDocumentCommand \inlinepara { 0{} m } {
5708   \beginingroup
5709   \__stex_statements_inlinepara_args:n{ #1 }
5710   \clist_set:No \l_tmpa_clist \sparagraphtype
5711   \str_if_empty:NTF \sparagraphid {
5712     \str_if_empty:NTF \sparagraphname {
5713       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5714         \stex_ref_new_doc_target:n {}
5715       }
5716     } {
5717       \stex_ref_new_doc_target:n {}
5718     }
5719   } {
5720     \stex_ref_new_doc_target:n \sparagraphid
5721   }
5722   \stex_if_smsmode:TF{
5723     \str_if_empty:NF \sparagraphname {
5724       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
5725     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5726   }
5727 }{
5728   \seq_clear:N \l_tmpb_seq
5729   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5730     \tl_if_empty:nF{ ##1 }{
5731       \stex_get_symbol:n { ##1 }
5732       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5733         \l_stex_get_symbol_uri_str
5734       }
5735     }
5736   }
5737   \exp_args:Nnx
5738   \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}{
5739     \str_if_empty:NF \sparagraphtype {
5740       \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
5741     }
5742     \str_if_empty:NF \sparagraphfrom {
5743       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5744     }
5745     \str_if_empty:NF \sparagraphto {
5746       \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5747     }
5748     \str_if_empty:NF \sparagraphname {
5749       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
5750     \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
5751     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5752   }
5753   \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5754     \clist_map_inline:Nn \l_tmpb_seq {
5755       \stex_ref_new_sym_target:n {##1}
5756     }
5757   }
5758   #2

```

```

5759     }
5760   }
5761   \endgroup
5762   \stex_smsmode_do:
5763 }
5764

```

*(End definition for \stexpatchparagraph. This function is documented on page [47](#).)*

```

5765 \endpackage

```

## Chapter 33

# The Implementation

```
5766 <*package>
5767 <@@=stex_sproof>
5768
5769 %%%%%%%%%% sproof.dtx %%%%%%%%%%
5770
```

### 33.1 Proofs

We first define some keys for the proof environment.

```
5771 \keys_define:nn { stex / spf } {
5772   id          .str_set_x:N = \spfid,
5773   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,
5774   from        .tl_set:N    = \l__stex_sproof_spf_from_tl ,
5775   proofend    .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
5776   type        .str_set_x:N = \spftype,
5777   title       .tl_set:N    = \spftitle,
5778   continues   .tl_set:N    = \l__stex_sproof_spf_continues_tl,
5779   functions   .tl_set:N    = \l__stex_sproof_spf_functions_tl,
5780   method      .tl_set:N    = \l__stex_sproof_spf_method_tl
5781 }
5782 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
5783   \str_clear:N \spfid
5784   \tl_clear:N \l__stex_sproof_spf_for_tl
5785   \tl_clear:N \l__stex_sproof_spf_from_tl
5786   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
5787   \str_clear:N \spftype
5788   \tl_clear:N \spftitle
5789   \tl_clear:N \l__stex_sproof_spf_continues_tl
5790   \tl_clear:N \l__stex_sproof_spf_functions_tl
5791   \tl_clear:N \l__stex_sproof_spf_method_tl
5792   \bool_set_false:N \l__stex_sproof_inc_counter_bool
5793   \keys_set:nn { stex / spf }{ #1 }
5794 }
```

\c\_\_stex\_sproof\_flow\_str

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

```
5795 \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,  $\text{\LaTeX}$  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.

```

5796 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
5797 \cs_new_protected:Npn \sproofnumber {
5798   \int_set:Nn \l_tmpa_int {1}
5799   \bool_while_do:nn {
5800     \int_compare_p:nNn {
5801       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5802     } > 0
5803   }{
5804     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
5805     \int_incr:N \l_tmpa_int
5806   }
5807 }
5808 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
5809   \int_set:Nn \l_tmpa_int {1}
5810   \bool_while_do:nn {
5811     \int_compare_p:nNn {
5812       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5813     } > 0
5814   }{
5815     \int_incr:N \l_tmpa_int
5816   }
5817   \int_compare:nNnF \l_tmpa_int = 1 {
5818     \int_decr:N \l_tmpa_int
5819   }
5820   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
5821     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1
5822   }
5823 }
5824
5825 \cs_new_protected:Npn \__stex_sproof_add_counter: {
5826   \int_set:Nn \l_tmpa_int {1}
5827   \bool_while_do:nn {
5828     \int_compare_p:nNn {
5829       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5830     } > 0
5831   }{
5832     \int_incr:N \l_tmpa_int
5833   }
5834   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 1 }
5835 }
5836
5837 \cs_new_protected:Npn \__stex_sproof_remove_counter: {
5838   \int_set:Nn \l_tmpa_int {1}
5839   \bool_while_do:nn {

```

```

5840 \int_compare_p:nNn {
5841 \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5842 } > 0
5843 }{
5844 \int_incr:N \l_tmpa_int
5845 }
5846 \int_decr:N \l_tmpa_int
5847 \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 0 }
5848 }

```

**\sproofend** This macro places a little box at the end of the line if there is space, or at the end of the next line if there isn't

```

5849 \def\sproof@box{
5850 \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
5851 }
5852 \def\sproofend{
5853 \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
5854 \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
5855 }
5856 }

```

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

**spf@\*@kw**

```

5857 \def\spf@proofsketch@kw{Proof~Sketch}
5858 \def\spf@proof@kw{Proof}
5859 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

5860 \AddToHook{begindocument}{
5861 \ltx@ifpackageloaded{babel}{
5862 \makeatletter
5863 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5864 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5865 \input{sproof-ngerman.ldf}
5866 }
5867 \clist_if_in:NnT \l_tmpa_clist {finnish}{
5868 \input{sproof-finnish.ldf}
5869 }
5870 \clist_if_in:NnT \l_tmpa_clist {french}{
5871 \input{sproof-french.ldf}
5872 }
5873 \clist_if_in:NnT \l_tmpa_clist {russian}{
5874 \input{sproof-russian.ldf}
5875 }
5876 \makeatother
5877 }{}
5878 }

```

**spfsketch**

```

5879 \newcommand\spsketch[2][]{
5880 \begin{group}
5881 \let \premise \stex_proof_premise:

```

```

5882 \__stex_sproof_spf_args:n{#1}
5883 \stex_if_smsmode:TF {
5884   \str_if_empty:NF \spfid {
5885     \stex_ref_new_doc_target:n \spfid
5886   }
5887 }{
5888   \seq_clear:N \l_tmpa_seq
5889   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5890     \tl_if_empty:nF{ ##1 }{
5891       \stex_get_symbol:n { ##1 }
5892       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5893         \l_stex_get_symbol_uri_str
5894       }
5895     }
5896   }
5897   \exp_args:Nnx
5898   \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
5899     \str_if_empty:NF \spftype {
5900       \stex_annotate_invisible:nnn{type}{\spftype}{
5901         }
5902       \clist_set:No \l_tmpa_clist \spftype
5903       \tl_set:Nn \l_tmpa_tl {
5904         \titleemph{
5905           \tl_if_empty:NTF \spftitle {
5906             \spf@proofsketch@kw
5907           }{
5908             \spftitle
5909           }
5910         }::~
5911       }
5912       \clist_map_inline:Nn \l_tmpa_clist {
5913         \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5914           \tl_clear:N \l_tmpa_tl
5915         }
5916       }
5917       \str_if_empty:NF \spfid {
5918         \stex_ref_new_doc_target:n \spfid
5919       }
5920       \l_tmpa_tl #2 \sproofend
5921     }
5922   }
5923   \endgroup
5924   \stex_smsmode_do:
5925 }
5926

```

(End definition for `spfsketch`. This function is documented on page 44.)

**spfeq** This is very similar to `spfsketch`, but uses a computation array<sup>1415</sup>

```

5927 \newenvironment{spfeq}[2][ ]{
5928   \__stex_sproof_spf_args:n{#1}

```

<sup>14</sup>EDNOTE: This should really be more like a tabular with an ensuremath in it. or invoke text on the last column

<sup>15</sup>EDNOTE: document above

```

5929 \let \premise \stex_proof_premise:
5930 \stex_if_smsmode:TF {
5931   \str_if_empty:NF \spfid {
5932     \stex_ref_new_doc_target:n \spfid
5933   }
5934 }{
5935   \seq_clear:N \l_tmpa_seq
5936   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5937     \tl_if_empty:nF{ ##1 }{
5938       \stex_get_symbol:n { ##1 }
5939       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5940         \l_stex_get_symbol_uri_str
5941       }
5942     }
5943   }
5944   \exp_args:Nnnx
5945   \begin{stex_annotate_env}{spfeq}{\seq_use:Nn \l_tmpa_seq {,}}
5946   \str_if_empty:NF \spftype {
5947     \stex_annotate_invisible:nnn{type}{\spftype}{}
5948   }
5949
5950   \clist_set:No \l_tmpa_clist \spftype
5951   \tl_clear:N \l_tmpa_tl
5952   \clist_map_inline:Nn \l_tmpa_clist {
5953     \tl_if_exist:cT {__stex_sproof_spfeq_##1_start:}{
5954       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_start:}}
5955     }
5956     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5957       \tl_set:Nn \l_tmpa_tl {\use:n{}}
5958     }
5959   }
5960   \tl_if_empty:NTF \l_tmpa_tl {
5961     \__stex_sproof_spfeq_start:
5962   }{
5963     \l_tmpa_tl
5964   }{~#2}
5965   \str_if_empty:NF \spfid {
5966     \stex_ref_new_doc_target:n \spfid
5967   }
5968   \begin{displaymath}\begin{array}{rcll}
5969 \end{array}
5970 \stex_smsmode_do:
5971 }{
5972   \stex_if_smsmode:F {
5973     \end{array}\end{displaymath}
5974     \clist_set:No \l_tmpa_clist \spftype
5975     \tl_clear:N \l_tmpa_tl
5976     \clist_map_inline:Nn \l_tmpa_clist {
5977       \tl_if_exist:cT {__stex_sproof_spfeq_##1_end:}{
5978         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_end:}}
5979       }
5980     }
5981     \tl_if_empty:NTF \l_tmpa_tl {
5982       \__stex_sproof_spfeq_end:

```

```

5983   }{
5984     \l_tmpa_tl
5985   }
5986   \end{stex_annotate_env}
5987 }
5988 }
5989
5990 \cs_new_protected:Nn \__stex_sproof_spfeq_start: {
5991   \titleemph{
5992     \tl_if_empty:NTF \spftitle {
5993       \spf@proof@kw
5994     }{
5995       \spftitle
5996     }
5997   }:
5998 }
5999 \cs_new_protected:Nn \__stex_sproof_spfeq_end: {\sproofend}
6000
6001 \newcommand\stexpatchspfeq[3] [] {
6002   \str_set:Nx \l_tmpa_str{ #1 }
6003   \str_if_empty:NTF \l_tmpa_str {
6004     \tl_set:Nn \__stex_sproof_spfeq_start: { #2 }
6005     \tl_set:Nn \__stex_sproof_spfeq_end: { #3 }
6006   }{
6007     \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_start:\endcsname{ #2 }
6008     \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_end:\endcsname{ #3 }
6009   }
6010 }
6011

```

(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.

```

6012 \newenvironment{sproof}[2] []{
6013   \let \premise \stex_proof_premise:
6014   \intarray_gzero:N \l__stex_sproof_counter_intarray
6015   \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
6016   \__stex_sproof_spf_args:n{#1}
6017   \stex_if_smsmode:TF {
6018     \str_if_empty:NF \spfid {
6019       \stex_ref_new_doc_target:n \spfid
6020     }
6021   }{
6022     \seq_clear:N \l_tmpa_seq
6023     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6024       \tl_if_empty:nF{ ##1 }{
6025         \stex_get_symbol:n { ##1 }
6026         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6027           \l_stex_get_symbol_uri_str
6028         }
6029       }
6030     }

```

```

6031 \exp_args:Nnnx
6032 \begin{stex_annotate_env}{sproof}{\seq_use:Nn \l_tmpa_seq {,}}
6033 \str_if_empty:NF \spftype {
6034 \stex_annotate_invisible:nnn{type}{\spftype}{}}
6035 }
6036
6037 \clist_set:No \l_tmpa_clist \spftype
6038 \tl_clear:N \l_tmpa_tl
6039 \clist_map_inline:Nn \l_tmpa_clist {
6040 \tl_if_exist:cT {__stex_sproof_sproof_##1_start:}{
6041 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_start:}}
6042 }
6043 \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6044 \tl_set:Nn \l_tmpa_tl {\use:n{}}
6045 }
6046 }
6047 \tl_if_empty:NTF \l_tmpa_tl {
6048 \__stex_sproof_sproof_start:
6049 }{
6050 \l_tmpa_tl
6051 }{~#2}
6052 \str_if_empty:NF \spfid {
6053 \stex_ref_new_doc_target:n \spfid
6054 }
6055 \begin{description}
6056 }
6057 \stex_smsmode_do:
6058 }{
6059 \stex_if_smsmode:F{
6060 \end{description}
6061 \clist_set:No \l_tmpa_clist \spftype
6062 \tl_clear:N \l_tmpa_tl
6063 \clist_map_inline:Nn \l_tmpa_clist {
6064 \tl_if_exist:cT {__stex_sproof_sproof_##1_end:}{
6065 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_end:}}
6066 }
6067 }
6068 \tl_if_empty:NTF \l_tmpa_tl {
6069 \__stex_sproof_sproof_end:
6070 }{
6071 \l_tmpa_tl
6072 }
6073 \end{stex_annotate_env}
6074 }
6075 }
6076
6077 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
6078 \par\noindent\titleemph{
6079 \tl_if_empty:NTF \spftype {
6080 \spf@proof@kw
6081 }{
6082 \spftype
6083 }
6084 }::

```

```

6085 }
6086 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
6087
6088 \newcommand\stexpatchproof[3] [] {
6089   \str_set:Nx \l_tmpa_str{ #1 }
6090   \str_if_empty:NTF \l_tmpa_str {
6091     \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
6092     \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
6093   }{
6094     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
6095     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
6096   }
6097 }

```

**\spfidea**

```

6098 \newcommand\spfidea[2] []{
6099   \__stex_sproof_spf_args:n{#1}
6100   \titleemph{
6101     \tl_if_empty:NTF \spftype {Proof~Idea}{
6102       \spftype
6103     } :
6104   }~#2
6105   \sproofend
6106 }

```

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

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**

```

6107 \newenvironment{spfstep}[1] []{
6108   \__stex_sproof_spf_args:n{#1}
6109   \stex_if_smsmode:TF {
6110     \str_if_empty:NF \spfid {
6111       \stex_ref_new_doc_target:n \spfid
6112     }
6113   }{
6114     \@in@omtexttrue
6115     \seq_clear:N \l_tmpa_seq
6116     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6117       \tl_if_empty:nF{ ##1 }{
6118         \stex_get_symbol:n { ##1 }
6119         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6120           \l_stex_get_symbol_uri_str
6121         }
6122       }
6123     }
6124     \exp_args:Nnnx
6125     \begin{stex_annotate_env}{spfstep}{\seq_use:Nn \l_tmpa_seq {,}}
6126     \str_if_empty:NF \spftype {
6127       \stex_annotate_invisible:nnn{type}{\spftype}{ }

```

```

6128   }
6129   \clist_set:Nn \l_tmpa_clist \spftype
6130   \tl_set:Nn \l_tmpa_tl {
6131     \item[\sproofnumber]
6132     \bool_set_true:N \l__stex_sproof_inc_counter_bool
6133   }
6134   \clist_map_inline:Nn \l_tmpa_clist {
6135     \exp_args:Nn \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6136       \tl_clear:N \l_tmpa_tl
6137     }
6138   }
6139   \l_tmpa_tl
6140   \tl_if_empty:NF \spftitle {
6141     {(\titleemph{\spftitle})\enspace}
6142   }
6143   \str_if_empty:NF \spfid {
6144     \stex_ref_new_doc_target:n \spfid
6145   }
6146 }
6147 \stex_smsmode_do:
6148 \ignorespacesandpars
6149 }{
6150   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6151     \__stex_sproof_inc_counter:
6152   }
6153   \stex_if_smsmode:F {
6154     \end{stex_annotate_env}
6155   }
6156 }

```

**spfcomment**

```

6157 \newenvironment{spfcomment}[1][]{
6158   \__stex_sproof_spf_args:n{#1}
6159   \clist_set:Nn \l_tmpa_clist \spftype
6160   \tl_set:Nn \l_tmpa_tl {
6161     \item[\sproofnumber]
6162     \bool_set_true:N \l__stex_sproof_inc_counter_bool
6163   }
6164   \clist_map_inline:Nn \l_tmpa_clist {
6165     \exp_args:Nn \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6166       \tl_clear:N \l_tmpa_tl
6167     }
6168   }
6169   \l_tmpa_tl
6170 }{
6171   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6172     \__stex_sproof_inc_counter:
6173   }
6174 }

```

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.



```

6175 \newenvironment{subproof}[2][]{
6176   \_stex_sproof_spf_args:n{#1}
6177   \stex_if_smsmode:TF{
6178     \str_if_empty:NF \spfid {
6179       \stex_ref_new_doc_target:n \spfid
6180     }
6181   }{
6182     \seq_clear:N \l_tmpa_seq
6183     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6184       \tl_if_empty:nF{ ##1 }{
6185         \stex_get_symbol:n { ##1 }
6186         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6187           \l_stex_get_symbol_uri_str
6188         }
6189       }
6190     }
6191     \exp_args:Nnnx
6192     \begin{stex_annotate_env}{subproof}{\seq_use:Nn \l_tmpa_seq {},}
6193     \str_if_empty:NF \spftype {
6194       \stex_annotate_invisible:nnn{type}{\spftype}{}
6195     }
6196
6197     \clist_set:No \l_tmpa_clist \spftype
6198     \tl_set:Nn \l_tmpa_tl {
6199       \item[\sproofnumber]
6200       \bool_set_true:N \l__stex_sproof_inc_counter_bool
6201     }
6202     \clist_map_inline:Nn \l_tmpa_clist {
6203       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6204         \tl_clear:N \l_tmpa_tl
6205       }
6206     }
6207     \l_tmpa_tl
6208     \tl_if_empty:NF \spftitle {
6209       {(\titleemph{\spftitle})\enspace}
6210     }
6211     {~#2}
6212     \str_if_empty:NF \spfid {
6213       \stex_ref_new_doc_target:n \spfid
6214     }
6215   }
6216   \_stex_sproof_add_counter:
6217   \stex_smsmode_do:
6218 }{
6219   \_stex_sproof_remove_counter:
6220   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6221     \_stex_sproof_inc_counter:
6222   }
6223   \stex_if_smsmode:F{
6224     \end{stex_annotate_env}
6225   }
6226 }

```

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

```

6227 \newenvironment{spfcases}[2] [] {
6228   \tl_if_empty:nTF{#1}{
6229     \begin{subproof}[method=by-cases]{#2}
6230   }{
6231     \begin{subproof}[#1,method=by-cases]{#2}
6232   }
6233 }{
6234   \end{subproof}
6235 }

```

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

```

6236 \newenvironment{spfcase}[2] [] {
6237   \__stex_sproof_spf_args:n{#1}
6238   \stex_if_smsmode:TF {
6239     \str_if_empty:NF \spfid {
6240       \stex_ref_new_doc_target:n \spfid
6241     }
6242   }{
6243     \seq_clear:N \l_tmpa_seq
6244     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6245       \tl_if_empty:nF{ ##1 }{
6246         \stex_get_symbol:n { ##1 }
6247         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6248           \l_stex_get_symbol_uri_str
6249         }
6250       }
6251     }
6252     \exp_args:Nnnx
6253     \begin{stex_annotate_env}{spfcase}{\seq_use:Nn \l_tmpa_seq {,}}
6254     \str_if_empty:NF \spftype {
6255       \stex_annotate_invisible:nnn{type}{\spftype}{ }
6256     }
6257     \clist_set:No \l_tmpa_clist \spftype
6258     \tl_set:Nn \l_tmpa_tl {
6259       \item[\sproofnumber]
6260       \bool_set_true:N \l__stex_sproof_inc_counter_bool
6261     }
6262     \clist_map_inline:Nn \l_tmpa_clist {
6263       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6264         \tl_clear:N \l_tmpa_tl
6265       }
6266     }
6267     \l_tmpa_tl
6268     \tl_if_empty:nF{#2}{
6269       \titleemph{#2}:~
6270     }
6271   }
6272   \__stex_sproof_add_counter:
6273   \stex_smsmode_do:
6274 }{
6275   \__stex_sproof_remove_counter:
6276   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6277     \__stex_sproof_inc_counter:

```

```

6278 }
6279 \stex_if_smsmode:F{
6280   \clist_set:No \l_tmpa_clist \spftype
6281   \tl_set:Nn \l_tmpa_tl{\sproofend}
6282   \clist_map_inline:Nn \l_tmpa_clist {
6283     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6284       \tl_clear:N \l_tmpa_tl
6285     }
6286   }
6287   \l_tmpa_tl
6288   \end{stex_annotate_env}
6289 }
6290 }

```

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

```

6291 \newcommand\spfcasesketch[3] [] {
6292   \begin{spfcase}[#1]{#2}#3\end{spfcase}
6293 }

```

## 33.2 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.

```

6294 \keys_define:nn { stex / just }{
6295   id      .str_set_x:N = \l__stex_sproof_just_id_str,
6296   method  .tl_set:N   = \l__stex_sproof_just_method_tl,
6297   premises .tl_set:N   = \l__stex_sproof_just_premises_tl,
6298   args    .tl_set:N   = \l__stex_sproof_just_args_tl
6299 }

```

The next three environments and macros are purely semantic, so we ignore the keyval arguments for now and only display the content.<sup>16</sup>

**\spfjust**

```

6300 \newcommand\spfjust[1] [] {}

```

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

**\premise**

```

6301 \newcommand\stex_proof_premise:[2] [] {#2}

```

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

**\justarg** the **\justarg** macro is purely semantic, so we ignore the keyval arguments for now and only display the content.

```

6302 \newcommand\justarg[2] [] {#2}
6303 \</package>

```

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

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

---

<sup>16</sup>EdNOTE: need to do something about the premise in draft mode.

## Chapter 34

# STEX -Others Implementation

```
6304 <*package>
6305
6306 %%%%%%%%%% others.dtx %%%%%%%%%%
6307
6308 <@@=stex_others>
        Warnings and error messages
6309 % None

\MSC Math subject classifier

6310 \NewDocumentCommand \MSC {m} {
6311 % TODO
6312 }

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

6313 \@ifpackageloaded{tikzinput}{
6314 \RequirePackage{stex-tikzinput}
6315 }{}
6316
6317 \bool_if:NT \c_stex_persist_mode_bool {
6318 \input{\jobname.sms}
6319 \prop_if_exist:NT\c_stex_mathhub_main_manifest_prop{
6320 \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
6321 \l_tmpa_str
6322 \prop_set_eq:cN { c_stex_mathhub\_l_tmpa_str_manifest_prop }
6323 \c_stex_mathhub_main_manifest_prop
6324 \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
6325 }
6326 }

6327 </package>
```

## Chapter 35

# STEX -Metatheory Implementation

```
6328 <*package>
6329 <@@=stex_modules>
6330
6331 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
6332
6333 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX/meta}
6334 \begingroup
6335 \stex_module_setup:nn{
6336   ns=\c_stex_metatheory_ns_str,
6337   meta=NONE
6338 }{Metatheory}
6339 \stex_reactivate_macro:N \symdecl
6340 \stex_reactivate_macro:N \notation
6341 \stex_reactivate_macro:N \symdef
6342 \ExplSyntaxOff
6343 \csname stex_suppress_html:n\endcsname{
6344   % is-a (a:A, a \in A, a is an A, etc.)
6345   \symdecl{isa}[args=ai]
6346   \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
6347   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
6348   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
6349
6350   % bind (\forall, \Pi, \lambda etc.)
6351   \symdecl{bind}[args=Bi]
6352   \notation{bind}[forall]{\comp\forall #1.;#2}{##1 \comp, ##2}
6353   \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6354   \notation{bind}[depfun]{\comp( #1 \comp{} \; \to \; ) #2}{##1 \comp, ##2}
6355
6356   % implicit bind
6357   \symdef{implicitbind}[args=Bi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6358
6359   % dummy variable
6360   \symdecl{dummyvar}
6361   \notation{dummyvar}[underscore]{\comp\_}
6362   \notation{dummyvar}[dot]{\comp\cdot}
```

```

6363 \notation{dummyvar}[dash]{\comp{\rm --}}
6364
6365 %fromto (function space, Hom-set, implication etc.)
6366 \symdecl{fromto}[args=ai]
6367 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
6368 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
6369
6370 % mapto (lambda etc.)
6371 \symdecl{mapto}[args=Bi]
6372 %\notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
6373 %\notation{mapto}[lambda]{\comp\lambda #1 \comp.; #2}{#1 \comp, #2}
6374 %\notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.; #2}{#1 \comp, #2}
6375
6376 % function/operator application
6377 \symdecl{apply}[args=ia]
6378 \notation{apply}[prec=0;0x\infpres,parens]{#1 \comp( #2 \comp)}{##1 \comp, ##2}
6379 \notation{apply}[prec=0;0x\infpres,lambda]{#1 \; #2 }{##1 \; ; ##2}
6380
6381 % collection of propositions/booleans/truth values
6382 \symdecl{prop}[name=proposition]
6383 \notation{prop}[prop]{\comp{\rm prop}}
6384 \notation{prop}[BOOL]{\comp{\rm BOOL}}
6385
6386 \symdecl{judgmentholds}[args=1]
6387 \notation{judgmentholds}[vdash,op=\vdash]{\comp\vdash\; ; #1}
6388
6389 % sequences
6390 \symdecl{seqtype}[args=1]
6391 \notation{seqtype}[kleene]{#1^{\comp\ast}}
6392
6393 \symdecl{seqexpr}[args=a]
6394 \notation{seqexpr}[angle,prec=nobrackets]{\comp\langle #1\comp\rangle}{##1\comp,##2}
6395
6396 \symdef{seqmap}[args=abi,setlike]{\comp{\#3 \comp| #2\comp\in \dobrackets{#1} \comp\}}{##1\comp,##2}
6397 \symdef{seqprepend}[args=ia]{#1 \comp{:} #2}{##1 \comp, ##2}
6398 \symdef{seqappend}[args=ai]{#1 \comp{:} #2}{##1 \comp, ##2}
6399 \symdef{seqfoldleft}[args=iabbi]{ \comp{foldl}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\comp}
6400 \symdef{seqfoldright}[args=iabbi,op=foldr]{ \comp{foldr}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\comp}
6401 \symdef{seqhead}[args=a]{\comp{head}\dobrackets{#1}}{##1 \comp, ##2}
6402 \symdef{seqtail}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6403 \symdef{seqlast}[args=a]{\comp{last}\dobrackets{#1}}{##1 \comp, ##2}
6404 \symdef{seqinit}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6405
6406 \symdef{sequence-index}[args=2,li,prec=nobrackets]{\comp{#1}_{#2}}
6407 \notation{sequence-index}[ui,prec=nobrackets]{\comp{#1}^{#2}}
6408
6409 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{\,\ellipses}}{##1\comp,##2}
6410 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{\,\ellipses,}#2}{##1\comp,##2}
6411 \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{\,\ellipses,}#2\comp{\,\ellipses,}#3}
6412
6413 % letin ("let", local definitions, variable substitution)
6414 \symdecl{letin}[args=bii]
6415 \notation{letin}[let]{\comp{\rm let}}{#1\comp{=}#2\; \comp{\rm in}}{#3}
6416 \notation{letin}[subst]{#3 \comp[ #1 \comp/ #2 \comp]}

```

```

6417 \notation{letin}[frac]{#3 \comp[ \frac{#2}{#1} \comp]}
6418
6419 % structures
6420 \symdecl*{module-type}[args=1]
6421 \notation{module-type}{\comp{\mathtt{MOD}}} #1}
6422 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
6423 \notation{mathstruct}[angle,prec=nobrackets]{\comp\langle #1 \comp\rangle}{##1 \comp, ##2}
6424
6425 % objects
6426 \symdecl{object}
6427 \notation{object}{\comp{\mathtt{OBJECT}}}
6428
6429 }
6430
6431 % The following are abbreviations in the sTeX corpus that are left over from earlier
6432 % developments. They will eventually be phased out.
6433
6434 \ExplSyntaxOn
6435 \stex_add_to_current_module:n{
6436   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
6437   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
6438   \def\livar{\csname sequence-index\endcsname[li]}
6439   \def\uivar{\csname sequence-index\endcsname[ui]}
6440   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
6441   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
6442 }
6443 \__stex_modules_end_module:
6444 \endgroup
6445 \</package>

```

## Chapter 36

# Tikzinput Implementation

```
6446 <@@=tikzinput>
6447 <*package>
6448
6449 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
6450
6451 \ProvidesExplPackage{tikzinput}{2022/02/26}{3.0.1}{tikzinput package}
6452 \RequirePackage{l3keys2e}
6453
6454 \keys_define:nn { tikzinput } {
6455   image .bool_set:N = \c_tikzinput_image_bool,
6456   image .default:n = false ,
6457   unknown .code:n = {}
6458 }
6459
6460 \ProcessKeysOptions { tikzinput }
6461
6462 \bool_if:NTF \c_tikzinput_image_bool {
6463   \RequirePackage{graphicx}
6464
6465   \providecommand\usetikzlibrary[]{}
6466   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
6467 }{
6468   \RequirePackage{tikz}
6469   \RequirePackage{standalone}
6470
6471   \newcommand \tikzinput [2] [] {
6472     \setkeys{Gin}{#1}
6473     \ifx \Gin@ewidth \Gin@exclamation
6474       \ifx \Gin@eheight \Gin@exclamation
6475         \input { #2 }
6476       \else
6477         \resizebox{!}{ \Gin@eheight }{
6478           \input { #2 }
6479         }
6480       \fi
6481     \else
6482       \ifx \Gin@eheight \Gin@exclamation
6483         \resizebox{ \Gin@ewidth }{!}{
```



```

6484         \input { #2 }
6485     }
6486     \else
6487         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
6488             \input { #2 }
6489         }
6490     \fi
6491 \fi
6492 }
6493 }
6494
6495 \newcommand \ctikzinput [2] [] {
6496     \begin{center}
6497         \tikzinput [#1] {#2}
6498     \end{center}
6499 }
6500
6501 \@ifpackageloaded{stex}{
6502     \RequirePackage{stex-tikzinput}
6503 }{}
6504
6505 </package>
6506 <*stex>
6507
6508 \ProvidesExplPackage{stex-tikzinput}{2022/02/26}{3.0.1}{stex-tikzinput}
6509 \RequirePackage{stex}
6510 \RequirePackage{tikzinput}
6511
6512 \newcommand\mhtikzinput[2] []{%
6513     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
6514     \stex_in_repository:nn\Gin@mhrepos{
6515         \tikzinput[#1]{\mhp@hpath{##1}{#2}}
6516     }
6517 }
6518
6519 \newcommand\cmhtikzinput[2] []{\begin{center}\mhtikzinput[#1]{#2}\end{center}}
6520
6521 \cs_new_protected:Nn \__tikzinput_usetikzlibrary:nn {
6522     \pgfkeys@spdef\pgf@temp{#1}
6523     \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
6524     \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgf@temp
6525     \expandafter\edef\csname tikz@library@#1@atcode\endcsname{\the\catcode'\@}
6526     \expandafter\edef\csname tikz@library@#1@barcode\endcsname{\the\catcode'\|}
6527     \expandafter\edef\csname tikz@library@#1@dollarcode\endcsname{\the\catcode'\$}
6528     \catcode'\@=11
6529     \catcode'\|=12
6530     \catcode'\$=3
6531     \pgfutil@InputIfFileExists{#2}{-}{-}
6532     \catcode'\@=\csname tikz@library@#1@atcode\endcsname
6533     \catcode'\|=\csname tikz@library@#1@barcode\endcsname
6534     \catcode'\$=\csname tikz@library@#1@dollarcode\endcsname
6535 }
6536
6537 \newcommand\libusetikzlibrary[1]{

```

```

6537 \prop_if_exist:NF \l_stex_current_repository_prop {
6538   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6539 }
6540 \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
6541   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6542 }
6543 \seq_clear:N \l__tikzinput_libinput_files_seq
6544 \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
6545 \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
6546
6547 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
6548   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / tikzlibrary
6549   \IfFileExists{ \l_tmpa_str }{
6550     \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6551   }{
6552     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
6553     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
6554   }
6555
6556   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / tikzlibrary #1 .code.t
6557   \IfFileExists{ \l_tmpa_str }{
6558     \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6559   }{
6560
6561   \seq_if_empty:NTF \l__tikzinput_libinput_files_seq {
6562     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .code.t
6563   }{
6564     \int_compare:nNnTF {\seq_count:N \l__tikzinput_libinput_files_seq} = 1 {
6565       \seq_map_inline:Nn \l__tikzinput_libinput_files_seq {
6566         \__tikzinput_usetikzlibrary:nn{#1}{ ##1 }
6567       }
6568     }{
6569       \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .co
6570     }
6571   }
6572 }
6573 </stex>

```

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

## Chapter 37

# document-structure.sty Implementation

```
6574 <*package>
6575 <@@=document_structure>
6576 \ProvidesExplPackage{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure}
6577 \RequirePackage{13keys2e}
```

### 37.1 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).

```
6578
6579 \keys_define:nn{ document-structure }{
6580   class      .str_set_x:N = \c_document_structure_class_str,
6581   topsect    .str_set_x:N = \c_document_structure_topsect_str,,
6582   unknown    .code:n      = {
6583     \PassOptionsToClass{\CurrentOption}{stex}
6584     \PassOptionsToClass{\CurrentOption}{tikzinput}
6585   }
6586   % showignores .bool_set:N = \c_document_structure_showignores_bool,
6587 }
6588 \ProcessKeysOptions{ document-structure }
6589 \str_if_empty:NT \c_document_structure_class_str {
6590   \str_set:Nn \c_document_structure_class_str {article}
6591 }
6592 \str_if_empty:NT \c_document_structure_topsect_str {
6593   \str_set:Nn \c_document_structure_topsect_str {section}
6594 }
```

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

```
6595 \RequirePackage{xspace}
6596 \RequirePackage{comment}
6597 \RequirePackage{stex}
6598 \AddToHook{begindocument}{
```

```

6599 \ltx@ifpackageloaded{babel}{
6600     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6601     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6602         \makeatletter\input{document-structure-ngerman.ldf}\makeatother
6603     }
6604 }{}
6605 }

```

`\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.

```

6606 \int_new:N \l_document_structure_section_level_int
6607 \str_case:NnF \c_document_structure_topsect_str {
6608     {part}}{
6609     \int_set:Nn \l_document_structure_section_level_int {0}
6610 }
6611 {chapter}}{
6612     \int_set:Nn \l_document_structure_section_level_int {1}
6613 }
6614 }{
6615     \str_case:NnF \c_document_structure_class_str {
6616         {book}}{
6617         \int_set:Nn \l_document_structure_section_level_int {0}
6618     }
6619     {report}}{
6620         \int_set:Nn \l_document_structure_section_level_int {0}
6621     }
6622 }{
6623     \int_set:Nn \l_document_structure_section_level_int {2}
6624 }
6625 }

```

## 37.2 Document Structure

The structure of the document is given by the `sfragment` environment. The hierarchy is adjusted automatically according to the L<sup>A</sup>T<sub>E</sub>X 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>17</sup>

```

6626 \def\current@section@level{document}%
6627 \newcommand\currentsectionlevel{\lowercase\expandafter\current@section@level\xspace}%
6628 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%

```

(End definition for `\currentsectionlevel`. This function is documented on page 52.)

`\skipfragment`

```

6629 \cs_new_protected:Npn \skipfragment {

```

<sup>17</sup>EdNOTE: MK: we may have to experiment with the more powerful uppercasing macro from `mfirstuc.sty` once we internationalize.

```

6630 \ifcase\l_document_structure_section_level_int
6631 \or\stepcounter{part}
6632 \or\stepcounter{chapter}
6633 \or\stepcounter{section}
6634 \or\stepcounter{subsection}
6635 \or\stepcounter{subsubsection}
6636 \or\stepcounter{paragraph}
6637 \or\stepcounter{subparagraph}
6638 \fi
6639 }

```

(End definition for `\skipfragment`. This function is documented on page 51.)

**blindfragment**

```

6640 \newcommand\at@begin@blindsfragment[1]{
6641 \newenvironment{blindfragment}
6642 {
6643 \int_incr:N\l_document_structure_section_level_int
6644 \at@begin@blindsfragment\l_document_structure_section_level_int
6645 }{}

```

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

```

6646 \newcommand\sfragment@nonum[2]{
6647 \ifx\hyper@anchor\@undefined\else\phantomsection\fi
6648 \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
6649 }

```

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

**\sfragment@num** convenience macro: `\sfragment@num{<level>}{<title>}` makes numbered sectioning with title `<title>` at level `<level>`. We have to check the `short` key was given in the `sfragment` 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.

```

6650 \newcommand\sfragment@num[2]{
6651 \tl_if_empty:NTF \l__document_structure_sfragment_short_tl {
6652 \@nameuse{#1}{#2}
6653 }{
6654 \cs_if_exist:NTF\rdfmata@sectioning{
6655 \@nameuse{rdfmata@#1@old}[\l__document_structure_sfragment_short_tl]{#2}
6656 }{
6657 \@nameuse{#1}[\l__document_structure_sfragment_short_tl]{#2}
6658 }
6659 }
6660 %\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\sfragment@id
6661 }

```

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

**sfragment**

```

6662 \keys_define:nn { document-structure / sfragment }{
6663 id .str_set_x:N = \l__document_structure_sfragment_id_str,
6664 date .str_set_x:N = \l__document_structure_sfragment_date_str,

```

```

6665 creators      .clist_set:N = \l__document_structure_sfragment_creators_clist,
6666 contributors   .clist_set:N = \l__document_structure_sfragment_contributors_clist,
6667 srccite         .tl_set:N    = \l__document_structure_sfragment_srccite_tl,
6668 type           .tl_set:N    = \l__document_structure_sfragment_type_tl,
6669 short          .tl_set:N    = \l__document_structure_sfragment_short_tl,
6670 display        .tl_set:N    = \l__document_structure_sfragment_display_tl,
6671 intro          .tl_set:N    = \l__document_structure_sfragment_intro_tl,
6672 imports        .tl_set:N    = \l__document_structure_sfragment_imports_tl,
6673 loadmodules    .bool_set:N  = \l__document_structure_sfragment_loadmodules_bool
6674 }
6675 \cs_new_protected:Nn \l__document_structure_sfragment_args:n {
6676   \str_clear:N \l__document_structure_sfragment_id_str
6677   \str_clear:N \l__document_structure_sfragment_date_str
6678   \clist_clear:N \l__document_structure_sfragment_creators_clist
6679   \clist_clear:N \l__document_structure_sfragment_contributors_clist
6680   \tl_clear:N \l__document_structure_sfragment_srccite_tl
6681   \tl_clear:N \l__document_structure_sfragment_type_tl
6682   \tl_clear:N \l__document_structure_sfragment_short_tl
6683   \tl_clear:N \l__document_structure_sfragment_display_tl
6684   \tl_clear:N \l__document_structure_sfragment_imports_tl
6685   \tl_clear:N \l__document_structure_sfragment_intro_tl
6686   \bool_set_false:N \l__document_structure_sfragment_loadmodules_bool
6687   \keys_set:nn { document-structure / sfragment } { #1 }
6688 }

```

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

```

6689 \newif\if@mainmatter\@mainmattertrue
6690 \newcommand\at@begin@sfragment[3][]{ }

```

Then we define a helper macro that takes care of the sectioning magic. It comes with its own key/value interface for customization.

```

6691 \keys_define:nn { document-structure / sectioning }{
6692   name      .str_set_x:N = \l__document_structure_sect_name_str   ,
6693   ref       .str_set_x:N = \l__document_structure_sect_ref_str    ,
6694   clear     .bool_set:N  = \l__document_structure_sect_clear_bool ,
6695   clear     .default:n   = {true}                                ,
6696   num       .bool_set:N  = \l__document_structure_sect_num_bool   ,
6697   num       .default:n   = {true}
6698 }
6699 \cs_new_protected:Nn \l__document_structure_sect_args:n {
6700   \str_clear:N \l__document_structure_sect_name_str
6701   \str_clear:N \l__document_structure_sect_ref_str
6702   \bool_set_false:N \l__document_structure_sect_clear_bool
6703   \bool_set_false:N \l__document_structure_sect_num_bool
6704   \keys_set:nn { document-structure / sectioning } { #1 }
6705 }
6706 \newcommand\omdoc@sectioning[3][]{
6707   \l__document_structure_sect_args:n {#1 }
6708   \let\omdoc@sect@name\l__document_structure_sect_name_str
6709   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
6710   \if@mainmatter% numbering not overridden by frontmatter, etc.
6711     \bool_if:NTF \l__document_structure_sect_num_bool {

```

```

6712     \sfragment@num{#2}{#3}
6713   }{
6714     \sfragment@nonum{#2}{#3}
6715   }
6716   \def\current@section@level{\omdoc@sect@name}
6717 \else
6718   \sfragment@nonum{#2}{#3}
6719 \fi
6720 }% 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.

```

6721 \newcommand\sfragment@redefine@addtocontents[1]{%
6722   %\edef\__document_structureimport{#1}%
6723   %\@for\@I:=\__document_structureimport\do{%
6724     %\edef\@path{\csname module@\@I @path\endcsname}%
6725     %\@ifundefined{tf@toc}\relax%
6726     %    {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
6727   %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
6728   %\def\addcontentsline##1##2##3{%
6729     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6730   %\else% hyperref.sty not loaded
6731   %\def\addcontentsline##1##2##3{%
6732     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6733   %\fi
6734   }% hypreref.sty loaded?

```

now the `sfragment` 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 sfragments in the `\sfragment@level` counter.

```

6735 \newenvironment{sfragment}[2] []% keys, title
6736 {
6737   \__document_structure_sfragment_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.

```

6738   \stex_csl_to_imports:No \usemodule \l__document_structure_sfragment_imports_tl
6739
6740   \bool_if:NT \l__document_structure_sfragment_loadmodules_bool {
6741     \sfragment@redefine@addtocontents{
6742       %\@ifundefined{module@id}\used@modules%
6743       %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}
6744     }
6745   }

```

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

```

6746
6747   \stex_document_title:n { #2 }
6748
6749   \int_incr:N\l__document_structure_section_level_int
6750   \ifcase\l__document_structure_section_level_int
6751     \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
6752     \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}

```

```

6753 \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
6754 \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
6755 \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
6756 \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
6757 \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
6758 \fi
6759 \at@begin@sfragment[#1]\l_document_structure_section_level_int{#2}
6760 \str_if_empty:NF \l__document_structure_sfragment_id_str {
6761   \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
6762 }
6763 }% for customization
6764 {}

```

and finally, we localize the sections

```

6765 \newcommand\omdoc@part@kw{Part}
6766 \newcommand\omdoc@chapter@kw{Chapter}
6767 \newcommand\omdoc@section@kw{Section}
6768 \newcommand\omdoc@subsection@kw{Subsection}
6769 \newcommand\omdoc@subsubsection@kw{Subsubsection}
6770 \newcommand\omdoc@paragraph@kw{paragraph}
6771 \newcommand\omdoc@subparagraph@kw{subparagraph}

```

## 37.3 Front and Backmatter

Index markup is provided by the `omtext` package [[Kohlhase:smmtf:git](#)], so in the `document-structure` package we only need to supply the corresponding `\printindex` command, if it is not already defined

`\printindex`

```

6772 \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).

```

6773 \cs_if_exist:NTF\frontmatter{
6774   \let\__document_structure_orig_frontmatter\frontmatter
6775   \let\frontmatter\relax
6776 }{
6777   \tl_set:Nn\__document_structure_orig_frontmatter{
6778     \clearpage
6779     \@mainmatterfalse
6780     \pagenumbering{roman}
6781   }
6782 }
6783 \cs_if_exist:NTF\backmatter{
6784   \let\__document_structure_orig_backmatter\backmatter
6785   \let\backmatter\relax
6786 }{
6787   \tl_set:Nn\__document_structure_orig_backmatter{
6788     \clearpage
6789     \@mainmatterfalse

```



```

6790     \pagenumbering{roman}
6791   }
6792 }

```

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.

```

6793 \newenvironment{frontmatter}{
6794   \_document_structure_orig_frontmatter
6795 }{
6796   \cs_if_exist:NTF\mainmatter{
6797     \mainmatter
6798   }{
6799     \clearpage
6800     \@mainmattertrue
6801     \pagenumbering{arabic}
6802   }
6803 }

```

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

```

6804 \newenvironment{backmatter}{
6805   \_document_structure_orig_backmatter
6806 }{
6807   \cs_if_exist:NTF\mainmatter{
6808     \mainmatter
6809   }{
6810     \clearpage
6811     \@mainmattertrue
6812     \pagenumbering{arabic}
6813   }
6814 }

```

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

```

6815 \@mainmattertrue\pagenumbering{arabic}

```

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

```

6816 \def \c__document_structure_document_str{document}
6817 \newcommand\afterprematurestop{}
6818 \def\prematurestop@endsfragment{
6819   \unless\ifx\@currenvir\c__document_structure_document_str
6820     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
6821     \expandafter\prematurestop@endsfragment
6822   \fi
6823 }
6824 \providecommand\prematurestop{
6825   \message{Stopping~sTeX~processing~prematurely}
6826   \prematurestop@endsfragment
6827   \afterprematurestop
6828   \end{document}
6829 }

```

(End definition for `\prematurestop`. This function is documented on page 52.)

## 37.4 Global Variables

**\setSGvar** set a global variable

```
6830 \RequirePackage{etoolbox}
6831 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}
```

*(End definition for \setSGvar. This function is documented on page 52.)*

**\useSGvar** use a global variable

```
6832 \newrobustcmd\useSGvar[1]{%
6833   \@ifundefined{sTeX@Gvar@#1}
6834   {\PackageError{document-structure}
6835     {The sTeX Global variable #1 is undefined}
6836     {set it with \protect\setSGvar}}
6837   \@nameuse{sTeX@Gvar@#1}}
```

*(End definition for \useSGvar. This function is documented on page 52.)*

**\ifSGvar** execute something conditionally based on the state of the global variable.

```
6838 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
6839   \@ifundefined{sTeX@Gvar@#1}
6840   {\PackageError{document-structure}
6841     {The sTeX Global variable #1 is undefined}
6842     {set it with \protect\setSGvar}}
6843   {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}
```

*(End definition for \ifSGvar. This function is documented on page 52.)*

## Chapter 38

# NotesSlides – Implementation

### 38.1 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.

```
6844 \*cls)
6845 \@@=notesslides)
6846 \ProvidesExplClass{notesslides}{2022/02/28}{3.1.0}{notesslides Class}
6847 \RequirePackage{13keys2e}
6848
6849 \keys_define:nn{notesslides / cls}{
6850   class .str_set_x:N = \c__notesslides_class_str,
6851   notes .bool_set:N = \c__notesslides_notes_bool ,
6852   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
6853   docopt .str_set_x:N = \c__notesslides_docopt_str,
6854   unknown .code:n = {
6855     \PassOptionsToPackage{\CurrentOption}{document-structure}
6856     \PassOptionsToClass{\CurrentOption}{beamer}
6857     \PassOptionsToPackage{\CurrentOption}{notesslides}
6858     \PassOptionsToPackage{\CurrentOption}{stex}
6859   }
6860 }
6861 \ProcessKeysOptions{ notesslides / cls }
6862
6863 \str_if_empty:NF \c__notesslides_class_str {
6864   \PassOptionsToPackage{class=\c__notesslides_class_str}{document-structure}
6865 }
6866
6867 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{book}{
6868   \PassOptionsToPackage{defaulttopsect=part}{notesslides}
6869 }
6870 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{report}{
6871   \PassOptionsToPackage{defaulttopsect=part}{notesslides}
6872 }
6873
6874 \RequirePackage{stex}
```

```

6875 \stex_html_backend:T {
6876   \bool_set_true:N\c__notesslides_notes_bool
6877 }
6878
6879 \bool_if:NTF \c__notesslides_notes_bool {
6880   \PassOptionsToPackage{notes=true}{notesslides}
6881 }{
6882   \PassOptionsToPackage{notes=false}{notesslides}
6883 }
6884 \</cls>

```

now we do the same for the notesslides package.

```

6885 \*package>
6886 \ProvidesExplPackage{notesslides}{2022/02/28}{3.1.0}{notesslides Package}
6887 \RequirePackage{13keys2e}
6888
6889 \keys_define:nn{notesslides / pkg}{
6890   topsect          .str_set_x:N = \c__notesslides_topsect_str,
6891   defaulttopsect   .str_set_x:N = \c__notesslides_defaulttopsec_str,
6892   notes            .bool_set:N  = \c__notesslides_notes_bool ,
6893   slides           .code:n       = { \bool_set_false:N \c__notesslides_notes_bool },
6894   sectocframes     .bool_set:N  = \c__notesslides_sectocframes_bool ,
6895   frameimages      .bool_set:N  = \c__notesslides_frameimages_bool ,
6896   fiboxed          .bool_set:N  = \c__notesslides_fiboxed_bool ,
6897   noproblems       .bool_set:N  = \c__notesslides_noproblems_bool,
6898   unknown          .code:n       = {
6899     \PassOptionsToClass{\CurrentOption}{stex}
6900     \PassOptionsToClass{\CurrentOption}{tikzinput}
6901   }
6902 }
6903 \ProcessKeysOptions{ notesslides / pkg }
6904
6905 \RequirePackage{stex}
6906 \stex_html_backend:T {
6907   \bool_set_true:N\c__notesslides_notes_bool
6908 }
6909
6910 \newif\ifnotes
6911 \bool_if:NTF \c__notesslides_notes_bool {
6912   \notesttrue
6913 }{
6914   \notesfalse
6915 }
6916

```

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

```

6917 \str_if_empty:NTF \c__notesslides_topsect_str {
6918   \str_set_eq:NN \__notesslides_topsect \c__notesslides_defaulttopsec_str
6919 }{
6920   \str_set_eq:NN \__notesslides_topsect \c__notesslides_topsect_str
6921 }
6922 \PassOptionsToPackage{topsect=\__notesslides_topsect}{document-structure}
6923 \</package>

```

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

```

6924 <*cls>
6925 \bool_if:NTF \c__notesslides_notes_bool {
6926   \str_if_empty:NT \c__notesslides_class_str {
6927     \str_set:Nn \c__notesslides_class_str {article}
6928   }
6929   \exp_after:wN\LoadClass\exp_after:wN[\c__notesslides_docostr]
6930   {\c__notesslides_class_str}
6931 }{
6932   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
6933   \newcounter{Item}
6934   \newcounter{paragraph}
6935   \newcounter{subparagraph}
6936   \newcounter{Hfootnote}
6937 }
6938 \RequirePackage{document-structure}

```

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

```

6939 \RequirePackage{notesslides}
6940 </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 `TeX` packages. The first batch of packages we want are loaded on `notesslides.sty`. These are the general ones, we will load the `TeX`-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.

```

6941 <*package>
6942 \bool_if:NT \c__notesslides_notes_bool {
6943   \RequirePackage{a4wide}
6944   \RequirePackage{marginnote}
6945   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
6946   \RequirePackage{mdframed}
6947   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
6948   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
6949 }
6950 \RequirePackage{stex-tikzinput}
6951 \RequirePackage{etoolbox}
6952 \RequirePackage{amssymb}
6953 \RequirePackage{amsmath}
6954 \RequirePackage{comment}
6955 \RequirePackage{textcomp}
6956 \RequirePackage{url}
6957 \RequirePackage{graphicx}
6958 \RequirePackage{pgf}

```

## 38.2 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>18</sup>

```

6959 \bool_if:NT \c__notesslides_notes_bool {
6960   \renewcommand\usetheme[2] [] {\usepackage[#1]{beamernotestheme#2}}
6961 }
6962
6963
6964 \NewDocumentCommand \libusetheme {0{} m} {
6965   \bool_if:NTF \c__notesslides_notes_bool {
6966     \libusepackage[#1]{beamernotestheme#2}
6967   }{
6968     \libusepackage[#1]{beamertheme#2}
6969   }
6970 }

```

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

```

6971 \newcounter{slide}
6972 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
6973 \newlength{\slideheight}\setlength{\slideheight}{9cm}

```

**note** The `note` environment is used to leave out text in the `slides` mode. It does not have a counterpart in OMDoc. So for course notes, we define the `note` environment to be a no-operation otherwise we declare the `note` environment as a comment via the `comment` package.

```

6974 \bool_if:NTF \c__notesslides_notes_bool {
6975   \renewenvironment{note}{\ignorespaces}{}
6976 }{
6977   \excludcomment{note}
6978 }

```

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.

```

6979 \bool_if:NT \c__notesslides_notes_bool {
6980   \newlength{\slideframewidth}
6981   \setlength{\slideframewidth}{1.5pt}

```

**frame** We first define the keys.

```

6982 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
6983   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
6984     \bool_set_true:N #1
6985   }{
6986     \bool_set_false:N #1
6987   }
6988 }
6989 \keys_define:nn{notesslides / frame}{
6990   label .str_set_x:N = \l__notesslides_frame_label_str,
6991   allowframebreaks .code:n = {
6992     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
6993   },
6994   allowdisplaybreaks .code:n = {

```

---

<sup>18</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.

```

6995     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_allowdisplaybreaks\_bool { #1 }
6996 },
6997 fragile .code:n = {
6998     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_fragile\_bool { #1 }
6999 },
7000 shrink .code:n = {
7001     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_shrink\_bool { #1 }
7002 },
7003 squeeze .code:n = {
7004     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_squeeze\_bool { #1 }
7005 },
7006 t .code:n = {
7007     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_t\_bool { #1 }
7008 },
7009 unknown .code:n = {}
7010 }
7011 \cs\_new\_protected:Nn \_notesslides\_frame\_args:n {
7012     \str\_clear:N \l\_notesslides\_frame\_label\_str
7013     \bool\_set\_true:N \l\_notesslides\_frame\_allowframebreaks\_bool
7014     \bool\_set\_true:N \l\_notesslides\_frame\_allowdisplaybreaks\_bool
7015     \bool\_set\_true:N \l\_notesslides\_frame\_fragile\_bool
7016     \bool\_set\_true:N \l\_notesslides\_frame\_shrink\_bool
7017     \bool\_set\_true:N \l\_notesslides\_frame\_squeeze\_bool
7018     \bool\_set\_true:N \l\_notesslides\_frame\_t\_bool
7019     \keys\_set:nn { notesslides / frame }{ #1 }
7020 }

```

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

```

7021 \renewenvironment{frame}[1][]{
7022     \_notesslides\_frame\_args:n{#1}
7023     \sffamily
7024     \stepcounter{slide}
7025     \def\@currentlabel{\theslide}
7026     \str\_if\_empty:NF \l\_notesslides\_frame\_label\_str {
7027         \label{\l\_notesslides\_frame\_label\_str}
7028     }

```

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

```

7029 \def\itemize@level{outer}
7030 \def\itemize@outer{outer}
7031 \def\itemize@inner{inner}
7032 \renewcommand\newpage{\addtocounter{framenum}{1}}
7033 %\newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
7034 \renewenvironment{itemize}{
7035     \ifx\itemize@level\itemize@outer
7036         \def\itemize@label{\$ \rhd \$}
7037     \fi
7038     \ifx\itemize@level\itemize@inner
7039         \def\itemize@label{\$ \scriptstyle \rhd \$}
7040     \fi
7041     \begin{list}
7042     {\itemize@label}
7043     {\setlength{\labelsep}{.3em}
7044     \setlength{\labelwidth}{.5em}
7045     \setlength{\leftmargin}{1.5em}

```

```

7046     }
7047     \edef\itemize@level{\itemize@inner}
7048   }{
7049     \end{list}
7050   }

```

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

```

7051     \stex_html_backend:TF {
7052       \begin{stex_annotate_env}{frame}{}\vbox\bgroup
7053         \mdf@patchamsthm
7054       }{
7055         \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwid
7056       }
7057     }{
7058       \stex_html_backend:TF {
7059         \miko@slidelabel\egroup\end{stex_annotate_env}
7060       }{\medskip\miko@slidelabel\end{mdframed}}
7061     }

```

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

`\frametitle`

```

7062     \renewcommand{\frametitle}[1]{
7063       \stex_document_title:n { #1 }
7064       {\Large\bf\sf\color{blue}{#1}}\medskip
7065     }
7066   }

```

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

EdN:19

`\pause` 19

```

7067     \bool_if:NT \c__notesslides_notes_bool {
7068       \newcommand\pause{}
7069     }

```

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

`nparagraph`

```

7070     \bool_if:NTF \c__notesslides_notes_bool {
7071       \newenvironment{nparagraph}[1] []{\begin{sparagraph}[#1]}\end{sparagraph}}
7072     }{
7073       \excludacomment{nparagraph}
7074     }

```

`nfragment`

```

7075     \bool_if:NTF \c__notesslides_notes_bool {
7076       \newenvironment{nfragment}[2] []{\begin{sfragment}[#1]{#2}}\end{sfragment}}
7077     }{
7078       \excludacomment{nfragment}
7079     }

```

---

<sup>19</sup>EdNOTE: MK: fake it in notes mode for now



ndefinition

```
7080 \bool_if:NTF \c__notesslides_notes_bool {
7081   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}\end{sdefinition}}
7082 }{
7083   \excludecomment{ndefinition}
7084 }
```

nassertion

```
7085 \bool_if:NTF \c__notesslides_notes_bool {
7086   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}\end{sassertion}}
7087 }{
7088   \excludecomment{nassertion}
7089 }
```

nsproof

```
7090 \bool_if:NTF \c__notesslides_notes_bool {
7091   \newenvironment{nproof}[2] [] {\begin{sproof}[#1]{#2}}\end{sproof}}
7092 }{
7093   \excludecomment{nproof}
7094 }
```

nexample

```
7095 \bool_if:NTF \c__notesslides_notes_bool {
7096   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}\end{sexample}}
7097 }{
7098   \excludecomment{nexample}
7099 }
```

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

```
7100 \def\inputref@preskip{\smallskip}
7101 \def\inputref@postskip{\medskip}
```

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

**\inputref\***

```
7102 \let\orig@inputref\inputref
7103 \def\inputref{\@ifstar\ninputref\orig@inputref}
7104 \newcommand\ninputref[2] [] {
7105   \bool_if:NT \c__notesslides_notes_bool {
7106     \orig@inputref[#1]{#2}
7107   }
7108 }
```

(End definition for \inputref\*. This function is documented on page 54.)

## 38.3 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>}`.

```

7109 \newlength{\slidelogoheight}
7110
7111 \bool_if:NTF \c_notesslides_notes_bool {
7112   \setlength{\slidelogoheight}{.4cm}
7113 }{
7114   \setlength{\slidelogoheight}{1cm}
7115 }
7116 \newsavebox{\slidelogo}
7117 \sbox{\slidelogo}{\text{\TeX}}
7118 \newrobustcmd{\setslidelogo}[1]{
7119   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
7120 }

```

(End definition for `\setslidelogo`. This function is documented on page 54.)

**\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.

```

7121 \def\source{Michael Kohlhase}% customize locally
7122 \newrobustcmd{\setsource}[1]{\def\source{#1}}

```

(End definition for `\setsource`. This function is documented on page 54.)

**\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.

```

7123 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
7124 \newsavebox{\cclogo}
7125 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
7126 \newif\ifcchref\cchreffalse
7127 \AtBeginDocument{
7128   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
7129 }
7130 \def\licensing{
7131   \ifcchref
7132     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
7133   \else
7134     {\usebox{\cclogo}}
7135   \fi
7136 }
7137 \newrobustcmd{\setlicensing}[2][]{
7138   \def\@url{#1}
7139   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
7140   \ifx\@url\@empty
7141     \def\licensing{{\usebox{\cclogo}}}
7142   \else
7143     \def\licensing{
7144       \ifcchref
7145         \href{#1}{\usebox{\cclogo}}
7146       \else
7147         {\usebox{\cclogo}}
7148       \fi

```

```

7149     }
7150     \fi
7151 }

```

(End definition for `\setlicensing`. This function is documented on page 54.)

EdN:20

`\slidelabel` Now, we set up the slide label for the article mode.<sup>20</sup>

```

7152 \newrobustcmd\miko@slidelabel{
7153   \vbox to \slidelogoheight{
7154     \vss\hbox to \slidewidth
7155     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
7156   }
7157 }

```

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

## 38.4 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.

```

7158 \def\Gin@mhrepos{}
7159 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7160 \define@key{Gin}{label}{\def\currentlabel{\arabic{slide}}\label{#1}}
7161 \newrobustcmd\frameimage[2][]{
7162   \stepcounter{slide}
7163   \bool_if:NT \c__notesslides_frameimages_bool {
7164     \def\Gin@ewidth{}\setkeys{Gin}{#1}
7165     \bool_if:NF \c__notesslides_notes_bool { \vfill }
7166     \begin{center}
7167       \bool_if:NTF \c__notesslides_fiboxed_bool {
7168         \fbox{
7169           \ifx\Gin@ewidth\@empty
7170             \ifx\Gin@mhrepos\@empty
7171               \mhgraphics[width=\slidewidth,#1]{#2}
7172             \else
7173               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7174             \fi
7175           \else% Gin@ewidth empty
7176             \ifx\Gin@mhrepos\@empty
7177               \mhgraphics[#1]{#2}
7178             \else
7179               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7180             \fi
7181           \fi% Gin@ewidth empty
7182         }
7183       }{
7184         \ifx\Gin@ewidth\@empty
7185           \ifx\Gin@mhrepos\@empty
7186             \mhgraphics[width=\slidewidth,#1]{#2}
7187           \else
7188             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7189           \fi

```

---

<sup>20</sup>EdNOTE: see that we can use the themes for the slides some day. This is all fake.

```

7190         \ifx\Gin@mhrepos\@empty
7191             \mhgraphics[#1]{#2}
7192         \else
7193             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7194         \fi
7195     \fi% Gin@ewidth empty
7196 }
7197 \end{center}
7198 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
7199 \bool_if:NF \c__notesslides_notes_bool { \vfill }
7200 }
7201 } % ifmks@sty@frameimages

```

(End definition for `\frameimage`. This function is documented on page 55.)

## 38.5 Colors and Highlighting

We first specify sans serif fonts as the default.

```

7202 \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

```

7203 \AddToHook{begindocument}{
7204     \definecolor{green}{rgb}{0,.5,0}
7205     \definecolor{purple}{cmyk}{.3,1,0,.17}
7206 }

```

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`.

```

7207 % \def\STpresent#1{\textcolor{blue}{#1}}
7208 \def\defemph#1{\textcolor{magenta}{#1}}
7209 \def\symrefemph#1{\textcolor{cyan}{#1}}
7210 \def\compemph#1{\textcolor{blue}{#1}}
7211 \def\titleemph#1{\textcolor{blue}{#1}}
7212 \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.

```

7213 \pgfdeclareimage[width=.8em]{miko@small@dbend}{stex-dangerous-bend}
7214 \def\smalltextwarning{
7215     \pgfuseimage{miko@small@dbend}
7216     \xspace
7217 }
7218 \pgfdeclareimage[width=1.2em]{miko@dbend}{stex-dangerous-bend}
7219 \newrobustcmd\textwarning{
7220     \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
7221     \xspace
7222 }
7223 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{stex-dangerous-bend}

```

```

7224 \newrobustcmd\bigtextwarning{
7225   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
7226   \xspace
7227 }
(End definition for \textwarning. This function is documented on page 55.)
7228 \newrobustcmd\putgraphicsat[3]{
7229   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
7230 }
7231 \newrobustcmd\putat[2]{
7232   \begin{picture}(0,0)\put(#1){#2}\end{picture}
7233 }

```

## 38.6 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`.

```

7234 \stex_html_backend:F {
7235   \bool_if:NT \c__notesslides_sectocframes_bool {
7236     \str_if_eq:VnTF \__notesslidesstopsect{part}{
7237       \newcounter{chapter}\counterwithin*{section}{chapter}
7238     }{
7239       \str_if_eq:VnT\__notesslidesstopsect{chapter}{
7240         \newcounter{chapter}\counterwithin*{section}{chapter}
7241       }
7242     }
7243   }
7244 }
\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
7245 \def\part@prefix{}
7246 \@ifpackageloaded{document-structure}{}{
7247   \str_case:VnF \__notesslidesstopsect {
7248     {part}{
7249       \int_set:Nn \l_document_structure_section_level_int {0}
7250       \def\thesection{\arabic{chapter}.\arabic{section}}
7251       \def\part@prefix{\arabic{chapter}.}
7252     }
7253     {chapter}{
7254       \int_set:Nn \l_document_structure_section_level_int {1}
7255       \def\thesection{\arabic{chapter}.\arabic{section}}
7256       \def\part@prefix{\arabic{chapter}.}
7257     }
7258   }{
7259     \int_set:Nn \l_document_structure_section_level_int {2}
7260     \def\part@prefix{}
7261   }
7262 }
7263
7264 \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 `sfragment` environment that choses the L<sup>A</sup>T<sub>E</sub>X sectioning macros according to `\section@level`.

`sfragment`

```

7265 \renewenvironment{sfragment}[2][]{
7266   \_document_structure_sfragment_args:n { #1 }
7267   \int_incr:N \l_document_structure_section_level_int
7268   \bool_if:NT \c__notesslides_sectocframes_bool {
7269     \stepcounter{slide}
7270     \begin{frame}[noframenumbering]
7271     \vfill\Large\centering
7272     \red{
7273       \ifcase\l_document_structure_section_level_int\or
7274         \stepcounter{part}
7275         \def\_notesslideslabel{\omdoc@part@kw}~\Roman{part}}
7276         \def\currentsectionlevel{\omdoc@part@kw}
7277       \or
7278         \stepcounter{chapter}
7279         \def\_notesslideslabel{\omdoc@chapter@kw}~\arabic{chapter}}
7280         \def\currentsectionlevel{\omdoc@chapter@kw}
7281       \or
7282         \stepcounter{section}
7283         \def\_notesslideslabel{\part@prefix\arabic{section}}
7284         \def\currentsectionlevel{\omdoc@section@kw}
7285       \or
7286         \stepcounter{subsection}
7287         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
7288         \def\currentsectionlevel{\omdoc@subsection@kw}
7289       \or
7290         \stepcounter{subsubsection}
7291         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
7292         \def\currentsectionlevel{\omdoc@subsubsection@kw}
7293       \or
7294         \stepcounter{paragraph}
7295         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
7296         \def\currentsectionlevel{\omdoc@paragraph@kw}
7297       \else
7298         \def\_notesslideslabel{}
7299         \def\currentsectionlevel{\omdoc@paragraph@kw}
7300       \fi% end ifcase
7301       \_notesslideslabel%\sref@label@id\_notesslideslabel
7302       \quad #2%
7303     }%
7304     \vfill%
7305     \end{frame}%
7306   }
7307   \str_if_empty:NF \l__document_structure_sfragment_id_str {
7308     \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
7309   }
7310 }{}
7311 }
```

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

```

7312 \def\inserttheorembodyfont{\normalfont}
7313 %\bool_if:NF \c_notesslides_notes_bool {
7314 % \defbeamertemplate{theorem begin}{miko}
7315 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
7316 % \ifx\inserttheoremaddition\empty\else\ (\inserttheoremaddition)\fi%
7317 % \inserttheorempunctuation\inserttheorembodyfont\xspace}
7318 % \defbeamertemplate{theorem end}{miko}{}
```

and we set it as the default one.

```

7319 % \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.

```

7320 % \expandafter\def\csname Parent2\endcsname{}
7321 %}
7322
7323 \AddToHook{begindocument}{ % this does not work for some reason
7324 \setbeamertemplate{theorems}[ams style]
7325 }
7326 \bool_if:NT \c_notesslides_notes_bool {
7327 \renewenvironment{columns}[1][]{%
7328 \par\noindent%
7329 \begin{minipage}%
7330 \slidewidth\centering\leavevmode%
7331 }{%
7332 \end{minipage}\par\noindent%
7333 }%
7334 \newsavebox\columnbox%
7335 \renewenvironment<>{column}[2][]{%
7336 \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
7337 }{%
7338 \end{minipage}\end{lrbox}\usebox\columnbox%
7339 }%
7340 }
7341 \bool_if:NTF \c_notesslides_noproblems_bool {
7342 \newenvironment{problems}{}{}
7343 }{
7344 \excludacomment{problems}
7345 }
```

## 38.7 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.

```

7346 \gdef\printexcursions{}
7347 \newcommand\excursionref[2]{% label, text
7348 \bool_if:NT \c_notesslides_notes_bool {
7349 \begin{sparagraph}[title=Excursion]
7350 #2 \sref[fallback=the appendix]{#1}.
7351 \end{sparagraph}}
```

```

7352 }
7353 }
7354 \newcommand\activate@excursion[2][]{
7355   \gappto\printexcursions{\inputref{#1}{#2}}
7356 }
7357 \newcommand\excursion[4][]{% repos, label, path, text
7358   \bool_if:NT \c__notesslides_notes_bool {
7359     \activate@excursion{#1}{#3}\excursionref{#2}{#4}
7360   }
7361 }

```

(End definition for \excursion. This function is documented on page 55.)

### \excursiongroup

```

7362 \keys_define:nn{notesslides / excursiongroup }{
7363   id          .str_set_x:N = \l__notesslides_excursion_id_str,
7364   intro       .tl_set:N   = \l__notesslides_excursion_intro_tl,
7365   mhrepos     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
7366 }
7367 \cs_new_protected:Nn \__notesslides_excursion_args:n {
7368   \tl_clear:N \l__notesslides_excursion_intro_tl
7369   \str_clear:N \l__notesslides_excursion_id_str
7370   \str_clear:N \l__notesslides_excursion_mhrepos_str
7371   \keys_set:nn {notesslides / excursiongroup }{ #1 }
7372 }
7373 \newcommand\excursiongroup[1][]{
7374   \__notesslides_excursion_args:n{ #1 }
7375   \ifdefempty\printexcursions{}% only if there are excursions
7376   {\begin{note}
7377     \begin{sfragment}[#1]{Excursions}%
7378     \ifdefempty\l__notesslides_excursion_intro_tl{\{
7379       \inputref[\l__notesslides_excursion_mhrepos_str]{
7380         \l__notesslides_excursion_intro_tl
7381       }
7382     }
7383     \printexcursions%
7384     \end{sfragment}
7385   }\end{note}}
7386 }
7387 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{\fi
7388 \</package>

```

(End definition for \excursiongroup. This function is documented on page 56.)



## Chapter 39

# The Implementation

### 39.1 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.

```
7389 <*package>
7390 <@@=problems>
7391 \ProvidesExplPackage{problem}{2022/02/26}{3.0.1}{Semantic Markup for Problems}
7392 \RequirePackage{l3keys2e,stex}
7393
7394 \keys_define:nn { problem / pkg }{
7395   notes      .default:n    = { true },
7396   notes      .bool_set:N   = \c__problems_notes_bool,
7397   gnotes     .default:n    = { true },
7398   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
7399   hints      .default:n    = { true },
7400   hints      .bool_set:N   = \c__problems_hints_bool,
7401   solutions  .default:n    = { true },
7402   solutions  .bool_set:N   = \c__problems_solutions_bool,
7403   pts        .default:n    = { true },
7404   pts        .bool_set:N   = \c__problems_pts_bool,
7405   min        .default:n    = { true },
7406   min        .bool_set:N   = \c__problems_min_bool,
7407   boxed      .default:n    = { true },
7408   boxed      .bool_set:N   = \c__problems_boxed_bool,
7409   unknown    .code:n       = {}
7410 }
7411 \newif\ifsolutions
7412
7413 \ProcessKeysOptions{ problem / pkg }
7414 \bool_if:NTF \c__problems_solutions_bool {
7415   \solutionstrue
7416 }{
7417   \solutionsfalse
7418 }
```

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

```
7419 \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.

```
7420 \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.

```
7421 \def\prob@problem@kw{Problem}
7422 \def\prob@solution@kw{Solution}
7423 \def\prob@hint@kw{Hint}
7424 \def\prob@note@kw{Note}
7425 \def\prob@gnote@kw{Grading}
7426 \def\prob@pt@kw{pt}
7427 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
7428 \AddToHook{begindocument}{
7429   \ltx@ifpackageloaded{babel}{
7430     \makeatletter
7431     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7432     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7433       \input{problem-ngerman.ldf}
7434     }
7435     \clist_if_in:NnT \l_tmpa_clist {finnish}{
7436       \input{problem-finnish.ldf}
7437     }
7438     \clist_if_in:NnT \l_tmpa_clist {french}{
7439       \input{problem-french.ldf}
7440     }
7441     \clist_if_in:NnT \l_tmpa_clist {russian}{
7442       \input{problem-russian.ldf}
7443     }
7444     \makeatother
7445   }{}
7446 }
```

## 39.2 Problems and Solutions

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

```
7447 \keys_define:nn{ problem / problem }{
7448   id      .str_set_x:N = \l__problems_prob_id_str,
7449   pts     .tl_set:N    = \l__problems_prob_pts_tl,
7450   min     .tl_set:N    = \l__problems_prob_min_tl,
7451   title   .tl_set:N    = \l__problems_prob_title_tl,
7452   type    .tl_set:N    = \l__problems_prob_type_tl,
7453   imports .tl_set:N    = \l__problems_prob_imports_tl,
7454   name    .str_set_x:N = \l__problems_prob_name_str,
7455   refnum  .int_set:N   = \l__problems_prob_refnum_int
```

```

7456 }
7457 \cs_new_protected:Nn \__problems_prob_args:n {
7458   \str_clear:N \l__problems_prob_id_str
7459   \str_clear:N \l__problems_prob_name_str
7460   \tl_clear:N \l__problems_prob_pts_tl
7461   \tl_clear:N \l__problems_prob_min_tl
7462   \tl_clear:N \l__problems_prob_title_tl
7463   \tl_clear:N \l__problems_prob_type_tl
7464   \tl_clear:N \l__problems_prob_imports_tl
7465   \int_zero_new:N \l__problems_prob_refnum_int
7466   \keys_set:nn { problem / problem }{ #1 }
7467   \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
7468     \let\l__problems_prob_refnum_int\undefined
7469   }
7470 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

7471 \newcounter{problem}[section]
7472 \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.

```

7473 \newcommand\prob@label[1]{\thesection.#1}

```

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

`\prob@number` We consolidate the problem number into a reusable internal macro

```

7474 \newcommand\prob@number{
7475   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
7476     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
7477   }{
7478     \int_if_exist:NTF \l__problems_prob_refnum_int {
7479       \prob@label{\int_use:N \l__problems_prob_refnum_int }
7480     }{
7481       \prob@label\theproblem
7482     }
7483   }
7484 }

```

(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.

```

7485 \newcommand\prob@title[3]{%
7486   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
7487     #2 \l__problems_inclprob_title_tl #3
7488   }{
7489     \tl_if_exist:NTF \l__problems_prob_title_tl {
7490       #2 \l__problems_prob_title_tl #3
7491     }{
7492       #1

```

```

7493     }
7494   }
7495 }

```

(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.

```

7496 \def\prob@heading{
7497   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
7498   %\sref@label@id{\prob@problem@kw~\prob@number}{~}
7499 }

```

(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`

```

7500 \newenvironment{sproblem}[1][{}]{
7501   \__problems_prob_args:n{#1}%\sref@target%
7502   \@in@omtexttrue% we are in a statement (for inline definitions)
7503   \stepcounter{problem}\record@problem
7504   \def\current@section@level{\prob@problem@kw}
7505
7506   \str_if_empty:NT \l__problems_prob_name_str {
7507     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
7508     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
7509     \seq_get_left:NN \l_tmpa_seq \l__problems_prob_name_str
7510   }
7511
7512   \stex_if_do_html:T{
7513     \tl_if_empty:NF \l__problems_prob_title_tl {
7514       \exp_args:No \stex_document_title:n \l__problems_prob_title_tl
7515     }
7516   }
7517
7518   \exp_args:Nno\stex_module_setup:nn{type=problem}\l__problems_prob_name_str
7519
7520   \stex_reactivate_macro:N \STEXexport
7521   \stex_reactivate_macro:N \importmodule
7522   \stex_reactivate_macro:N \symdecl
7523   \stex_reactivate_macro:N \notation
7524   \stex_reactivate_macro:N \symdef
7525
7526   \stex_if_do_html:T{
7527     \begin{stex_annotate_env} {problem} {
7528       \l_stex_module_ns_str ? \l_stex_module_name_str
7529     }
7530
7531     \stex_annotate_invisible:nnn{header}{} {
7532       \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}

```

```

7533     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
7534     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
7535         \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
7536     }
7537 }
7538 }
7539
7540 \stex_csl_to_imports:No \importmodule \l__problems_prob_imports_tl
7541
7542
7543 \tl_if_exist:NTF \l__problems_inclprob_type_tl {
7544     \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
7545 }{
7546     \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
7547 }
7548 \str_if_exist:NTF \l__problems_inclprob_id_str {
7549     \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
7550 }{
7551     \str_set_eq:NN \sproblemid \l__problems_prob_id_str
7552 }
7553
7554
7555 \stex_if_smsmode:F {
7556     \clist_set:No \l_tmpa_clist \sproblemtype
7557     \tl_clear:N \l_tmpa_tl
7558     \clist_map_inline:Nn \l_tmpa_clist {
7559         \tl_if_exist:cT {__problems_sproblem_##1_start:}{
7560             \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_start:}}
7561         }
7562     }
7563     \tl_if_empty:NTF \l_tmpa_tl {
7564         \__problems_sproblem_start:
7565     }{
7566         \l_tmpa_tl
7567     }
7568 }
7569 \stex_ref_new_doc_target:n \sproblemid
7570 \stex_smsmode_do:
7571 }{
7572     \__stex_modules_end_module:
7573     \stex_if_smsmode:F{
7574         \clist_set:No \l_tmpa_clist \sproblemtype
7575         \tl_clear:N \l_tmpa_tl
7576         \clist_map_inline:Nn \l_tmpa_clist {
7577             \tl_if_exist:cT {__problems_sproblem_##1_end:}{
7578                 \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_end:}}
7579             }
7580         }
7581         \tl_if_empty:NTF \l_tmpa_tl {
7582             \__problems_sproblem_end:
7583         }{
7584             \l_tmpa_tl
7585         }
7586     }

```

```

7587 \stex_if_do_html:T{
7588   \end{stex_annotate_env}
7589 }
7590
7591 \smallskip
7592 }
7593
7594 \seq_put_right:Nx\g_stex_smsmode_allowedenvs_seq{\tl_to_str:n{sproblem}}
7595
7596
7597
7598 \cs_new_protected:Nn \__problems_sproblem_start: {
7599   \par\noindent\textbf{\prob@heading\show@pts\show@min\\\ignorespacesandpars
7600 }
7601 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
7602
7603 \newcommand\stexpatchproblem[3][] {
7604   \str_set:Nx \l_tmpa_str{ #1 }
7605   \str_if_empty:NTF \l_tmpa_str {
7606     \tl_set:Nn \__problems_sproblem_start: { #2 }
7607     \tl_set:Nn \__problems_sproblem_end: { #3 }
7608   }{
7609     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
7610     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
7611   }
7612 }
7613
7614
7615 \bool_if:NT \c__problems_boxed_bool {
7616   \surroundwithmdframed{problem}
7617 }

```

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

```

7618 \def\record@problem{
7619   \protected@write\@auxout{}
7620   {
7621     \string\@problem{\prob@number}
7622     {
7623       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7624         \l__problems_inclprob_pts_tl
7625       }{
7626         \l__problems_prob_pts_tl
7627       }
7628     }%
7629     {
7630       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7631         \l__problems_inclprob_min_tl
7632       }{
7633         \l__problems_prob_min_tl
7634       }
7635     }
7636   }
7637 }

```

(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).

```
7638 \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.

```
7639 \keys_define:nn { problem / solution }{
7640   id          .str_set_x:N = \l__problems_solution_id_str ,
7641   for         .tl_set:N   = \l__problems_solution_for_tl ,
7642   height      .dim_set:N  = \l__problems_solution_height_dim ,
7643   creators    .clist_set:N = \l__problems_solution_creators_clist ,
7644   contributors .clist_set:N = \l__problems_solution_contributors_clist ,
7645   srccite     .tl_set:N    = \l__problems_solution_srccite_tl
7646 }
7647 \cs_new_protected:Nn \__problems_solution_args:n {
7648   \str_clear:N \l__problems_solution_id_str
7649   \tl_clear:N \l__problems_solution_for_tl
7650   \tl_clear:N \l__problems_solution_srccite_tl
7651   \clist_clear:N \l__problems_solution_creators_clist
7652   \clist_clear:N \l__problems_solution_contributors_clist
7653   \dim_zero:N \l__problems_solution_height_dim
7654   \keys_set:nn { problem / solution }{ #1 }
7655 }
```

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

```
7656 \newcommand\@startsolution[1][]{
7657   \__problems_solution_args:n { #1 }
7658   \@in@omtexttrue% we are in a statement.
7659   \bool_if:NF \c__problems_boxed_bool { \hrule }
7660   \smallskip\noindent
7661   {\textbf{\prob@solution@kw :}\enspace}
7662   \begin{small}
7663   \def\current@section@level{\prob@solution@kw}
7664   \ignorespacesandpars
7665 }
```

`\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.

```
7666 \box_new:N \l__problems_solution_box
7667 \newenvironment{solution}[1][]{
7668   \stex_html_backend:TF{
7669     \stex_if_do_html:T{
7670       \begin{stex_annotate_env}{solution}{}}
7671   }
7672   }{
7673     \setbox\l__problems_solution_box\vbox\bgroup
7674     \par\smallskip\hrule\smallskip
7675     \noindent\textbf{Solution:}~
7676   }
7677   }{
7678     \stex_html_backend:TF{
```

```

7679 \stex_if_do_html:T{
7680 \end{stex_annotate_env}
7681 }
7682 }{
7683 \smallskip\hrule
7684 \egroup
7685 \bool_if:NT \c__problems_solutions_bool {
7686 \box\l__problems_solution_box
7687 }
7688 }
7689 }
7690
7691 \newcommand\startsolutions{
7692 \bool_set_true:N \c__problems_solutions_bool
7693 % \specialcomment{solution}{\@startsolution}{
7694 % \bool_if:NF \c__problems_boxed_bool {
7695 % \hrule\medskip
7696 % }
7697 % \end{small}}%
7698 % }
7699 % \bool_if:NT \c__problems_boxed_bool {
7700 % \surroundwithmdframed{solution}
7701 % }
7702 }

```

(End definition for \startsolutions. This function is documented on page 57.)

## **\stopsolutions**

```

7703 \newcommand\stopsolutions{\bool_set_false:N \c__problems_solutions_bool}%\excludecomment{sol

```

(End definition for \stopsolutions. This function is documented on page 57.)

so it only remains to start/stop solutions depending on what option was specified.

```

7704 \ifsolutions
7705 \startsolutions
7706 \else
7707 \stopsolutions
7708 \fi

```

## **exnote**

```

7709 \bool_if:NTF \c__problems_notes_bool {
7710 \newenvironment{exnote}[1][{}]{
7711 \par\smallskip\hrule\smallskip
7712 \noindent\textbf{\prob@note@kw :~ }\small
7713 }{
7714 \smallskip\hrule
7715 }
7716 }{
7717 \excludecomment{exnote}
7718 }

```

## **hint**

```

7719 \bool_if:NTF \c__problems_notes_bool {
7720 \newenvironment{hint}[1][{}]{
7721 \par\smallskip\hrule\smallskip

```



```

7722 \noindent\textbf{\prob@hint@kw :~ }\small
7723 }{
7724 \smallskip\hrule
7725 }
7726 \newenvironment{exhint}[1][]{
7727 \par\smallskip\hrule\smallskip
7728 \noindent\textbf{\prob@hint@kw :~ }\small
7729 }{
7730 \smallskip\hrule
7731 }
7732 }{
7733 \excludecomment{hint}
7734 \excludecomment{exhint}
7735 }

```

gnote

```

7736 \bool_if:NTF \c__problems_notes_bool {
7737 \newenvironment{gnote}[1][]{
7738 \par\smallskip\hrule\smallskip
7739 \noindent\textbf{\prob@gnote@kw :~ }\small
7740 }{
7741 \smallskip\hrule
7742 }
7743 }{
7744 \excludecomment{gnote}
7745 }

```

## 39.3 Multiple Choice Blocks

EdN:21

mcb 21

```

7746 \newenvironment{mcb}{
7747 \begin{enumerate}
7748 }{
7749 \end{enumerate}
7750 }

```

we define the keys for the mcb macro

```

7751 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
7752 \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
7753 \bool_set_true:N #1
7754 }{
7755 \bool_set_false:N #1
7756 }
7757 }
7758 \keys_define:nn { problem / mcb }{
7759 id .str_set:x:N = \l__problems_mcc_id_str ,
7760 feedback .tl_set:N = \l__problems_mcc_feedback_tl ,
7761 T .default:n = { false } ,
7762 T .bool_set:N = \l__problems_mcc_t_bool ,
7763 F .default:n = { false } ,
7764 F .bool_set:N = \l__problems_mcc_f_bool ,

```

---

<sup>21</sup>EdNOTE: MK: maybe import something better here from a dedicated MC package

```

7765 Ttext      .tl_set:N      = \l__problems_mcc_Ttext_str ,
7766 Ftext      .tl_set:N      = \l__problems_mcc_Ftext_str
7767 }
7768 \cs_new_protected:Nn \l__problems_mcc_args:n {
7769   \str_clear:N \l__problems_mcc_id_str
7770   \tl_clear:N \l__problems_mcc_feedback_tl
7771   \bool_set_false:N \l__problems_mcc_t_bool
7772   \bool_set_false:N \l__problems_mcc_f_bool
7773   \tl_clear:N \l__problems_mcc_Ttext_tl
7774   \tl_clear:N \l__problems_mcc_Ftext_tl
7775   \str_clear:N \l__problems_mcc_id_str
7776   \keys_set:nn { problem / mcc }{ #1 }
7777 }

```

**\mcc**

```

7778 \def\mccTrueText{\textbf{(true)}~}
7779 \def\mccFalseText{\textbf{(false)}~}
7780 \newcommand\mcc[2][] {
7781   \l__problems_mcc_args:n{ #1 }
7782   \item[{$\Box$}] #2
7783   \ifsolutions
7784     \\\
7785     \bool_if:NT \l__problems_mcc_t_bool {
7786       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccTrueText\l__problems_mcc_Ttext_tl
7787     }
7788     \bool_if:NT \l__problems_mcc_f_bool {
7789       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccFalseText\l__problems_mcc_Ftext_tl
7790     }
7791     \tl_if_empty:NF \l__problems_mcc_feedback_tl {
7792       \emph{(\l__problems_mcc_feedback_tl)}
7793     }
7794   \fi
7795 } %solutions

```

(End definition for \mcc. This function is documented on page 58.)

## 39.4 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.

```

7796
7797 \keys_define:nn{ problem / inclproblem }{
7798   id      .str_set_x:N      = \l__problems_inclprob_id_str,
7799   pts     .tl_set:N         = \l__problems_inclprob_pts_tl,
7800   min     .tl_set:N         = \l__problems_inclprob_min_tl,
7801   title   .tl_set:N         = \l__problems_inclprob_title_tl,
7802   refnum  .int_set:N        = \l__problems_inclprob_refnum_int,
7803   type    .tl_set:N         = \l__problems_inclprob_type_tl,
7804   mhrepos .str_set_x:N      = \l__problems_inclprob_mhrepos_str
7805 }
7806 \cs_new_protected:Nn \__problems_inclprob_args:n {
7807   \str_clear:N \l__problems_prob_id_str

```

```

7808 \tl_clear:N \l__problems_inclprob_pts_tl
7809 \tl_clear:N \l__problems_inclprob_min_tl
7810 \tl_clear:N \l__problems_inclprob_title_tl
7811 \tl_clear:N \l__problems_inclprob_type_tl
7812 \int_zero_new:N \l__problems_inclprob_refnum_int
7813 \str_clear:N \l__problems_inclprob_mhrepos_str
7814 \keys_set:nn { problem / inclproblem }{ #1 }
7815 \tl_if_empty:NT \l__problems_inclprob_pts_tl {
7816   \let\l__problems_inclprob_pts_tl\undefined
7817 }
7818 \tl_if_empty:NT \l__problems_inclprob_min_tl {
7819   \let\l__problems_inclprob_min_tl\undefined
7820 }
7821 \tl_if_empty:NT \l__problems_inclprob_title_tl {
7822   \let\l__problems_inclprob_title_tl\undefined
7823 }
7824 \tl_if_empty:NT \l__problems_inclprob_type_tl {
7825   \let\l__problems_inclprob_type_tl\undefined
7826 }
7827 \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
7828   \let\l__problems_inclprob_refnum_int\undefined
7829 }
7830 }
7831
7832 \cs_new_protected:Nn \__problems_inclprob_clear: {
7833   \let\l__problems_inclprob_id_str\undefined
7834   \let\l__problems_inclprob_pts_tl\undefined
7835   \let\l__problems_inclprob_min_tl\undefined
7836   \let\l__problems_inclprob_title_tl\undefined
7837   \let\l__problems_inclprob_type_tl\undefined
7838   \let\l__problems_inclprob_refnum_int\undefined
7839   \let\l__problems_inclprob_mhrepos_str\undefined
7840 }
7841 \__problems_inclprob_clear:
7842
7843 \newcommand\includeproblem[2][ ]{
7844   \__problems_inclprob_args:n{ #1 }
7845   \exp_args:No \stex_in_repository:nn\l__problems_inclprob_mhrepos_str{
7846     \stex_html_backend:TF {
7847       \str_clear:N \l_tmpa_str
7848       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
7849         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
7850       }
7851       \stex_annotate_invisible:nnn{includeproblem}{
7852         \l_tmpa_str / #2
7853       }{}
7854     }{
7855       \begingroup
7856         \inputreftrue
7857         \tl_if_empty:nTF{ ##1 }{
7858           \input{#2}
7859         }{
7860           \input{ \c_stex_mathhub_str / ##1 / source / #2 }
7861         }

```

```

7862     \endgroup
7863   }
7864 }
7865 \__problems_inclprob_clear:
7866 }

```

(End definition for `\includeproblem`. This function is documented on page 59.)

## 39.5 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.

```

7867 \AddToHook{enddocument}{
7868   \bool_if:NT \c__problems_pts_bool {
7869     \message{Total:~\arabic{pts}~points}
7870   }
7871   \bool_if:NT \c__problems_min_bool {
7872     \message{Total:~\arabic{min}~minutes}
7873   }
7874 }

```

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

```

7875 \def\pts#1{
7876   \bool_if:NT \c__problems_pts_bool {
7877     \marginpar{#1~\prob@pt@kw}
7878   }
7879 }
7880 \def\min#1{
7881   \bool_if:NT \c__problems_min_bool {
7882     \marginpar{#1~\prob@min@kw}
7883   }
7884 }

```

`\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.

```

7885 \newcounter{pts}
7886 \def\show@pts{
7887   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7888     \bool_if:NT \c__problems_pts_bool {
7889       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
7890       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
7891     }
7892   }{
7893     \tl_if_exist:NT \l__problems_prob_pts_tl {
7894       \bool_if:NT \c__problems_pts_bool {
7895         \tl_if_empty:NTF \l__problems_prob_pts_tl{
7896           \tl_set:Nn \l__problems_prob_pts_tl {0}
7897         }
7898         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
7899         \addtocounter{pts}{\l__problems_prob_pts_tl}
7900       }
7901     }

```

```

7902 }
7903 }

```

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

\show@min

```

7904 \newcounter{min}
7905 \def\show@min{
7906   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7907     \bool_if:NT \c__problems_min_bool {
7908       \marginpar{\l__problems_inclprob_pts_tl\ min}
7909       \addtocounter{min}{\l__problems_inclprob_min_tl}
7910     }
7911   }{
7912     \tl_if_exist:NT \l__problems_prob_min_tl {
7913       \bool_if:NT \c__problems_min_bool {
7914         \tl_if_empty:NT\l__problems_prob_min_tl{
7915           \tl_set:Nn \l__problems_prob_min_tl {0}
7916         }
7917         \marginpar{\l__problems_prob_min_tl\ min}
7918         \addtocounter{min}{\l__problems_prob_min_tl}
7919       }
7920     }
7921   }
7922 }
7923 \</package>

```

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

## 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.

```
7924 \*package>
7925 \ProvidesExplPackage{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
7926 \RequirePackage{13keys2e}
7927
7928 \newif\iftest\testfalse
7929 \DeclareOption{test}{\testtrue}
7930 \newif\ifmultiple\multiplefalse
7931 \DeclareOption{multiple}{\multipletrue}
7932 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
7933 \ProcessOptions
```

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

```
7934 \RequirePackage{keyval}[1997/11/10]
7935 \RequirePackage{problem}
```

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

```
7936 \newcommand\hwexam@assignment@kw{Assignment}
7937 \newcommand\hwexam@given@kw{Given}
7938 \newcommand\hwexam@due@kw{Due}
7939 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
7940 blank~for~extra~space}
7941 \def\hwexam@minutes@kw{minutes}
7942 \newcommand\correction@probs@kw{prob.}
7943 \newcommand\correction@pts@kw{total}
7944 \newcommand\correction@reached@kw{reached}
7945 \newcommand\correction@sum@kw{Sum}
7946 \newcommand\correction@grade@kw{grade}
7947 \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

```

7948 \AddToHook{begindocument}{
7949 \ltx@ifpackageloaded{babel}{
7950 \makeatletter
7951 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7952 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7953 \input{hwexam-ngerman.ldf}
7954 }
7955 \clist_if_in:NnT \l_tmpa_clist {finnish}{
7956 \input{hwexam-finnish.ldf}
7957 }
7958 \clist_if_in:NnT \l_tmpa_clist {french}{
7959 \input{hwexam-french.ldf}
7960 }
7961 \clist_if_in:NnT \l_tmpa_clist {russian}{
7962 \input{hwexam-russian.ldf}
7963 }
7964 \makeatother
7965 }{}
7966 }
7967

```

## 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.

```

7968 \newcounter{assignment}
7969 %\numberproblemsin{assignment}

We will prepare the keyval support for the assignment environment.

7970 \keys_define:nn { hwexam / assignment } {
7971 id .str_set:N = \l_@@_assign_id_str,
7972 number .int_set:N = \l_@@_assign_number_int,
7973 title .tl_set:N = \l_@@_assign_title_tl,
7974 type .tl_set:N = \l_@@_assign_type_tl,
7975 given .tl_set:N = \l_@@_assign_given_tl,
7976 due .tl_set:N = \l_@@_assign_due_tl,
7977 loadmodules .code:n = {
7978 \bool_set_true:N \l_@@_assign_loadmodules_bool
7979 }
7980 }
7981 \cs_new_protected:Nn \_@@_assignment_args:n {
7982 \str_clear:N \l_@@_assign_id_str
7983 \int_set:Nn \l_@@_assign_number_int {-1}
7984 \tl_clear:N \l_@@_assign_title_tl
7985 \tl_clear:N \l_@@_assign_type_tl
7986 \tl_clear:N \l_@@_assign_given_tl
7987 \tl_clear:N \l_@@_assign_due_tl
7988 \bool_set_false:N \l_@@_assign_loadmodules_bool
7989 \keys_set:nn { hwexam / assignment }{ #1 }
7990 }

```

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.

```

7991 \newcommand\given@due[2]{
7992 \bool_lazy_all:nF {
7993 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
7994 { \tl_if_empty_p:V \l_@@_assign_given_tl }
7995 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
7996 { \tl_if_empty_p:V \l_@@_assign_due_tl }
7997 }{ #1 }
7998
7999 \tl_if_empty:NTF \l_@@_inclasssign_given_tl {
8000 \tl_if_empty:NF \l_@@_assign_given_tl {
8001 \hwexam@given@kw\xspace\l_@@_assign_given_tl
8002 }
8003 }{
8004 \hwexam@given@kw\xspace\l_@@_inclasssign_given_tl
8005 }
8006
8007 \bool_lazy_or:nnF {
8008 \bool_lazy_and_p:nn {
8009 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
8010 }{
8011 \tl_if_empty_p:V \l_@@_assign_due_tl
8012 }
8013 }{
8014 \bool_lazy_and_p:nn {
8015 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
8016 }{
8017 \tl_if_empty_p:V \l_@@_assign_due_tl
8018 }
8019 }{ ,~ }
8020
8021 \tl_if_empty:NTF \l_@@_inclasssign_due_tl {
8022 \tl_if_empty:NF \l_@@_assign_due_tl {
8023 \hwexam@due@kw\xspace \l_@@_assign_due_tl
8024 }
8025 }{
8026 \hwexam@due@kw\xspace \l_@@_inclasssign_due_tl
8027 }
8028
8029 \bool_lazy_all:nF {
8030 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
8031 { \tl_if_empty_p:V \l_@@_assign_given_tl }
8032 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
8033 { \tl_if_empty_p:V \l_@@_assign_due_tl }
8034 }{ #2 }
8035 }

```

`\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.

```

8036 \newcommand\assignment@title[3]{
8037 \tl_if_empty:NTF \l_@@_inclasssign_title_tl {
8038 \tl_if_empty:NTF \l_@@_assign_title_tl {
8039 #1
8040 }{
8041 #2\l_@@_assign_title_tl#3
8042 }
8043 }{
8044 #2\l_@@_inclasssign_title_tl#3
8045 }
8046 }

```

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

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

```

8047 \newcommand\assignment@number{
8048 \int_compare:nNnTF \l_@@_inclasssign_number_int = {-1} {
8049 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8050 \arabic{assignment}
8051 } {
8052 \int_use:N \l_@@_assign_number_int
8053 }
8054 }{
8055 \int_use:N \l_@@_inclasssign_number_int
8056 }
8057 }

```

(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.

```

8058 \newenvironment{assignment}[1][]{
8059 \_@@_assignment_args:n { #1 }
8060 %\sref@target
8061 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8062 \global\stepcounter{assignment}
8063 }{
8064 \global\setcounter{assignment}{\int_use:N\l_@@_assign_number_int}
8065 }
8066 \setcounter{problem}{0}
8067 \renewcommand\prob@label[1]{\assignment@number.##1}
8068 \def\current@section@level{\document@hwexamtype}
8069 %\sref@label{id}{\document@hwexamtype \thesection}
8070 \begin{@assignment}
8071 ){
8072 \end{@assignment}
8073 }

```

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

```

8074 \def\ass@title{
8075 {\protect\document@hwexamtype}\arabic{assignment}
8076 \assignment@title{}\;{}{}\;} -- \given@due{}\;{}
8077 }
8078 \ifmultiple
8079 \newenvironment{@assignment}{
8080 \bool_if:NTF \l_@@_assign_loadmodules_bool {
8081 \begin{sfragment}[loadmodules]{\ass@title}
8082 }{
8083 \begin{sfragment}{\ass@title}
8084 }
8085 }{
8086 \end{sfragment}
8087 }

```

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

```

8088 \else
8089 \newenvironment{@assignment}{
8090 \begin{center}\bf
8091 \Large@title\strut\
8092 \document@hwexamtype\arabic{assignment}\assignment@title{}\;{}{}\;{}
8093 \large\given@due{--}\;{}{}\;{}
8094 \end{center}
8095 }{}
8096 \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.

```

8097 \keys_define:nn { hwexam / inclassignment } {
8098 %id .str_set_x:N = \l_@@_assign_id_str,
8099 number .int_set:N = \l_@@_inclassign_number_int,
8100 title .tl_set:N = \l_@@_inclassign_title_tl,
8101 type .tl_set:N = \l_@@_inclassign_type_tl,
8102 given .tl_set:N = \l_@@_inclassign_given_tl,
8103 due .tl_set:N = \l_@@_inclassign_due_tl,
8104 mhrepos .str_set_x:N = \l_@@_inclassign_mhrepos_str
8105 }
8106 \cs_new_protected:Nn \_@@_inclassignment_args:n {
8107 \int_set:Nn \l_@@_inclassign_number_int {-1}
8108 \tl_clear:N \l_@@_inclassign_title_tl
8109 \tl_clear:N \l_@@_inclassign_type_tl
8110 \tl_clear:N \l_@@_inclassign_given_tl
8111 \tl_clear:N \l_@@_inclassign_due_tl
8112 \str_clear:N \l_@@_inclassign_mhrepos_str
8113 \keys_set:nn { hwexam / inclassignment }{ #1 }
8114 }
8115 \_@@_inclassignment_args:n {}
8116
8117 \newcommand\inputassignment[2][{}]{

```

```

8118 \_@@_inclassassignment_args:n { #1 }
8119 \str_if_empty:NTF \l_@@_inclasssign_mhrepos_str {
8120 \input{#2}
8121 }{
8122 \stex_in_repository:nn{\l_@@_inclasssign_mhrepos_str}{
8123 \input{\mhpath{\l_@@_inclasssign_mhrepos_str}{#2}}
8124 }
8125 }
8126 \_@@_inclassassignment_args:n {}
8127 }
8128 \newcommand\includeassignment[2][]{
8129 \newpage
8130 \inputassignment[#1]{#2}
8131 }

```

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

## 40.4 Typesetting Exams

\quizheading

```

8132 \ExplSyntaxOff
8133 \newcommand\quizheading[1]{%
8134 \def\@tas{#1}%
8135 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
8136 \ifx\@tas\@empty\else%
8137 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
8138 \fi%
8139 }
8140 \ExplSyntaxOn

```

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

\testheading

```

8141
8142 \def\hwexamheader{\input{hwexam-default.header}}
8143
8144 \def\hwexamminutes{
8145 \tl_if_empty:NTF \testheading@duration {
8146 {\testheading@min}~\hwexam@minutes@kw
8147 }{
8148 \testheading@duration
8149 }
8150 }
8151
8152 \keys_define:nn { hwexam / testheading } {
8153 min .tl_set:N = \testheading@min,
8154 duration .tl_set:N = \testheading@duration,
8155 reqpts .tl_set:N = \testheading@reqpts,
8156 tools .tl_set:N = \testheading@tools
8157 }
8158 \cs_new_protected:Nn \_@@_testheading_args:n {
8159 \tl_clear:N \testheading@min
8160 \tl_clear:N \testheading@duration

```

```

8161 \tl_clear:N \testheading@reqpts
8162 \tl_clear:N \testheading@tools
8163 \keys_set:nn { hwexam / testheading }{ #1 }
8164 }
8165 \newenvironment{testheading}[1][]{
8166 \_@@_testheading_args:n{ #1 }
8167 \newcount\check@time\check@time=\testheading@min
8168 \advance\check@time by -\theassignment@totalmin
8169 \newif\if@bonuspoints
8170 \tl_if_empty:NTF \testheading@reqpts {
8171 \@bonuspointsfalse
8172 }{
8173 \newcount\bonus@pts
8174 \bonus@pts=\theassignment@totalpts
8175 \advance\bonus@pts by -\testheading@reqpts
8176 \edef\bonus@pts{\the\bonus@pts}
8177 \@bonuspointstrue
8178 }
8179 \edef\check@time{\the\check@time}
8180
8181 \makeatletter\hwexamheader\makeatother
8182 }{
8183 \newpage
8184 }

```

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

\testspace

```

8185 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

8186 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

8187 \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.

```

8188 <@=problems>
8189 \renewcommand\@problem[3]{
8190 \stepcounter{assignment@probs}
8191 \def\__problemspts{#2}
8192 \ifx\__problemspts\@empty\else
8193 \addtocounter{assignment@totalpts}{#2}
8194 \fi
8195 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
8196 \xdef\correction@probs{\correction@probs & #1}%
8197 \xdef\correction@pts{\correction@pts & #2}
8198 \xdef\correction@reached{\correction@reached &}

```

```

8199 }
8200 <@@=hwexam>

```

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

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

```

8201 \newcounter{assignment@probs}
8202 \newcounter{assignment@totalpts}
8203 \newcounter{assignment@totalmin}
8204 \def\correction@probs{\correction@probs@kw}
8205 \def\correction@pts{\correction@pts@kw}
8206 \def\correction@reached{\correction@reached@kw}
8207 \stepcounter{assignment@probs}
8208 \newcommand\correction@table{
8209 \resizebox{\textwidth}{!}{%
8210 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
8211 &\multicolumn{\theassignment@probs}{c|}|%|
8212 {\footnotesize\correction@forgrading@kw} &\\ \hline
8213 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
8214 \correction@pts & \theassignment@totalpts & \\ \hline
8215 \correction@reached & & \[.7cm]\hline
8216 \end{tabular}}
8217 \end{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}

```

# Chapter 41

## References

- [Bus+04] Stephen Buswell et al. *The Open Math Standard, Version 2.0*. Tech. rep. The OpenMath Society, 2004. URL: <http://www.openmath.org/standard/om20>.
- [CR99] David Carlisle and Sebastian Rathz. *The graphicx package*. Part of the T<sub>E</sub>X distribution. The Comprehensive T<sub>E</sub>X Archive Network. 1999. URL: <https://www.tug.org/texlive/devsrc/Master/texmf-dist/doc/latex/graphics/graphicx.pdf>.
- [DCM03] The DCMI Usage Board. *DCMI Metadata Terms*. DCMI Recommendation. Dublin Core Metadata Initiative, 2003. URL: <http://dublincore.org/documents/dcmi-terms/>.
- [Koh06] Michael Kohlhase. *OMDoc – An open markup format for mathematical documents [Version 1.2]*. LNAI 4180. Springer Verlag, Aug. 2006. URL: <http://omdoc.org/pubs/omdoc1.2.pdf>.
- [LMH] *LMH Scripts*. URL: <https://github.com/sLaTeX/lmhtools>.
- [MMT] *MMT – Language and System for the Uniform Representation of Knowledge*. Project web site. URL: <https://uniformal.github.io/> (visited on 01/15/2019).
- [MRK18] Dennis Müller, Florian Rabe, and Michael Kohlhase. “Theories as Types”. In: *9th International Joint Conference on Automated Reasoning*. Ed. by Didier Galmiche, Stephan Schulz, and Roberto Sebastiani. Springer Verlag, 2018. URL: <https://kwarc.info/kohlhase/papers/ijcar18-records.pdf>.
- [Rab15] Florian Rabe. “The Future of Logic: Foundation-Independence”. In: *Logica Universalis* 10.1 (2015). 10.1007/s11787-015-0132-x; Winner of the Contest “The Future of Logic” at the World Congress on Universal Logic, pp. 1–20.
- [RK13] Florian Rabe and Michael Kohlhase. “A Scalable Module System”. In: *Information & Computation* 0.230 (2013), pp. 1–54. URL: <https://kwarc.info/frabe/Research/mmt.pdf>.
- [RT] *sLaTeX/RusTeX*. URL: <https://github.com/sLaTeX/RusTeX> (visited on 04/22/2022).

---

<sup>22</sup>EDNOTE: we need an un-numbered version sfragment\*

- [SIa] *sLaTeX/sTeX-IDE*. URL: <https://github.com/slatex/sTeX-IDE> (visited on 04/22/2022).
- [SIb] *sLaTeX/stexls-vscode-plugin*. URL: <https://github.com/slatex/stexls-vscode-plugin> (visited on 04/22/2022).
- [SLS] *sLaTeX/stexls*. URL: <https://github.com/slatex/stexls> (visited on 04/22/2022).
- [ST] *sTeX – An Infrastructure for Semantic Preloading of LaTeX Documents*. URL: <https://ctan.org/pkg/stex> (visited on 04/22/2022).
- [sTeX] *sTeX: A semantic Extension of TeX/LaTeX*. URL: <https://github.com/sLaTeX/sTeX> (visited on 05/11/2020).
- [Tana] Till Tantau. *beamer – A LaTeX class for producing presentations and slides*. URL: <http://ctan.org/pkg/beamer> (visited on 01/07/2014).
- [Tanb] Till Tantau. *User Guide to the Beamer Class*. URL: <http://ctan.org/macros/latex/contrib/beamer/doc/beameruserguide.pdf>.
- [TL] *TeX Live*. URL: <http://www.tug.org/texlive/> (visited on 12/11/2012).