

# The sTeX3 Package \*

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

2022-02-24

## Abstract

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

sTeX augments L<sup>A</sup>T<sub>E</sub>X with

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

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

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

---

\*Version 3.0 (last revised 2022-02-24)

# Contents

<b>I</b>	<b>Manual</b>	<b>1</b>
<b>1</b>	<b>What is sTeX?</b>	<b>2</b>
<b>2</b>	<b>Quickstart</b>	<b>3</b>
2.1	Setup	3
2.1.1	The sTeX IDE	3
2.1.2	Manual Setup	3
2.2	A First sTeX Document	4
<b>3</b>	<b>Using sTeX</b>	<b>6</b>
<b>4</b>	<b>sTeX Archives</b>	<b>7</b>
4.1	The Local MathHub-Directory	7
4.2	The Structure of sTeX Archives	7
4.3	MANIFEST.MF-Files	8
<b>5</b>	<b>Creating New Modules and Symbols</b>	<b>9</b>
5.1	Advanced Structuring Mechanisms	9
5.2	Primitive Symbols (The sTeX Metatheory)	10
<b>6</b>	<b>sTeX Statements (Definitions, Theorems, Examples, ...)</b>	<b>11</b>
<b>7</b>	<b>Additional Packages</b>	<b>12</b>
7.1	Modular Document Structuring	12
7.2	Slides and Course Notes	12
7.3	Homework, Problems and Exams	12
<b>8</b>	<b>Stuff</b>	<b>13</b>
8.1	Modules	13
8.1.1	Semantic Macros and Notations	13
	Other Argument Types	15
	Precedences	17
8.1.2	Archives and Imports	17
	Namespaces	17
	Paths in Import-Statements	18
<b>II</b>	<b>Documentation</b>	<b>19</b>
<b>9</b>	<b>sTeX-Basics</b>	<b>20</b>
9.1	Macros and Environments	20
9.1.1	HTML Annotations	20
9.1.2	Babel Languages	21
9.1.3	Auxiliary Methods	21

<b>10</b>	<b>sTeX-MathHub</b>	<b>22</b>
10.1	Macros and Environments . . . . .	22
10.1.1	Files, Paths, URIs . . . . .	22
10.1.2	MathHub Archives . . . . .	23
10.1.3	Using Content in Archives . . . . .	24
<b>11</b>	<b>sTeX-References</b>	<b>25</b>
11.1	Macros and Environments . . . . .	25
11.1.1	Setting Reference Targets . . . . .	25
11.1.2	Using References . . . . .	26
<b>12</b>	<b>sTeX-Modules</b>	<b>27</b>
12.1	Macros and Environments . . . . .	27
12.1.1	The <code>smodule</code> environment . . . . .	29
<b>13</b>	<b>sTeX-Module Inheritance</b>	<b>31</b>
13.1	Macros and Environments . . . . .	31
13.1.1	SMS Mode . . . . .	31
13.1.2	Imports and Inheritance . . . . .	32
<b>14</b>	<b>sTeX-Symbols</b>	<b>34</b>
14.1	Macros and Environments . . . . .	34
<b>15</b>	<b>sTeX-Terms</b>	<b>36</b>
15.1	Macros and Environments . . . . .	36
<b>16</b>	<b>sTeX-Structural Features</b>	<b>38</b>
16.1	Macros and Environments . . . . .	38
16.1.1	Structures . . . . .	38
<b>17</b>	<b>sTeX-Statements</b>	<b>39</b>
17.1	Macros and Environments . . . . .	39
<b>18</b>	<b>sTeX-Proofs: Structural Markup for Proofs</b>	<b>40</b>
18.1	Introduction . . . . .	42
18.2	The User Interface . . . . .	43
18.2.1	Package Options . . . . .	43
18.2.2	Proofs and Proof steps . . . . .	43
18.2.3	Justifications . . . . .	43
18.2.4	Proof Structure . . . . .	45
18.2.5	Proof End Markers . . . . .	45
18.2.6	Configuration of the Presentation . . . . .	45
18.3	Limitations . . . . .	46
<b>19</b>	<b>sTeX-Metatheory</b>	<b>47</b>
19.1	Symbols . . . . .	47
<b>III</b>	<b>Extensions</b>	<b>48</b>

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

<b>25</b>	<b><code>sTeX</code>-Basics Implementation</b>	<b>69</b>
25.1	The <code>sTeXDocument</code> Class . . . . .	69
25.2	Preliminaries . . . . .	69
25.3	Messages and logging . . . . .	70
25.4	HTML Annotations . . . . .	71
25.5	Babel Languages . . . . .	74
25.6	Auxiliary Methods . . . . .	75
<b>26</b>	<b><code>sTeX</code>-MathHub Implementation</b>	<b>76</b>
26.1	Generic Path Handling . . . . .	76
26.2	PWD and <code>kpsewhich</code> . . . . .	78
26.3	File Hooks and Tracking . . . . .	79
26.4	MathHub Repositories . . . . .	80
26.5	Using Content in Archives . . . . .	84
<b>27</b>	<b><code>sTeX</code>-References Implementation</b>	<b>89</b>
27.1	Document URIs and URLs . . . . .	89
27.2	Setting Reference Targets . . . . .	91
27.3	Using References . . . . .	93
<b>28</b>	<b><code>sTeX</code>-Modules Implementation</b>	<b>96</b>
28.1	The <code>smodule</code> environment . . . . .	100
28.2	Invoking modules . . . . .	105
<b>29</b>	<b><code>sTeX</code>-Module Inheritance Implementation</b>	<b>107</b>
29.1	SMS Mode . . . . .	107
29.2	Inheritance . . . . .	110
<b>30</b>	<b><code>sTeX</code>-Symbols Implementation</b>	<b>115</b>
30.1	Symbol Declarations . . . . .	115
30.2	Notations . . . . .	122
30.3	Variables . . . . .	132
<b>31</b>	<b><code>sTeX</code>-Terms Implementation</b>	<b>136</b>
31.1	Symbol Invocations . . . . .	136
31.2	Terms . . . . .	141
31.3	Notation Components . . . . .	148
31.4	Variables . . . . .	150
<b>32</b>	<b><code>sTeX</code>-Structural Features Implementation</b>	<b>152</b>
32.1	Imports with modification . . . . .	152
32.2	The feature environment . . . . .	159
32.3	Features . . . . .	160
<b>33</b>	<b><code>sTeX</code>-Statements Implementation</b>	<b>166</b>
33.1	Definitions . . . . .	166
33.2	Assertions . . . . .	171
33.3	Examples . . . . .	174
33.4	Logical Paragraphs . . . . .	177

<b>34 The Implementation</b>	<b>181</b>
34.1 Package Options . . . . .	181
34.2 Proofs . . . . .	181
34.3 Justifications . . . . .	192
<b>35 <math>\text{\LaTeX}</math>-Others Implementation</b>	<b>193</b>
<b>36 <math>\text{\LaTeX}</math>-Metatheory Implementation</b>	<b>194</b>
<b>37 Tikzinput Implementation</b>	<b>197</b>
<b>38 document-structure.sty Implementation</b>	<b>199</b>
38.1 The document-structure Class . . . . .	199
38.2 Class Options . . . . .	199
38.3 Beefing up the <code>document</code> environment . . . . .	200
38.4 Implementation: document-structure Package . . . . .	200
38.5 Package Options . . . . .	200
38.6 Document Structure . . . . .	202
38.7 Front and Backmatter . . . . .	205
38.8 Global Variables . . . . .	207
<b>39 NotesSlides – Implementation</b>	<b>208</b>
39.1 Class and Package Options . . . . .	208
39.2 Notes and Slides . . . . .	210
39.3 Header and Footer Lines . . . . .	214
39.4 Frame Images . . . . .	215
39.5 Colors and Highlighting . . . . .	216
39.6 Sectioning . . . . .	217
39.7 Excursions . . . . .	219
<b>40 The Implementation</b>	<b>221</b>
40.1 Package Options . . . . .	221
40.2 Problems and Solutions . . . . .	222
40.3 Multiple Choice Blocks . . . . .	228
40.4 Including Problems . . . . .	229
40.5 Reporting Metadata . . . . .	230
<b>41 Implementation: The hwexam Class</b>	<b>232</b>
41.1 Class Options . . . . .	232
<b>42 Implementation: The hwexam Package</b>	<b>234</b>
42.1 Package Options . . . . .	234
42.2 Assignments . . . . .	235
42.3 Including Assignments . . . . .	238
42.4 Typesetting Exams . . . . .	239
42.5 Leftovers . . . . .	241

**Part I**  
**Manual**

# Chapter 1

## What is sTeX?

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

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

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

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

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

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



# Chapter 2

## Quickstart

### 2.1 Setup

#### 2.1.1 The sTeX IDE

TODO: VSCode Plugin

#### 2.1.2 Manual Setup

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

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

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

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

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

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

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

---

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

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

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

## 2.2 A First $\text{\texttt{sTeX}}$ Document

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

The document we will consider is the following:

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

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

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

Compiling this document with `pdflatex` should yield the output

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

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

### \usemodule

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

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

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

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

---

`\symref`  
`\symname`

---

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

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

---

`\importmodule`

---

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

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

## Chapter 3

# Using $\text{\LaTeX}$

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

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

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

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

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

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

TODO

## Chapter 4

# TeX Archives

### 4.1 The Local MathHub-Directory

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

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

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

### 4.2 The Structure of TeX Archives

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

Each such archive needs two subdirectories:

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

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

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

### 4.3 MANIFEST.MF-Files

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

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

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

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

`source-base` or

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

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

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

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

## Chapter 5

# Creating New Modules and Symbols

TODO

### Example 1

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

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

## 5.1 Advanced Structuring Mechanisms

Given modules:

### Example 2

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

Module 2:  
Module 3:  
Module 4:

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

### Example 3

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

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

TODO: explain donotclone

### Example 4

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

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

Module 6:

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



## Chapter 6

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

## Chapter 7

# Additional Packages

7.1 Modular Document Structuring

7.2 Slides and Course Notes

7.3 Homework, Problems and Exams

# Chapter 8

# Stuff

## 8.1 Modules

---

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

---

### 8.1.1 Semantic Macros and Notations

Semantic macros invoke a formally declared symbol.

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

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

#### Example 5

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

$ab$

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

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

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

#### Example 6

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

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

.

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

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

#### Example 7

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

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

.

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

#### Example 8

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

Multiplying again by  $b$  yields...

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

#### Example 9

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

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

.

<sup>4</sup>EdNOTE: TODO

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

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

### Example 10

```
\symdef{add}{args=2,op={+}}{#1 \comp+ #2}
The operator  $\textcolor{blue}{+}$  adds two elements, as in  $\textcolor{blue}{a+b}$ .
```

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

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

### Example 11

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

$\textcolor{blue}{\cdot}$  (denoted by  $\textcolor{blue}{\cdot}$ ) is defined by...

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

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

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

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

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

## Other Argument Types

So far, we have stated the arity of a semantic macro directly. This works if we only have “normal” (or more precisely: *i*-type) arguments. To make use of other argument types, instead of providing the arity numerically, we can provide it as a sequence of characters representing the argument types – e.g. instead of writing `args=2`, we can equivalently write `args=ii`, indicating that the macro takes two *i*-type arguments.

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

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

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

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

### Example 12

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

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

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

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

### Example 13

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

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

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

5 6

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

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

## Precedences

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

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

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

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

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

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

For example:

**Module 9:**

### Example 14

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

$a+b\cdot c$  and  $a\cdot(b+c)$

## 8.1.2 Archives and Imports

### Namespaces

Ideally,  $\text{\TeX}$  would use arbitrary URIs for modules, with no forced relationships between the *logical* namespace of a module and the *physical* location of the file declaring the module – like MMT does things.

Unfortunately,  $\text{\TeX}$  only provides very restricted access to the file system, so we are forced to generate namespaces systematically in such a way that they reflect the physical location of the associated files, so that  $\text{\TeX}$  can resolve them accordingly. Largely, users need not concern themselves with namespaces at all, but for completeness sake, we describe how they are constructed:

- If `\begin{module}{Foo}` occurs in a file `/path/to/file/Foo[.<lang>].tex` which does not belong to an archive, the namespace is `file://path/to/file`.
- If the same statement occurs in a file `/path/to/file/bar[.<lang>].tex`, the namespace is `file://path/to/file/bar`.

In other words: outside of archives, the namespace corresponds to the file URI with the filename dropped iff it is equal to the module name, and ignoring the (optional) language suffix<sup>1</sup>.

If the current file is in an archive, the procedure is the same except that the initial segment of the file path up to the archive's `source`-folder is replaced by the archive's namespace URI.

### Paths in Import-Statements

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

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

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

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

Since this is less compatible with a modular development, using full URIs directly is discouraged.

---

<sup>1</sup>which is internally attached to the module name instead, but a user need not worry about that.



## Part II

# Documentation

# Chapter 9

## sTeX-Basics

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

### 9.1 Macros and Environments

---

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

---

---

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

---

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

#### 9.1.1 HTML Annotations

---

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

---

---

<code>\latexml_if_p: *</code>	L <sup>A</sup> T <sub>E</sub> X3 conditionals for L <sup>A</sup> T <sub>E</sub> X <sub>ML</sub> .
<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>X<sub>ML</sub> 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> <code>⟨content⟩</code> <code>\end{stex_annotate_env}</code> behaves like <code>\stex_annotate:nnn {⟨property⟩} {⟨resource⟩} {⟨content⟩}</code> .
--------------------------------	--

### 9.1.2 Babel Languages

---

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

---

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

### 9.1.3 Auxiliary Methods

---

<code>\stex_deactivate_macro:Nn</code>	<code>\stex_deactivate_macro:Nn⟨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 10

## STEX-MathHub

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

### 10.1 Macros and Environments

---

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

---

#### 10.1.1 Files, Paths, URIs

---

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

---

---

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

---

---

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

---

---

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

---

---

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

---

---

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

---



---

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

---

### 10.1.2 MathHub Archives

---

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

---

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

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

---

<code>\l_stex_current_repository_prop</code>
--

---

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

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

---

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

---

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

---

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

---



---

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

---

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

### 10.1.3 Using Content in Archives

<hr/> <hr/> <code>\mhp</code> <hr/>	<code>\mhp{<i>archive-ID</i>}{<i>filename</i>}</code>
	Expands to the full path of file <i>filename</i> in repository <i>archive-ID</i> . Does not check whether the file or the repository exist.
<hr/> <hr/> <code>\inputref</code> <code>\mhinput</code> <hr/>	<code>\inputref[<i>archive-ID</i>]{<i>filename</i>}</code> Both <code>\input</code> the file <i>filename</i> in archive <i>archive-ID</i> (relative to the <code>source-</code> subdirectory). <code>\mhinput</code> does so directly. <code>\inputref</code> does so within an <code>\begingroup... \endgroup-</code> block, and skips it in <code>html</code> -mode, inserting a <i>reference</i> to the file instead. Both also set <code>\ifinputref</code> to true.
<hr/> <hr/> <code>\addmhbibresource</code> <hr/>	<code>\inputref[<i>archive-ID</i>]{<i>filename</i>}</code> Adds a <code>.bib</code> -file <i>filename</i> in archive <i>archive-ID</i> (relative to the top-directory of the archive!).
<hr/> <hr/> <code>\libinput</code> <hr/>	<code>\libinput{<i>filename</i>}</code> Inputs <i>filename.tex</i> from the <code>lib</code> folders in the current archive and the <code>meta-inf-</code> archive of the current archive group(s) (if existent) in descending order. Throws an error if no file by that name exists in any of the relevant <code>lib</code> -folders.
<hr/> <hr/> <code>\libusepackage</code> <hr/>	<code>\libusepackage[<i>args</i>]{<i>filename</i>}</code> Like <code>\libinput</code> , but looks for <code>.sty</code> -files and calls <code>\usepackage[<i>meta</i>{<i>args</i>}]<i>Arg</i>{<i>filename</i>}</code> instead of <code>\input</code> . Throws an error, if none or more than one suitable package file is found.
<hr/> <hr/> <code>\mhgraphics</code> <code>\cmhgraphics</code> <hr/>	<i>If</i> the <code>graphicx</code> package is loaded, these macros are defined at <code>\begin{document}</code> . <code>\mhgraphics</code> takes the same arguments as <code>\includegraphics</code> , with the additional optional key <code>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. <code>\cmhgraphics</code> additional wraps the image in a <code>center</code> -environment.
<hr/> <hr/> <code>\lstinputmhlisting</code> <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 11

## STEX-References

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

### 11.1 Macros and Environments

---

**\STEXreftitle**

---

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

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

---

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

---

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

---

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

---

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

---

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

---

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

---

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

---

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

#### 11.1.1 Setting Reference Targets

---

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

---

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

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

---

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

---

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

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

### 11.1.2 Using References

---

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

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

---

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

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

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

---

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

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



# Chapter 12

## STEX-Modules

This sub package contains code related to Modules

### 12.1 Macros and Environments

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

---

 **$\backslash\text{c\_stex\_module\_}\langle URI \rangle\_prop$** 

---

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.

---

 **$\backslash\text{c\_stex\_module\_}\langle URI \rangle\_code$** 

---

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

---

 **$\backslash\text{c\_stex\_module\_}\langle URI \rangle\_constants$** 

---

The names of all constants declared in the module

---

 **$\backslash\text{c\_stex\_module\_}\langle URI \rangle\_constants$** 

---

The full URIs of all modules imported in this module

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

---

---

`\stex_modules_current_namespace:`

Computes the current namespace as follows:

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

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

### 12.1.1 The `smodule` environment

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

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

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

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

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

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

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

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

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

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

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

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

---

---

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

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

---

---

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

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

---

---

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

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

---

---

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

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

---

---

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

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

## Chapter 13

# STEX-Module Inheritance

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

### 13.1 Macros and Environments

#### 13.1.1 SMS Mode

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

---

`\g_stex_smsmode_allowedmacros_tl`

---

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

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

---

`\g_stex_smsmode_allowedmacros_escape_tl`

---

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

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

---

`\g_stex_smsmode_allowedenvs_seq`

---

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

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

---

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

---

Tests whether SMS mode is currently active.

---

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

---

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

---

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

---

### 13.1.2 Imports and Inheritance

---

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

---

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

---

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

---

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

---

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

---

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

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

1. If `<archive-ID>` is empty:
  - (a) If `<path>` is empty, then `<name>` must have been declared earlier in the same file and retrievable from `\g_stex_modules_in_file_seq`, or a file with name `<name>.<lang>.tex` must exist in the same folder, containing a module `<name>`. That module should have the same namespace as the current one.
  - (b) If `<path>` is not empty, it must point to the relative path of the containing file as well as the namespace.
2. Otherwise:
  - (a) If `<path>` is empty, then `<name>` must have been declared earlier in the same file and retrievable from `\g_stex_modules_in_file_seq`, or a file with name `<name>.<lang>.tex` must exist in the top `source` folder of the archive, containing a module `<name>`. That module should lie directly in the namespace of the archive.
  - (b) If `<path>` is not empty, it must point to the path of the containing file as well as the namespace, relative to the namespace of the archive. If a module by that namespace exists, it is returned. Otherwise, we call `\stex_require_module:nn` on the `source` directory of the archive to find the file.

---

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

---



---

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

---

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

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

# Chapter 14

## STEX-Symbols

Code related to symbol declarations and notations

### 14.1 Macros and Environments

---

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

# Chapter 15

## STEX-Terms

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

### 15.1 Macros and Environments

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

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

## Chapter 16

# STEX-Structural Features

Code related to structural features

### 16.1 Macros and Environments

#### 16.1.1 Structures

`mathstructure` TODO

## Chapter 17

# sTeX-Statements

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

### 17.1 Macros and Environments

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

## Chapter 18

# sTeX-Proofs: Structural Markup for Proofs

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

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

## Contents

## 18.1 Introduction

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

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

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

Example 1: A very explicit proof, marked up semantically

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

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



## 18.2 The User Interface

### 18.2.1 Package Options

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

### 18.2.2 Proofs and Proof steps

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

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

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

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

### 18.2.3 Justifications

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

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

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

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

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

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

18.2.4 Proof Structure

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

18.2.5 Proof End Markers

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

\sproofend	The <code>sproof</code> package provides the <code>\sproofend</code> macro for this. If a different symbol for the proof end is to be used (e.g. <i>q.e.d</i> ), then this can be obtained by specifying it using the
\sProofEndSymbol	<code>\sProofEndSymbol</code> configuration macro (e.g. by specifying <code>\sProofEndSymbol{q.e.d}</code> ). Some of the proof structuring macros above will insert proof end symbols for subproofs, in most cases, this is desirable to make the proof structure explicit, but sometimes this wastes space (especially, if a proof ends in a case analysis which will supply its own proof end marker). To suppress it locally, just set <code>proofend={}</code> in them or use use <code>\sProofEndSymbol{}</code> .

18.2.6 Configuration of the Presentation

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

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

Figure 1: Configuration Hooks for Semantic Proof Markup

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

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

EdN:8

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

## 18.3 Limitations

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

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

## Chapter 19

# sTeX-Metatheory

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

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

### 19.1 Symbols

**Part III**  
**Extensions**

## Chapter 20

# Tikzinput

### 20.1 Macros and Environments

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

## Chapter 21

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

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

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

### 21.1 Introduction

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

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

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



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

## 21.2 The User Interface

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

### 21.2.1 Package and Class Options

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

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

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

### 21.2.2 Document Structure

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

```

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

```

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

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

`blindomgroup`

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

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

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

Example 3: A typical Document Structure of a Book

`\skipomgroup`

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

`\currentsectionlevel`  
`\CurrentSectionLevel`

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

---

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

### 21.2.3 Ignoring Inputs

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

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

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

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

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

### 21.2.4 Structure Sharing

`\STRlabel` The `\STRlabel` macro takes two arguments: a label and the content and stores the content for later use by `\STRcopy[⟨URL⟩]{⟨label⟩}`, which expands to the previously stored content. If the `\STRlabel` macro was in a different file, then we can give a URL `⟨URL⟩` that lets  $\text{\LaTeX}$ ML generate the correct reference.

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

### 21.2.5 Global Variables

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

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

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

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

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

### 21.2.6 Colors

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

## 21.3 Limitations

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

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

## Chapter 22

# NotesSlides – Slides and Course Notes

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

### 22.1 Introduction

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

### 22.2 The User Interface

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

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

#### 22.2.1 Package Options

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

- |                     |   |
|---------------------|---|
| <code>slides</code> | • The options <code>slides</code> and <code>notes</code> switch between slides mode and notes mode (see |
| <code>notes</code>  | Section 22.2.2).  |

<code>sectocframes</code>	<ul style="list-style-type: none"> <li>If the option <code>sectocframes</code> is given, then for the <code>omgroups</code>, special frames with the <code>omgroup</code> title (and number) are generated.</li> </ul>
<code>showmeta</code>	<ul style="list-style-type: none"> <li><code>showmeta</code>. If this is set, then the metadata keys are shown (see [Koh20b] for details and customization options).</li> </ul>
<code>frameimages</code> <code>fiboxed</code>	<ul style="list-style-type: none"> <li>If the option <code>frameimages</code> is set, then slide mode also shows the <code>\frameimage</code>-generated frames (see section 22.2.4). If also the <code>fiboxed</code> option is given, the slides are surrounded by a box.</li> </ul>
<code>topsect</code>	<ul style="list-style-type: none"> <li><code>topsect=&lt;sect&gt;</code> can be used to specify the top-level sectioning level; the default for <code>&lt;sect&gt;</code> is <code>section</code>.</li> </ul>

### 22.2.2 Notes and Slides

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

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

```
\ifnotes\maketitle\else
\frame[noframenumbering]\maketitle\fi

\begin{note}
  We start this course with ...
\end{note}

\begin{frame}
  \frametitle{The first slide}
  ...
\end{frame}
\begin{note}
  ... and more explanatory text
\end{note}

\begin{frame}
  \frametitle{The second slide}
  ...
\end{frame}
...
```

Example 4: A typical Course Notes File

By interleaving the `frame` and `note` environments, we can build course notes as shown in Figure 4.

`\ifnotes` Note the use of the `\ifnotes` conditional, which allows different treatment between

<sup>11</sup>EDNOTE: leaving out `noproblems` for the moment until we decide what to do with it.

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

`notes` and `slides` mode – manually setting `\notesttrue` or `\notesfalse` is strongly discouraged however.

⚠: We need to give the title frame the `noframenumbering` option so that the frame numbering is kept in sync between the slides and the course notes.

⚠: The `beamer` class recommends not to use the `allowframebreaks` option on frames (even though it is very convenient). This holds even more in the `notesslides` case: At least in conjunction with `\newpage`, frame numbering behaves funnily (we have tried to fix this, but who knows).

If we want to transclude a the contents of a file as a note, we can use a new variant `\inputref*` of the `\inputref` macro from [KGA20]: `\inputref*{foo}` is equivalent to `\begin{note}\inputref{foo}\end{note}`.

There are some environments that tend to occur at the top-level of `note` environments. We make convenience versions of these: e.g. the `nparagraph` environment is just an `sparagraph` inside a `note` environment (but looks nicer in the source, since it avoids one level of source indenting). Similarly, we have the `nomgroup`, `ndefinition`, `nexample`, `nsproof`, and `nassertion` environments.

### 22.2.3 Header and Footer Lines of the Slides

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

The default footer line of the `notesslides` package mentions copyright and licensing. In the `beamer` class, `\source` stores the author's name as the copyright holder . By default it is *Michael Kohlhase* in the `notesslides` package since he is the main user and designer of this package. `\setsource{<name>}` can change the writer's name. For licensing, we use the Creative Commons Attribution-ShareAlike license by default to strengthen the public domain. If package `hyperref` is loaded, then we can attach a hyperlink to the license logo. `\setlicensing[<url>]{<logo name>}` is used for customization, where `<url>` is optional.

### 22.2.4 Frame Images

Sometimes, we want to integrate slides as images after all – e.g. because we already have a PowerPoint presentation, to which we want to add  $\TeX$ notes. In this case we can use `\frameimage[<opt>]{<path>}`, where `<opt>` are the options of `\includegraphics` from the `graphicx` package [CR99] and `<path>` is the file path (extension can be left off like in `\includegraphics`). We have added the `label` key that allows to give a frame label that can be referenced like a regular `beamer` frame.<sup>12</sup>

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

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

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

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


<sup>12</sup>EdNOTE: MK: the `hyperref` link does not seem to work yet. I wonder why but do not have the time to fix it.

Note that the `\mhframeimage` form is more semantic, which allows more advanced document management features in MathHub.

If `baz/foobar` is the “current module”, i.e. if we are on the MathHub path `...MathHub/fooMH/bar...`, then stating the repository in the first optional argument is redundant, so we can just use

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

## 22.2.5 Colors and Highlighting

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

## 22.2.6 Front Matter, Titles, etc.

### 22.2.7 Excursions

In course notes, we sometimes want to point to an “excursion” – material that is either presupposed or tangential to the course at the moment – e.g. in an appendix. The typical setup is the following:

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

```
\excursion      The \excursion{<ref>}{<path>}{<text>} is syntactic sugar for
\activateexcursion
\begin{nparagraph}[title=Excursion]
  \activateexcursion{founif}{../ex/founif}
  We will cover first-order unification in \sref{founif}.
\end{nparagraph}
```

```
\activateexcursion      where \activateexcursion{<path>} augments the \printexcursions macro by a
\printexcursions        call \inputref{<path>}. In this way, the3 \printexcursions macro (usually in the
                        appendix) will collect up all excursions that are specified in the main text.
```

Sometimes, we want to reference – in an excursion – part of another. We can use

```
\excursionref \excursionref{<label>} for that.
```

Finally, we usually want to put the excursions into an `omgroup` environment and add an introduction, therefore we provide the a variant of the `\printexcursions` macro:

```
\excursiongroup \excursiongroup[id=<id>,intro=<path>] is equivalent to
```

```
\begin{note}
\begin{omgroup}[id=<id>]{Excursions}
  \inputref{<path>}
  \printexcursions
\end{omgroup}
\end{note}
```



### 22.2.8 Miscellaneous

## 22.3 Limitations

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

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

## Chapter 23

# problem.sty: An Infrastructure for formatting Problems

The `problem` package supplies an infrastructure that allows specify problems and to reuse them efficiently in multiple environments.

### 23.1 Introduction

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

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

### 23.2 The User Interface

#### 23.2.1 Package Options

<code>solutions</code>	The <code>problem</code> package takes the options <code>solutions</code> (should solutions be output?), <code>notes</code>
<code>notes</code>	(should the problem notes be presented?), <code>hints</code> (do we give the hints?), <code>gnotes</code> (do we
<code>hints</code>	show grading notes?), <code>pts</code> (do we display the points awarded for solving the problem?),
<code>gnotes</code>	<code>min</code> (do we display the estimated minutes for problem soling). If theses are specified, then
<code>pts</code>	the corresponding auxiliary parts of the problems are output, otherwise, they remain
<code>min</code>	invisible.
<code>boxed</code>	The <code>boxed</code> option specifies that problems should be formatted in framed boxes so
<code>test</code>	that they are more visible in the text. Finally, the <code>test</code> option signifies that we are in
	a test situation, so this option does not show the solutions (of course), but leaves space
	for the students to solve them.
<code>mh</code>	The <code>mh</code> option turns on MathHub support; see [ <code>Kohlhase:mss</code> ].
<code>showmeta</code>	Finally, if the <code>showmeta</code> is set, then the metadata keys are shown (see [ <code>Kohlhase:metakeys</code> ]
	for details and customization options).

---

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

### 23.2.2 Problems and Solutions

**problem** The main environment provided by the **problem** package is (surprise surprise) the **problem** environment. It is used to mark up problems and exercises. The environment takes an optional KeyVal argument with the keys **id** as an identifier that can be reference later, **pts** for the points to be gained from this exercise in homework or quiz situations, **min** for the estimated minutes needed to solve the problem, and finally **title** for an informative title of the problem. For an example of a marked up problem see Figure 5 and the resulting markup see Figure 6.

```
\usepackage[solutions,hints,pts,min]{problem}
\begin{document}
  \begin{sproblem}[id=elephants,pts=10,min=2,title=Fitting Elephants]
    How many Elephants can you fit into a Volkswagen beetle?
  \begin{hint}
    Think positively, this is simple!
  \end{hint}
  \begin{exnote}
    Justify your answer
  \end{exnote}
  \begin{solution}[for=elephants,height=3cm]
    Four, two in the front seats, and two in the back.
  \begin{gnote}
    if they do not give the justification deduct 5 pts
  \end{gnote}
  \end{solution}
  \end{sproblem}
\end{document}
```

Example 5: A marked up Problem

**solution** The **solution** environment can be to specify a solution to a problem. If the **solutions** option is set or **\solutionstrue** is set in the text, then the solution will be presented in the output. The **solution** environment takes an optional KeyVal argument with the keys **id** for an identifier that can be reference **for** to specify which problem this is a solution for, and **height** that allows to specify the amount of space to be left in test situations (i.e. if the **test** option is set in the **\usepackage** statement).

```
Problem 0.1 (Fitting Elephants)
How many Elephants can you fit into a Volkswagen beetle?


---


Hint: Think positively, this is simple!


---


Note:Justify your answer


---


Solution: Four, two in the front seats, and two in the back.


---


```

Example 6: The Formatted Problem from Figure 5

**hint** The **hint** and **exnote** environments can be used in a **problem** environment to give hints and to make notes that elaborate certain aspects of the problem.

**exnote**

**gnote** The **gnote** (grading notes) environment can be used to document situations that

may arise in grading.

Sometimes we would like to locally override the `solutions` option we have given to the package. To turn on solutions we use the `\startsolutions`, to turn them off, `\stopsolutions`. These two can be used at any point in the documents.

Also, sometimes, we want content (e.g. in an exam with master solutions) conditional on whether solutions are shown. This can be done with the `\ifsolutions` conditional.

### 23.2.3 Multiple Choice Blocks

Multiple choice blocks can be formatted using the `mcb` environment, in which single choices are marked up with `\mcc[⟨keyvals⟩]{⟨text⟩}` macro, which takes an optional key/value argument `⟨keyvals⟩` for choice metadata and a required argument `⟨text⟩` for the proposed answer text. The following keys are supported

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

See Figure ?? for an example

### 23.2.4 Including Problems

The `\includeproblem` macro can be used to include a problem from another file. It takes an optional `KeyVal` argument and a second argument which is a path to the file containing the problem (the macro assumes that there is only one problem in the include file). The keys `title`, `min`, and `pts` specify the problem title, the estimated minutes for solving the problem and the points to be gained, and their values (if given) overwrite the ones specified in the `problem` environment in the included file.

### 23.2.5 Reporting Metadata

The sum of the points and estimated minutes (that we specified in the `pts` and `min` keys to the `problem` environment or the `\includeproblem` macro) to the log file and the screen after each run. This is useful in preparing exams, where we want to make sure that the students can indeed solve the problems in an allotted time period.

The `\min` and `\pts` macros allow to specify (i.e. to print to the margin) the distribution of time and reward to parts of a problem, if the `pts` and `pts` package options are set. This allows to give students hints about the estimated time and the points to be awarded.

## 23.3 Limitations

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

1. none reported yet

```

\begin{sproblem}[title=Functions]
  What is the keyword to introduce a function definition in python?
  \begin{mcb}
    \mcc[T]{def}
    \mcc[F,feedback=that is for C and C++){function}
    \mcc[F,feedback=that is for Standard ML]{fun}
    \mcc[F,Ftext=Noooooooooooo,feedback=that is for Java]{public static void}
  \end{mcb}
\end{sproblem}

```

---

**Problem 0.2 (Functions)**

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

1. def
2. function
3. fun
4. public static void

---

**Problem 0.3 (Functions)**

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

1. def  
!
2. function  
that is for C and C++
3. fun  
that is for Standard ML
4. public static void  
that is for Java

Example 7: A Problem with a multiple choice block

## Chapter 24

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

The `hwexam` package and class allows individual course assignment sheets and compound assignment documents using problem files marked up with the `problem` package.

### Contents

## 24.1 Introduction

The `hwexam` package and class supplies an infrastructure that allows to format nice-looking assignment sheets by simply including problems from problem files marked up with the `problem` package [Kohlhase:problem]. It is designed to be compatible with `problems.sty`, and inherits some of the functionality.

## 24.2 The User Interface

### 24.2.1 Package and Class Options

The `hwexam` package and class take the options `solutions`, `notes`, `hints`, `gnotes`, `pts`, `min`, and `boxed` that are just passed on to the `problems` package (cf. its documentation for a description of the intended behavior).

`showmeta` If the `showmeta` option is set, then the metadata keys are shown (see [Kohlhase:metakeys] for details and customization options).

The `hwexam` class additionally accepts the options `report`, `book`, `chapter`, `part`, and `showignores`, of the `omdoc` package [Kohlhase:smomdl] on which it is based and passes them on to that. For the `extrefs` option see [Kohlhase:sref].

### 24.2.2 Assignments

`assignment` This package supplies the `assignment` environment that groups problems into assignment sheets. It takes an optional KeyVal argument with the keys `number` (for the assignment number; if none is given, 1 is assumed as the default or — in multi-assignment documents  
`number` — the ordinal of the `assignment` environment), `title` (for the assignment title; this is referenced in the title of the assignment sheet), `type` (for the assignment type; e.g. “quiz”, or “homework”), `given` (for the date the assignment was given), and `due` (for the date the assignment is due).

### 24.2.3 Typesetting Exams

`multiple` Furthermore, the `hwexam` package takes the option `multiple` that allows to combine multiple assignment sheets into a compound document (the assignment sheets are treated as section, there is a table of contents, etc.).

`test` Finally, there is the option `test` that modifies the behavior to facilitate formatting tests. Only in `test` mode, the macros `\testspace`, `\testnewpage`, and `\testemptypage` have an effect: they generate space for the students to solve the given problems. Thus they can be left in the L<sup>A</sup>T<sub>E</sub>X source.

`\testspace` `\testspace` takes an argument that expands to a dimension, and leaves vertical space accordingly. `\testnewpage` makes a new page in `test` mode, and `\testemptypage` generates an empty page with the cautionary message that this page was intentionally left empty.

`testheading` Finally, the `\testheading` takes an optional keyword argument where the keys  
`duration` `duration` specifies a string that specifies the duration of the test, `min` specifies the equivalent in number of minutes, and `reqpts` the points that are required for a perfect grade.  
`min`  
`reqpts`

### 24.2.4 Including Assignments

`\inputassignment` The `\inputassignment` macro can be used to input an assignment from another file. It takes an optional KeyVal argument and a second argument which is a path to the file containing the problem (the macro assumes that there is only one `assignment` environment in the included file). The keys `number`, `title`, `type`, `given`, and `due` are just as for the `assignment` environment and (if given) overwrite the ones specified in the `assignment` environment in the included file.

## 24.3 Limitations

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

1. none reported yet.



---

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

2022-02-24

Write the solutions to the sheet.

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

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

[illegible]

Example 8: A generated test heading.

Part IV

# Implementation

## Chapter 25

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

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

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

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

### 25.2 Preliminaries

```
15 <*package>
16
17 %%%%%%%%% basics.dtx %%%%%%%%%
18
19 \RequirePackage{expl3,l3keys2e,ltxcmds}
20 \ProvidesExplPackage{stex}{2021/08/01}{1.9}{bla}
21
22 %\RequirePackage{morewrites}
23 %\RequirePackage{amsmath}
24
25 Package options:
26 \keys_define:nn { stex } {
```

```

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

```

**\stex** The  $\text{\TeX}$  logo:

**\sTeX**

```

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

```

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

## 25.3 Messages and logging

```

41 <@@=stex_log>

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

```

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

```

52 \cs_new_protected:Nn \stex_debug:nn {
53   \clist_if_in:NnTF \c_stex_debug_clist { all } {
54     \exp_args:Nnnx\msg_set:nnn{stex}{debug / #1}{
55       \\Debug~#1:~#2\\
56     }
57     \msg_none:nn{stex}{debug / #1}
58   }{
59     \clist_if_in:NnT \c_stex_debug_clist { #1 } {
60       \exp_args:Nnnx\msg_set:nnn{stex}{debug / #1}{
61         \\Debug~#1:~#2\\
62       }
63       \msg_none:nn{stex}{debug / #1}
64     }
65   }
66 }

```

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

Redirecting messages:

```

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

```

## 25.4 HTML Annotations

```

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

```

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

```

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

```

Conditionals for `LATXML`:

`\if@latexml`

```

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

```

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

`\latexml_if_p:`  
`\latexml_if:TF`

```

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

```

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

`\l__stex_annotate_arg_tl`  
`\c__stex_annotate_emptyarg_tl`

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

```

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

```

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

`\_stex_annotate_checkempty:n`

```

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

```

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

`\stex_if_do_html_p:`

Whether to (locally) produce HTML output

`\stex_if_do_html:TF`

```

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

```

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

`\stex_suppress_html:n`

Whether to (locally) produce HTML output

```

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

```

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

`\stex_annotate:nnx`

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

`\stex_annotate_invisible:n`

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

`\stex_annotate_invisible:nnn`

```

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

```

```

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

```

```

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

```

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

## 25.5 Babel Languages

```

213 <@@=stex_language>

```

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

```

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

```



```

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

```

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

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

```

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

```

## 25.6 Auxiliary Methods

`\stex_deactivate_macro:Nn`

```

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

```

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

`\stex_reactivate_macro:N`

```

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

```

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

`\ignorespacesandpars`

```

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

```

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

## Chapter 26

# STEX -MathHub Implementation

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

### 26.1 Generic Path Handling

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

`\stex_path_from_string:Nn`

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

```

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

```

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

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

```

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

```

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

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

```

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

```

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

```

```

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

```

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

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

```

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

```

(End definition for `\stex_path_if_absolute:N $\underline{TF}$` . This function is documented on page 22.)

## 26.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

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

```

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

We determine the PWD

`\c_stex_pwd_seq`  
`\c_stex_pwd_str`

```

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

```

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

## 26.3 File Hooks and Tracking

```

387 <@@=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

```

388 \seq_gclear_new:N\g__stex_files_stack

```

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`  
`\c_stex_mainfile_str`

```

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

```

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

`\g_stex_currentfile_seq`

```

392 \seq_gclear_new:N\g_stex_currentfile_seq

```

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

`\stex_filestack_push:n`

```

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

```



```

444 }
445 \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
446 \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
447 }

```

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

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

```

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

```

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

`\l__stex_mathhub_manifest_file_seq`

```

464 \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`:

```

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

```

```

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

```

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

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

```
500 \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:

```

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

```



```

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

```

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

`\stex_set_current_repository:n`

```

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

```

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

`\stex_require_repository:n`

```

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

```

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

`\l_stex_current_repository_prop`

Current MathHub repository

```

554 %\prop_new:N \l_stex_current_repository_prop
555
556 \_stex_mathhub_find_manifest:N \c_stex_pwd_seq
557 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
558     \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
559 } {
560     \_stex_mathhub_parse_manifest:n { main }
561     \prop_get:Nn \c_stex_mathhub_main_manifest_prop {id}
562     \l_tmpa_str
563     \prop_set_eq:cN { c_stex_mathhub\_l_tmpa_str_manifest_prop }
564     \c_stex_mathhub_main_manifest_prop
565     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
566     \stex_debug:nn{mathhub}{Current~repository:~
567         \prop_item:Nn \l_stex_current_repository_prop {id}
568     }
569 }

```

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

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

```

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

```

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

## 26.5 Using Content in Archives

`\mhpath`

```

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

```

```

615 }
616 }

```

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

`\inputref`  
`\mhinput`

```

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

```

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

`\addmhbibresource`

```

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

```

```

662 }
663 \newcommand\addmhbibresource[2][]{
664   \_stex_mathhub_mhbibresource:nn{ #1 }{ #2 }
665 }

```

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

### \libinput

```

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

```

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

### \libusepackage

```

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

```

```

709
710 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
711   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2.sty}
712   \IfFileExists{ \l_tmpa_str }{
713     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
714   }{}
715   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
716   \seq_put_right:No \l_tmpa_seq \l_tmpa_str
717 }
718
719 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2.sty}
720 \IfFileExists{ \l_tmpa_str }{
721   \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
722 }{}
723
724 \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
725   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
726 }{
727   \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
728     \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
729       \usepackage[#1]{ #1 }
730     }
731   }{
732     \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
733   }
734 }
735 }

```

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

`\mhgraphics`  
`\cmhgraphics`

```

736
737 \AddToHook{begindocument}{
738   \ltx@ifpackageloaded{graphicx}{
739     \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
740     \newcommand\mhgraphics[2][]{\%
741       \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
742       \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}}
743     \newcommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
744   }{}

```

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

`\lstinputmhlisting`  
`\clstinputmhlisting`

```

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

```

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

## Chapter 27

# STEX -References Implementation

```
755 <*package>
756
757 %%%%%%%%%%% references.dtx %%%%%%%%%%%
758
759 <@@=stex_refs>
    Warnings and error messages
760
```

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

```
761 %\iow_new:N \c__stex_refs_refs_iow
762 \AddToHook{begindocument}{
763 % \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
764 }
765 \AddToHook{enddocument}{
766 % \iow_close:N \c__stex_refs_refs_iow
767 }
```

`\STEXreftitle`

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

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

### 27.1 Document URIs and URLs

`\l_stex_current_docns_str`

```
773 \str_new:N \l_stex_current_docns_str
```

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

`\stex_get_document_uri:`

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

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

`\l_stex_current_docurl_str`

```
814 \str_new:N \l_stex_current_docurl_str
```

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

`\stex_get_document_url:`

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



```

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

```

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

## 27.2 Setting Reference Targets

```

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

```

`\stex_ref_new_doc_target:n`

```

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

```

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

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

```

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

```

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

```

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

```

`\stex_ref_new_sym_target:n`

```

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

```

```

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

```

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

## 27.3 Using References

```

926 \str_new:N \l__stex_refs_indocument_str

```

**\sref** Optional arguments:

```

927
928 \keys_define:nn { stex / sref } {
929   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
930   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
931   pre           .tl_set:N = \l__stex_refs_pre_tl ,
932   post          .tl_set:N = \l__stex_refs_post_tl ,
933 }
934 \cs_new_protected:Nn \__stex_refs_args:n {
935   \tl_clear:N \l__stex_refs_linktext_tl
936   \tl_clear:N \l__stex_refs_fallback_tl
937   \tl_clear:N \l__stex_refs_pre_tl
938   \tl_clear:N \l__stex_refs_post_tl
939   \str_clear:N \l__stex_refs_repo_str
940   \keys_set:nn { stex / sref } { #1 }
941 }

```

The actual macro:

```

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

```

```

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

```

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

## \srefsym

```

1004 \NewDocumentCommand \srefsym { 0{} m}{
1005   \stex_get_symbol:n { #2 }
1006   \__stex_refs_sym_aux:nn{##1}{\l_stex_get_symbol_uri_str}
1007 }

```

```

1008
1009 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1010   \str_if_exist:cTF {sref_sym_#2 _label_str }{
1011     \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1012   }{
1013     \__stex_refs_args:n { #1 }
1014     \str_if_empty:NTF \l__stex_refs_indocument_str {
1015       \tl_if_exist:cTF{sref_sym_#2 _type}{
1016         % doc uri in \l_tmpb_str
1017         \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1018         \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1019           % reference
1020           \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1021             \cs_if_exist:cTF{autoref}{
1022               \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1023             }{
1024               \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1025             }
1026           }{
1027             \ltx@ifpackageloaded{hyperref}{
1028               \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1029             }{
1030               \l__stex_refs_linktext_tl
1031             }
1032           }
1033         }{
1034           % URL
1035           \ltx@ifpackageloaded{hyperref}{
1036             \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl \
1037           }{
1038             \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_re
1039           }
1040         }
1041       }{
1042         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_l
1043       }
1044     }{
1045       % TODO
1046     }
1047   }
1048 }

```

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

**\srefsymuri**

```

1049 \cs_new_protected:Npn \srefsymuri #1 #2 {
1050   \__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1051 }

```

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

```

1052 </package>

```

## Chapter 28

# STEX -Modules Implementation

```
1053 <*package>
1054
1055 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1056
1057 <@@=stex_modules>
1058
1059     Warnings and error messages
1058 \msg_new:nnn{stex}{error/unknownmodule}{
1059     No~module~#1~found
1060 }
1061 \msg_new:nnn{stex}{error/syntax}{
1062     Syntax~error:~#1
1063 }
1064 \msg_new:nnn{stex}{error/siglanguage}{
1065     Module~#1~declares~signature~#2,~but~does~not~
1066     declare~its~language
1067 }
1068 \msg_new:nnn{stex}{warning/deprecated}{
1069     #1~is~deprecated;~please~use~#2~instead!
1070 }
1071
1072 \msg_new:nnn{stex}{error/conflictingmodules}{
1073     Conflicting~imports~for~module~#1
1074 }
1075
\l_stex_current_module_str The current module:
1075 \str_new:N \l_stex_current_module_str
1076
(End definition for \l_stex_current_module_str. This variable is documented on page 28.)

\l_stex_all_modules_seq Stores all available modules
1076 \seq_new:N \l_stex_all_modules_seq
1077
(End definition for \l_stex_all_modules_seq. This variable is documented on page 28.)
```

```

\stex_if_in_module_p:
\stex_if_in_module:TF
1077 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1078   \str_if_empty:NTF \l_stex_current_module_str
1079   \prg_return_false: \prg_return_true:
1080 }

```

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

```

\stex_if_module_exists_p:n
\stex_if_module_exists:nTF
1081 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1082   \prop_if_exist:cTF { c_stex_module_#1_prop }
1083   \prg_return_true: \prg_return_false:
1084 }

```

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

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

```

\STEXexport
1085 \cs_new_protected:Nn \stex_add_to_current_module:n {
1086   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1087 }
1088 \cs_new_protected:Npn \STEXexport {
1089   \begingroup
1090   \newlinechar=-1\relax
1091   \endlinechar=-1\relax
1092   %\catcode'\ = 9\relax
1093   \expandafter\endgroup\__stex_modules_export:n
1094 }
1095 \cs_new_protected:Nn \__stex_modules_export:n {
1096   \ignorespaces #1
1097   \stex_add_to_current_module:n { \ignorespaces #1 }
1098   \stex_smsmode_do:
1099 }
1100 \stex_deactivate_macro:Nn \STEXexport {module~environments}

```

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

```

\stex_add_constant_to_current_module:n
1101 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1102   \str_set:Nx \l_tmpa_str { #1 }
1103   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1104 }

```

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

```

\stex_add_import_to_current_module:n
1105 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1106   \str_set:Nx \l_tmpa_str { #1 }
1107   \exp_args:Nno
1108   \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1109     \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1110   }
1111 }

```

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

`\stex_collect_imports:n`

```

1112 \cs_new_protected:Nn \stex_collect_imports:n {
1113   \seq_clear:N \l_stex_collect_imports_seq
1114   \__stex_modules_collect_imports:n {#1}
1115 }
1116 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1117   \seq_map_inline:cn {c_stex_module_#1_imports} {
1118     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1119       \__stex_modules_collect_imports:n { ##1 }
1120     }
1121   }
1122   \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1123     \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1124   }
1125 }

```

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

`\stex_do_up_to_module:n`

```

1126 \int_new:N \l__stex_modules_group_depth_int
1127 \tl_new:N \l__stex_modules_aftergroup_tl
1128 \cs_new_protected:Nn \stex_do_up_to_module:n {
1129   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1130     #1
1131   }{
1132     #1
1133     \expandafter \tl_gset:Nn \expandafter \l__stex_modules_aftergroup_tl \expandafter { \l__
1134       \aftergroup\__stex_modules_aftergroup_do:
1135     }
1136   }
1137   \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1138     \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1139       \l__stex_modules_aftergroup_tl
1140       \tl_clear:N \l__stex_modules_aftergroup_tl
1141     }{
1142       \l__stex_modules_aftergroup_tl
1143       \aftergroup\__stex_modules_aftergroup_do:
1144     }
1145   }

```

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

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

1146

(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.

```

1147 \str_new:N \l_stex_modules_ns_str
1148 \str_new:N \l_stex_modules_subpath_str

```



```

1149 \cs_new_protected:Nn \__stex_modules_compute_namespace:nN {
1150   \str_set:Nx \l_tmpa_str { #1 }
1151   \seq_set_eq:NN \l_tmpa_seq #2
1152   % split off file extension
1153   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1154   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1155   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1156   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1157
1158   \bool_set_true:N \l_tmpa_bool
1159   \bool_while_do:Nn \l_tmpa_bool {
1160     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1161     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1162       {source} { \bool_set_false:N \l_tmpa_bool }
1163     }{}{
1164       \seq_if_empty:NT \l_tmpa_seq {
1165         \bool_set_false:N \l_tmpa_bool
1166       }
1167     }
1168   }
1169
1170   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
1171   \str_if_empty:NTF \l_stex_modules_subpath_str {
1172     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
1173   }{
1174     \str_set:Nx \l_stex_modules_ns_str {
1175       \l_tmpa_str/\l_stex_modules_subpath_str
1176     }
1177   }
1178 }
1179
1180 \cs_new_protected:Nn \stex_modules_current_namespace: {
1181   \str_clear:N \l_stex_modules_subpath_str
1182   \prop_if_exist:NTF \l_stex_current_repository_prop {
1183     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1184     \__stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1185   }{
1186     % split off file extension
1187     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1188     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1189     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1190     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1191     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1192     \str_set:Nx \l_stex_modules_ns_str {
1193       file:\stex_path_to_string:N \l_tmpa_seq
1194     }
1195   }
1196 }

```

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

## 28.1 The smodule environment

smodule arguments:

```

1197 \keys_define:nn { stex / module } {
1198   title      .tl_set:N      = \smodulename ,
1199   type       .str_set_x:N   = \smodulename ,
1200   id         .str_set_x:N   = \smoduleid ,
1201   deprecate  .str_set_x:N   = \l_stex_module_deprecate_str ,
1202   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1203   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1204   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1205   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1206   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1207   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1208   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1209 }
1210
1211 \cs_new_protected:Nn \__stex_modules_args:n {
1212   \str_clear:N \smodulename
1213   \str_clear:N \smodulename
1214   \str_clear:N \smoduleid
1215   \str_clear:N \l_stex_module_ns_str
1216   \str_clear:N \l_stex_module_deprecate_str
1217   \str_clear:N \l_stex_module_lang_str
1218   \str_clear:N \l_stex_module_sig_str
1219   \str_clear:N \l_stex_module_creators_str
1220   \str_clear:N \l_stex_module_contributors_str
1221   \str_clear:N \l_stex_module_meta_str
1222   \str_clear:N \l_stex_module_srccite_str
1223   \keys_set:nn { stex / module } { #1 }
1224 }
1225
1226 % module parameters here? In the body?
1227
```

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

```

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

```

```

1243 \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1244 \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1245 / {\l_stex_module_ns_str}
1246 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1247 \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1248 \str_set:Nx \l_stex_module_ns_str {
1249 \stex_path_to_string:N \l_tmpa_seq
1250 }
1251 }
1252 }
1253 }

```

Next, we determine the language of the module:

```

1254 \str_if_empty:NT \l_stex_module_lang_str {
1255 \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1256 \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1257 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1258 \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1259 \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1260 \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1261 inferred~from~file~name}
1262 \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1263 }
1264 }
1265
1266 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
1267 \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1268 \l_tmpa_str {
1269 \ltx@ifpackageloaded{babel}{
1270 \exp_args:Nx \selectlanguage { \l_tmpa_str }
1271 }{}
1272 } {
1273 \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
1274 }
1275 }}

```

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

```

1276 \str_if_empty:NTF \l_stex_module_sig_str {
1277 \exp_args:Nnx \prop_gset_from_keyval:cn {
1278 c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1279 } {
1280 name = \l_stex_module_name_str ,
1281 ns = \l_stex_module_ns_str ,
1282 file = \exp_not:o { \g_stex_currentfile_seq } ,
1283 lang = \l_stex_module_lang_str ,
1284 sig = \l_stex_module_sig_str ,
1285 deprecate = \l_stex_module_deprecate_str ,
1286 meta = \l_stex_module_meta_str
1287 }
1288 \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1289 \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1290 \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1291 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1292 \str_if_empty:NT \l_stex_module_meta_str {
1293   \str_set:Nx \l_stex_module_meta_str {
1294     \c_stex_metatheory_ns_str ? Metatheory
1295   }
1296 }
1297 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1298   \bool_set_true:N \l_stex_in_meta_bool
1299   \exp_args:Nx \stex_add_to_current_module:n {
1300     \bool_set_true:N \l_stex_in_meta_bool
1301     \stex_activate_module:n {\l_stex_module_meta_str}
1302     \bool_set_false:N \l_stex_in_meta_bool
1303   }
1304   \stex_activate_module:n {\l_stex_module_meta_str}
1305   \bool_set_false:N \l_stex_in_meta_bool
1306 }
1307 }{
1308   \str_if_empty:NT \l_stex_module_lang_str {
1309     \msg_error:nnxx{stex}{error/siglanguage}{
1310       \l_stex_module_ns_str?\l_stex_module_name_str
1311     }{\l_stex_module_sig_str}
1312   }
1313
1314   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1315   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1316   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1317   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1318   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1319   \str_set:Nx \l_tmpa_str {
1320     \stex_path_to_string:N \l_tmpa_seq /
1321     \l_tmpa_str . \l_stex_module_sig_str .tex
1322   }
1323   \IfFileExists \l_tmpa_str {
1324     \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1325       \str_clear:N \l_stex_current_module_str
1326       \seq_clear:N \l_stex_all_modules_seq
1327       \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1328     }
1329   }{
1330     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1331   }
1332   \stex_if_smsmode:F {
1333     \stex_activate_module:n {
1334       \l_stex_module_ns_str ? \l_stex_module_name_str
1335     }
1336   }
1337   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1338 }
1339 \str_if_empty:NF \l_stex_module_deprecate_str {
1340   \msg_warning:nnxx{stex}{warning/deprecated}{
1341     Module~\l_stex_current_module_str
1342   }{
1343     \l_stex_module_deprecate_str
1344   }

```

```

1345 }
1346 \seq_put_right:Nx \l_stex_all_modules_seq {
1347   \l_stex_module_ns_str ? \l_stex_module_name_str
1348 }
1349 }

```

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

**smodule** The module environment.

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

```

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

```

1387 \cs_new_protected:Nn \_stex_modules_end_module: {
1388   \stex_debug:nn{modules}{Closing~module~\prop_item:cn {c_stex_module\_l_stex_current_module}
1389 }

```

(End definition for `\_stex_modules_end_module:.`)

The core environment

```

1390 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1391 \NewDocumentEnvironment { smodule } { 0{} m } {
1392   \stex_module_setup:nn{#1}{#2}
1393   \par
1394   \stex_if_smsmode:F{
1395     \tl_clear:N \l_tmpa_tl
1396     \clist_map_inline:Nn \smodulotype {
1397       \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1398         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1399       }
1400     }
1401     \tl_if_empty:NTF \l_tmpa_tl {
1402       \__stex_modules_smodule_start:
1403     }{
1404       \l_tmpa_tl
1405     }
1406   }
1407   \__stex_modules_begin_module:
1408   \str_if_empty:NF \smoduleid {
1409     \stex_ref_new_doc_target:n \smoduleid
1410   }
1411   \stex_smsmode_do:
1412 } {
1413   \__stex_modules_end_module:
1414   \stex_if_smsmode:F {
1415     \end{stex_annotate_env}
1416     \clist_set:No \l_tmpa_clist \smodulotype
1417     \tl_clear:N \l_tmpa_tl
1418     \clist_map_inline:Nn \l_tmpa_clist {
1419       \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1420         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1421       }
1422     }
1423     \tl_if_empty:NTF \l_tmpa_tl {
1424       \__stex_modules_smodule_end:
1425     }{
1426       \l_tmpa_tl
1427     }
1428   }
1429 }

```

`\stexpatchmodule`

```

1430 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1431 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1432
1433 \newcommand\stexpatchmodule[3] [] {
1434   \str_set:Nx \l_tmpa_str{ #1 }
1435   \str_if_empty:NTF \l_tmpa_str {
1436     \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1437     \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1438   }{

```

```

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

```

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

## 28.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n
1443 \NewDocumentCommand \STEXModule { m } {
1444   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1445   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1446   \tl_set:Nn \l_tmpa_tl {
1447     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1448   }
1449   \seq_map_inline:Nn \l_stex_all_modules_seq {
1450     \str_set:Nn \l_tmpb_str { ##1 }
1451     \str_if_eq:eeT { \l_tmpa_str } {
1452       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1453     } {
1454       \seq_map_break:n {
1455         \tl_set:Nn \l_tmpa_tl {
1456           \stex_invoke_module:n { ##1 }
1457         }
1458       }
1459     }
1460   }
1461   \l_tmpa_tl
1462 }
1463
1464 \cs_new_protected:Nn \stex_invoke_module:n {
1465   \stex_debug:nn{modules}{Invoking~module~#1}
1466   \peek_charcode_remove:NTF ! {
1467     \__stex_modules_invoke_uri:nN { #1 }
1468   } {
1469     \peek_charcode_remove:NTF ? {
1470       \__stex_modules_invoke_symbol:nn { #1 }
1471     } {
1472       \msg_error:nnx{stex}{error/syntax}{
1473         ?~or~!~expected~after~
1474         \c_backslash_str STEXModule{#1}
1475       }
1476     }
1477   }
1478 }
1479
1480 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1481   \str_set:Nn #2 { #1 }
1482 }
1483
1484 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1485   \stex_invoke_symbol:n{#1?#2}

```

```
1486 }
```

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

`\stex_activate_module:n`

```
1487 \bool_new:N \l_stex_in_meta_bool
1488 \bool_set_false:N \l_stex_in_meta_bool
1489 \cs_new_protected:Nn \stex_activate_module:n {
1490   \stex_debug:nn{modules}{Activating~module~#1}
1491   \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1492     \msg_error:nnn{stex}{error/conflictingmodules}{ #1 }
1493   }
1494   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1495     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1496     \use:c{ c_stex_module_#1_code }
1497   }
1498 }
```

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

```
1499 \</package>
```



## Chapter 29

# STEX -Module Inheritance Implementation

```
1500 <*package>
1501
1502 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1503
```

### 29.1 SMS Mode

```
1504 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1505 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1506 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1507 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1508
1509 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1510   \makeatletter
1511   \makeatother
1512   \ExplSyntaxOn
1513   \ExplSyntaxOff
1514   \rustexBREAK
1515 }
1516
1517 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1518   \symdef
1519   \importmodule
1520   \notation
1521   \symdecl
1522   \STEXexport
1523   \inlineass
1524   \inlinedef
1525   \inlineex
1526   \endinput
1527   \setnotation
```

```

1528 \copynotation
1529 }
1530
1531 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1532   \tl_to_str:n {
1533     smodule,
1534     copymodule,
1535     interpretmodule,
1536     sdefinition,
1537     sexample,
1538     sassertion,
1539     sparagraph
1540   }
1541 }

```

(End definition for `\g_stex_smsmode_allowedmacros_tl`, `\g_stex_smsmode_allowedmacros_escape_tl`, and `\g_stex_smsmode_allowedenvs_seq`. These variables are documented on page 31.)

`\stex_if_smsmode_p:`  
`\stex_if_smsmode:TF`

```

1542 \bool_new:N \g__stex_smsmode_bool
1543 \bool_set_false:N \g__stex_smsmode_bool
1544 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1545   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1546 }

```

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

`\_stex_smsmode_in_smsmode:nn`

```

1547 \cs_new_protected:Nn \_stex_smsmode_in_smsmode:nn {
1548   \vbox_set:Nn \l_tmpa_box {
1549     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1550     \bool_gset_true:N \g__stex_smsmode_bool
1551     #2
1552     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1553   }
1554   \box_clear:N \l_tmpa_box
1555 }

```

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

`\stex_file_in_smsmode:nn`

```

1556 \quark_new:N \q__stex_smsmode_break
1557
1558 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1559   \stex_filestack_push:n{#1}
1560   \_stex_smsmode_in_smsmode:nn{#1} {
1561     #2
1562     \everyeof{\q__stex_smsmode_break\noexpand}
1563     \expandafter\expandafter\expandafter
1564     \stex_smsmode_do:
1565     \csname @ @ input\endcsname "#1"\relax
1566   }
1567   \stex_filestack_pop:
1568 }

```

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

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

```

1569 \cs_new_protected:Npn \stex_smsmode_do: {
1570   \stex_if_smsmode:T {
1571     \__stex_smsmode_do:w
1572   }
1573 }
1574 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1575   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
1576     \expandafter\if\expandafter\relax\noexpand#1
1577     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1578   } \else\expandafter\__stex_smsmode_do:w\fi
1579 }{
1580   \__stex_smsmode_do:w % #1
1581 }
1582 }
1583 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1584   \cs_if_eq:NNTF #1 \q__stex_smsmode_break {
1585     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1586       #1\__stex_smsmode_do:w
1587     }{
1588       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1589         #1
1590       }{
1591         \cs_if_eq:NNTF \begin #1 {
1592           \__stex_smsmode_check_begin:n
1593         }{
1594           \cs_if_eq:NNTF \end #1 {
1595             \__stex_smsmode_check_end:n
1596           }{
1597             \__stex_smsmode_do:w
1598           }
1599         }
1600       }
1601     }
1602   }
1603 }
1604
1605 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1606   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1607     \begin{#1}
1608   }{
1609     \__stex_smsmode_do:w
1610   }
1611 }
1612 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1613   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1614     \end{#1}\__stex_smsmode_do:w
1615   }{
1616     \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1617   }
1618 }

```

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

## 29.2 Inheritance

```

1619 <@@=stex_importmodule>

\stex_import_module_uri:nn

1620 \cs_new_protected:Nn \stex_import_module_uri:nn {
1621   \str_set:Nx \l_stex_import_archive_str { #1 }
1622   \str_set:Nn \l_stex_import_path_str { #2 }
1623
1624   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1625   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1626   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1627
1628   \stex_modules_current_namespace:
1629   \bool_lazy_all:nTF {
1630     {\str_if_empty_p:N \l_stex_import_archive_str}
1631     {\str_if_empty_p:N \l_stex_import_path_str}
1632     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1633   }{
1634     \str_set_eq:NN \l_stex_import_path_str \l_stex_modules_subpath_str
1635     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1636   }{
1637     \str_if_empty:NT \l_stex_import_archive_str {
1638       \prop_if_exist:NT \l_stex_current_repository_prop {
1639         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1640       }
1641     }
1642     \str_if_empty:NTF \l_stex_import_archive_str {
1643       \str_if_empty:NF \l_stex_import_path_str {
1644         \str_set:Nx \l_stex_import_ns_str {
1645           \l_stex_module_ns_str / \l_stex_import_path_str
1646         }
1647       }
1648     }{
1649       \stex_require_repository:n \l_stex_import_archive_str
1650       \prop_get:cnN { c_stex_mathhub \l_stex_import_archive_str _manifest_prop } { ns }
1651       \l_stex_import_ns_str
1652       \str_if_empty:NF \l_stex_import_path_str {
1653         \str_set:Nx \l_stex_import_ns_str {
1654           \l_stex_import_ns_str / \l_stex_import_path_str
1655         }
1656       }
1657     }
1658   }
1659 }
```

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

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

```
1663 \str_new:N \l_stex_import_ns_str
```

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

```
\stex_import_require_module:nnnn
    {\ns} {\{archive-ID\}} {\{path\}} {\{name\}}
1664 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1665   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1666
1667     % archive
1668     \str_set:Nx \l_tmpa_str { #2 }
1669     \str_if_empty:NTF \l_tmpa_str {
1670       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1671     } {
1672       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1673       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1674       \seq_put_right:Nn \l_tmpa_seq { source }
1675     }
1676
1677     % path
1678     \str_set:Nx \l_tmpb_str { #3 }
1679     \str_if_empty:NTF \l_tmpb_str {
1680       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1681
1682       \ltx@ifpackageloaded{babel} {
1683         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1684           { \language } \l_tmpb_str {
1685           \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1686         }
1687       } {
1688         \str_clear:N \l_tmpb_str
1689       }
1690
1691       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1692       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1693         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1694       }{
1695         \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1696         \IfFileExists{ \l_tmpa_str.tex }{
1697           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1698         }{
1699           % try english as default
1700           \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1701           \IfFileExists{ \l_tmpa_str.en.tex }{
1702             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1703           }{
1704             \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1705           }
1706         }
1707       }
1708
1709     } {
1710       \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1711       \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1712
```

```

1713 \ltx@ifpackageloaded{babel} {
1714   \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1715   { \language } \l_tmpb_str {
1716     \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1717   }
1718 } {
1719   \str_clear:N \l_tmpb_str
1720 }
1721
1722 \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1723
1724 \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1725 \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1726   \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1727 }{
1728   \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1729   \IfFileExists{ \l_tmpa_str/#4.tex }{
1730     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1731   }{
1732     % try english as default
1733     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1734     \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1735       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1736     }{
1737       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1738       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1739         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1740       }{
1741         \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1742         \IfFileExists{ \l_tmpa_str.tex }{
1743           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1744         }{
1745           % try english as default
1746           \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1747           \IfFileExists{ \l_tmpa_str.en.tex }{
1748             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1749           }{
1750             \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1751           }
1752         }
1753       }
1754     }
1755   }
1756 }
1757 }
1758
1759 \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
1760   \seq_clear:N \l_stex_all_modules_seq
1761   \str_clear:N \l_stex_current_module_str
1762   \str_set:Nx \l_tmpb_str { #2 }
1763   \str_if_empty:NF \l_tmpb_str {
1764     \stex_set_current_repository:n { #2 }
1765   }
1766   \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}

```

```

1767     }
1768
1769     \stex_if_module_exists:nF { #1 ? #4 } {
1770       \msg_error:nnx{stex}{error/unknownmodule}{
1771         #1?#4~(in~file~\g__stex_importmodule_file_str)
1772       }
1773     }
1774   }
1775   \stex_activate_module:n { #1 ? #4 }
1776 }

```

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

### `\importmodule`

```

1777 \NewDocumentCommand \importmodule { 0{ } m } {
1778   \stex_import_module_uri:nn { #1 } { #2 }
1779   \stex_debug:nn{modules}{Importing~module:~
1780     \l_stex_import_ns_str ? \l_stex_import_name_str
1781   }
1782   \stex_if_smsmode:F {
1783     \stex_import_require_module:nnnn
1784     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1785     { \l_stex_import_path_str } { \l_stex_import_name_str }
1786     \stex_annotate_invisible:nnn
1787     {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1788   }
1789   \exp_args:Nx \stex_add_to_current_module:n {
1790     \stex_import_require_module:nnnn
1791     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1792     { \l_stex_import_path_str } { \l_stex_import_name_str }
1793   }
1794   \exp_args:Nx \stex_add_import_to_current_module:n {
1795     \l_stex_import_ns_str ? \l_stex_import_name_str
1796   }
1797   \stex_smsmode_do:
1798   \ignorespacesandpars
1799 }
1800 \stex_deactivate_macro:Nn \importmodule {module-environments}

```

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

### `\usemodule`

```

1801 \NewDocumentCommand \usemodule { 0{ } m } {
1802   \stex_if_smsmode:F {
1803     \stex_import_module_uri:nn { #1 } { #2 }
1804     \stex_import_require_module:nnnn
1805     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1806     { \l_stex_import_path_str } { \l_stex_import_name_str }
1807     \stex_annotate_invisible:nnn
1808     {usemodule} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1809   }
1810   \stex_smsmode_do:
1811   \ignorespacesandpars
1812 }

```

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

1813 `\endpackage`



## Chapter 30

# STEX -Symbols Implementation

```
1814 <*package>
1815
1816 %%%%%%%%%% symbols.dtx %%%%%%%%%%
1817
    Warnings and error messages
1818 \msg_new:nnn{stex}{error/wrongargs}{
1819   args~value~in~symbol~declaration~for~#1~
1820   needs~to~be~i,~a,~b~or~B,~but~#2~given
1821 }
```

### 30.1 Symbol Declarations

```
1822 <@@=stex_symdecl>

\stex_all_symbols:n Map over all available symbols

1823 \cs_new_protected:Nn \stex_all_symbols:n {
1824   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
1825   \seq_map_inline:Nn \l_stex_all_modules_seq {
1826     \seq_map_inline:cn{c_stex_module_##1_constants}{
1827       \__stex_symdecl_all_symbols_cs{##1?###1}
1828     }
1829   }
1830 }

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

\STEXsymbol

1831 \NewDocumentCommand \STEXsymbol { m } {
1832   \stex_get_symbol:n { #1 }
1833   \exp_args:No
1834   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1835 }
```

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

`symdecl` arguments:

```

1836 \keys_define:nn { stex / symdecl } {
1837   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1838   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1839   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1840   type      .tl_set:N = \l_stex_symdecl_type_tl ,
1841   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,
1842   align     .str_set:N = \l_stex_symdecl_align_str , % TODO(?)
1843   gfc       .str_set:N = \l_stex_symdecl_gfc_str , % TODO(?)
1844   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
1845   def       .tl_set:N = \l_stex_symdecl_definiens_tl ,
1846   assoc     .choices:nn =
1847     {bin,binl,binr,pre,conj,pwconj}
1848     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}
1849 }
1850
1851 \bool_new:N \l_stex_symdecl_make_macro_bool
1852
1853 \cs_new_protected:Nn \__stex_symdecl_args:n {
1854   \str_clear:N \l_stex_symdecl_name_str
1855   \str_clear:N \l_stex_symdecl_args_str
1856   \str_clear:N \l_stex_symdecl_deprecate_str
1857   \str_clear:N \l_stex_symdecl_assoctype_str
1858   \bool_set_false:N \l_stex_symdecl_local_bool
1859   \tl_clear:N \l_stex_symdecl_type_tl
1860   \tl_clear:N \l_stex_symdecl_definiens_tl
1861
1862   \keys_set:nn { stex / symdecl } { #1 }
1863 }

```

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

```

1864
1865 \NewDocumentCommand \symdecl { s m O{} } {
1866   \__stex_symdecl_args:n { #3 }
1867   \IfBooleanTF #1 {
1868     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1869   } {
1870     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1871   }
1872   \stex_symdecl_do:n { #2 }
1873   \stex_smsmode_do:
1874 }
1875
1876 \cs_new_protected:Nn \stex_symdecl_do:nn {
1877   \__stex_symdecl_args:n{#1}
1878   \bool_set_false:N \l_stex_symdecl_make_macro_bool
1879   \stex_symdecl_do:n{#2}
1880 }
1881
1882 \stex_deactivate_macro:Nn \symdecl {module~environments}

```

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

`\stex_symdecl_do:n`

```
1883 \cs_new_protected:Nn \stex_symdecl_do:n {
1884   \stex_if_in_module:F {
1885     % TODO throw error? some default namespace?
1886   }
1887
1888   \str_if_empty:NT \l_stex_symdecl_name_str {
1889     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1890   }
1891
1892   \prop_if_exist:cT { l_stex_symdecl_
1893     \l_stex_current_module_str ?
1894     \l_stex_symdecl_name_str
1895   }_prop
1896   ){
1897     % TODO throw error (beware of circular dependencies)
1898   }
1899
1900   \prop_clear:N \l_tmpa_prop
1901   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
1902   \seq_clear:N \l_tmpa_seq
1903   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1904   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1905
1906   \str_if_empty:NT \l_stex_symdecl_deprecate_str {
1907     \str_if_empty:NF \l_stex_module_deprecate_str {
1908       \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
1909     }
1910   }
1911   \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
1912
1913   \exp_args:No \stex_add_constant_to_current_module:n {
1914     \l_stex_symdecl_name_str
1915   }
1916
1917   % arity/args
1918   \int_zero:N \l_tmpb_int
1919
1920   \bool_set_true:N \l_tmpa_bool
1921   \str_map_inline:Nn \l_stex_symdecl_args_str {
1922     \token_case_meaning:NnF ##1 {
1923       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1924       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1925       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1926       {\tl_to_str:n a} {
1927         \bool_set_false:N \l_tmpa_bool
1928         \int_incr:N \l_tmpb_int
1929       }
1930       {\tl_to_str:n B} {
1931         \bool_set_false:N \l_tmpa_bool
1932         \int_incr:N \l_tmpb_int
1933       }
1934     }{
1935       \msg_error:nnxx{stex}{error/wrongargs}{
```

```

1936         \l_stex_current_module_str ?
1937         \l_stex_symdecl_name_str
1938     }{##1}
1939 }
1940 }
1941 \bool_if:NTF \l_tmpa_bool {
1942     % possibly numeric
1943     \str_if_empty:NTF \l_stex_symdecl_args_str {
1944         \prop_put:Nnn \l_tmpa_prop { args } {}
1945         \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1946     }{
1947         \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1948         \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1949         \str_clear:N \l_tmpa_str
1950         \int_step_inline:nn \l_tmpa_int {
1951             \str_put_right:Nn \l_tmpa_str i
1952         }
1953         \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1954     }
1955 } {
1956     \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1957     \prop_put:Nnx \l_tmpa_prop { arity }
1958     { \str_count:N \l_stex_symdecl_args_str }
1959 }
1960 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
1961
1962
1963 % semantic macro
1964
1965 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1966     \exp_args:Nx \stex_do_up_to_module:n {
1967         \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1968             \l_stex_current_module_str ? \l_stex_symdecl_name_str
1969         }}
1970     }
1971
1972     \bool_if:NF \l_stex_symdecl_local_bool {
1973         \exp_args:Nx \stex_add_to_current_module:n {
1974             \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1975                 \l_stex_current_module_str ? \l_stex_symdecl_name_str
1976             } }
1977         }
1978     }
1979 }
1980
1981 \stex_debug:nn{symbols}{New~symbol:~
1982     \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
1983     Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
1984     Args:~\prop_item:Nn \l_tmpa_prop { args }
1985 }
1986
1987 % circular dependencies require this:
1988
1989 \prop_if_exist:cF {

```

```

1990     \l_stex_symdecl_
1991     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1992     _prop
1993   } {
1994     \prop_set_eq:cN {
1995       \l_stex_symdecl_
1996       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1997       _prop
1998     } \l_tmpa_prop
1999   }
2000
2001   \seq_clear:c {
2002     \l_stex_symdecl_
2003     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2004     _notations
2005   }
2006
2007   \bool_if:NF \l_stex_symdecl_local_bool {
2008     \exp_args:Nx
2009     \stex_add_to_current_module:n {
2010       \seq_clear:c {
2011         \l_stex_symdecl_
2012         \l_stex_current_module_str ? \l_stex_symdecl_name_str
2013         _notations
2014       }
2015       \prop_set_from_keyval:cn {
2016         \l_stex_symdecl_
2017         \l_stex_current_module_str ? \l_stex_symdecl_name_str
2018         _prop
2019       } {
2020         name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2021         module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2022         type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2023         args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2024         arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2025         assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
2026       }
2027     }
2028   }
2029
2030   \stex_if_smsmode:F {
2031     % \exp_args:Nx \stex_do_up_to_module:n {
2032     %   \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2033     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2034     %   }
2035     % }
2036   \stex_if_do_html:T {
2037     \stex_annotate_invisible:nnn {symdecl} {
2038       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2039     } {
2040       \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}{}}{\l_st
2041       \stex_annotate_invisible:nnn{args}{}}{
2042         \prop_item:Nn \l_tmpa_prop { args }
2043       }

```

```

2044 \stex_annotate_invisible:nnn{macroname}{#1}{ }
2045 \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2046   \stex_annotate_invisible:nnn{definiens}{ }
2047   { $\l_stex_symdecl_definiens_tl$ }
2048 }
2049 \str_if_empty:NF \l_stex_symdecl_assoctype_str {
2050   \stex_annotate_invisible:nnn{assoctype}{\l_stex_symdecl_assoctype_str}{ }
2051 }
2052 }
2053 }
2054 }
2055 }

```

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

### `\stex_get_symbol:n`

```

2056 \str_new:N \l_stex_get_symbol_uri_str
2057
2058 \cs_new_protected:Nn \stex_get_symbol:n {
2059   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2060     \__stex_symdecl_get_symbol_from_cs:n { #1 }
2061   }{
2062     % argument is a string
2063     % is it a command name?
2064     \cs_if_exist:cTF { #1 }{
2065       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2066       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2067       \str_if_empty:NTF \l_tmpa_str {
2068         \exp_args:Nx \cs_if_eq:NNTF {
2069           \tl_head:N \l_tmpa_tl
2070         } \stex_invoke_symbol:n {
2071           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
2072         }{
2073           \__stex_symdecl_get_symbol_from_string:n { #1 }
2074         }
2075       } {
2076         \__stex_symdecl_get_symbol_from_string:n { #1 }
2077       }
2078     }{
2079       % argument is not a command name
2080       \__stex_symdecl_get_symbol_from_string:n { #1 }
2081       % \l_stex_all_symbols_seq
2082     }
2083   }
2084   \str_if_eq:eeF {
2085     \prop_item:cn {
2086       l_stex_symdecl_\l_stex_get_symbol_uri_str _prop
2087     }{ deprecate }
2088   }{}{
2089     \msg_warning:nnxx{stex}{warning/deprecated}{
2090       Symbol~\l_stex_get_symbol_uri_str
2091     }{
2092       \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{ deprecate }
2093     }
2094   }

```

```

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

```

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

## 30.2 Notations

```

2136 <@=stex_notation>
      notation arguments:
2137 \keys_define:nn { stex / notation } {
2138   lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2139   variant   .tl_set_x:N = \l__stex_notation_variant_str ,
2140   prec      .str_set_x:N = \l__stex_notation_prec_str ,

```

```

2141 op .tl_set:N = \l__stex_notation_op_tl ,
2142 primary .bool_set:N = \l__stex_notation_primary_bool ,
2143 primary .default:n = {true} ,
2144 unknown .code:n = \str_set:Nx
2145 \l__stex_notation_variant_str \l_keys_key_str
2146 }
2147
2148 \cs_new_protected:Nn \stex_notation_args:n {
2149 \str_clear:N \l__stex_notation_lang_str
2150 \str_clear:N \l__stex_notation_variant_str
2151 \str_clear:N \l__stex_notation_prec_str
2152 \tl_clear:N \l__stex_notation_op_tl
2153 \bool_set_false:N \l__stex_notation_primary_bool
2154
2155 \keys_set:nn { stex / notation } { #1 }
2156 }

```

**\notation**

```

2157 \NewDocumentCommand \notation { s m O{} } {
2158 \stex_notation_args:n { #3 }
2159 \tl_clear:N \l_stex_symdecl_definiens_tl
2160 \stex_get_symbol:n { #2 }
2161 \tl_set:Nn \l_stex_notation_after_do_tl {
2162 \__stex_notation_final:
2163 \IfBooleanTF#1{
2164 \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2165 }{}
2166 \stex_smsmode_do:
2167 }
2168 \stex_notation_do:nnnn
2169 { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str_prop } { args } }
2170 { \prop_item:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str_prop } { arity } }
2171 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2172 }
2173 \stex_deactivate_macro:Nn \notation {module-environments}

```

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

**\stex\_notation\_do:nnnn**

```

2174 \seq_new:N \l__stex_notation_precedences_seq
2175 \tl_new:N \l__stex_notation_opprec_tl
2176 \int_new:N \l__stex_notation_currarg_int
2177 \tl_new:N \stex_symbol_after_invocation_tl
2178
2179 \cs_new_protected:Nn \stex_notation_do:nnnn {
2180 \let\l_stex_current_symbol_str\relax
2181 \seq_clear:N \l__stex_notation_precedences_seq
2182 \tl_clear:N \l__stex_notation_opprec_tl
2183 \str_set:Nx \l__stex_notation_args_str { #1 }
2184 \str_set:Nx \l__stex_notation_arity_str { #2 }
2185 \str_set:Nx \__stex_notation_suffix_str { #3 }
2186
2187 % precedences
2188 \str_if_empty:NTF \l__stex_notation_prec_str {
2189 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {

```



```

2190     \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2191   }{
2192     \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2193   }
2194 } {
2195   \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2196     \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2197     \int_step_inline:nn { \l__stex_notation_arity_str } {
2198       \exp_args:NNo
2199       \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2200     }
2201   }{
2202     \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2203     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2204       \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2205       \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2206         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2207           \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2208         \seq_map_inline:Nn \l_tmpa_seq {
2209           \seq_put_right:Nn \l_tmpb_seq { ##1 }
2210         }
2211       }
2212     }{
2213       \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2214         \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2215       }{
2216         \tl_set:No \l__stex_notation_opprec_tl { 0 }
2217       }
2218     }
2219   }
2220 }
2221
2222 \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2223 \int_step_inline:nn { \l__stex_notation_arity_str } {
2224   \seq_pop_left:NNTF \l_tmpa_seq \l_tmpb_str {
2225     \exp_args:NNo
2226     \seq_put_right:No \l__stex_notation_precedences_seq {
2227       \l__stex_notation_opprec_tl
2228     }
2229   }
2230 }
2231 \tl_clear:N \l_stex_notation_dummyargs_tl
2232
2233 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2234   \exp_args:NNe
2235   \cs_set:Npn \l_stex_notation_macrocode_cs {
2236     \_stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2237     { \_stex_notation_suffix_str }
2238     { \l__stex_notation_opprec_tl }
2239     { \exp_not:n { #4 } }
2240   }
2241   \l_stex_notation_after_do_tl
2242 }{
2243   \str_if_in:NnTF \l__stex_notation_args_str b {

```

```

2244 \exp_args:Nne \use:nn
2245 {
2246 \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2247 \cs_set:Npn \l__stex_notation_arity_str } { {
2248 \stex_term_math_omb:nmmm { \l_stex_current_symbol_str }
2249 { \__stex_notation_suffix_str }
2250 { \l__stex_notation_opprec_tl }
2251 { \exp_not:n { #4 } }
2252 }}
2253 }{
2254 \str_if_in:NnTF \l__stex_notation_args_str B {
2255 \exp_args:Nne \use:nn
2256 {
2257 \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2258 \cs_set:Npn \l__stex_notation_arity_str } { {
2259 \stex_term_math_omb:nmmm { \l_stex_current_symbol_str }
2260 { \__stex_notation_suffix_str }
2261 { \l__stex_notation_opprec_tl }
2262 { \exp_not:n { #4 } }
2263 } }
2264 }{
2265 \exp_args:Nne \use:nn
2266 {
2267 \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2268 \cs_set:Npn \l__stex_notation_arity_str } { {
2269 \stex_term_math_oma:nmmm { \l_stex_current_symbol_str }
2270 { \__stex_notation_suffix_str }
2271 { \l__stex_notation_opprec_tl }
2272 { \exp_not:n { #4 } }
2273 } }
2274 }
2275 }
2276
2277 \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2278 \int_zero:N \l__stex_notation_currarg_int
2279 \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2280 \__stex_notation_arguments:
2281 }
2282 }

```

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

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

```

2283 \cs_new_protected:Nn \__stex_notation_arguments: {
2284 \int_incr:N \l__stex_notation_currarg_int
2285 \str_if_empty:NnTF \l__stex_notation_remaining_args_str {
2286 \l_stex_notation_after_do_tl
2287 }{
2288 \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2289 \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2290 \str_if_eq:NnTF \l_tmpa_str a {
2291 \__stex_notation_argument_assoc:n
2292 }{
2293 \str_if_eq:NnTF \l_tmpa_str B {

```

```

2294     \_stex_notation_argument_assoc:n
2295   }{
2296     \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2297     \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2298       { \_stex_term_math_arg:nnn
2299         { \int_use:N \l__stex_notation_currarg_int }
2300         { \l_tmpa_str }
2301         { ####\int_use:N \l__stex_notation_currarg_int }
2302       }
2303     }
2304     \_stex_notation_arguments:
2305   }
2306 }
2307 }
2308 }

```

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

\\_stex\_notation\_argument\_assoc:n

```

2309 \cs_new_protected:Nn \_stex_notation_argument_assoc:n {
2310
2311   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2312     {\l__stex_notation_arity_str}{
2313       #1
2314     }
2315   \int_zero:N \l_tmpa_int
2316   \tl_clear:N \l_tmpa_tl
2317   \str_map_inline:Nn \l__stex_notation_args_str {
2318     \int_incr:N \l_tmpa_int
2319     \tl_put_right:Nx \l_tmpa_tl {
2320       \str_if_eq:nnTF {##1}{a}{ {} } {}
2321       \str_if_eq:nnTF {##1}{B}{ {} } {}
2322       {\_stex_term_arg:nn{\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa_in
2323     }
2324   }
2325 }
2326 }
2327 \exp_after:wN\exp_after:wN\exp_after:wN \def
2328 \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2329 \exp_after:wN\exp_after:wN\exp_after:wN ##
2330 \exp_after:wN\exp_after:wN\exp_after:wN 1
2331 \exp_after:wN\exp_after:wN\exp_after:wN ##
2332 \exp_after:wN\exp_after:wN\exp_after:wN 2
2333 \exp_after:wN\exp_after:wN\exp_after:wN {
2334   \exp_after:wN \exp_after:wN \exp_after:wN
2335   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2336     \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2337   }
2338 }
2339
2340 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2341 \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
2342   \_stex_term_math_assoc_arg:nnnn
2343   { \int_use:N \l__stex_notation_currarg_int }

```

```

2344     { \l_tmpa_str }
2345     { ####\int_use:N \l__stex_notation_currarg_int }
2346     { \l_tmpa_cs {####1} {####2} }
2347   } }
2348   \__stex_notation_arguments:
2349 }

```

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

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

```

2350 \cs_new_protected:Nn \__stex_notation_final: {
2351   \exp_args:Nne \use:nn
2352   {
2353     \cs_generate_from_arg_count:cNnn {
2354       stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2355       \__stex_notation_suffix_str
2356       _cs
2357     }
2358     \cs_set:Npn \l__stex_notation_arity_str { { {
2359       \exp_after:wN \exp_after:wN \exp_after:wN
2360       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2361       { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2362     } } }
2363
2364     \tl_if_empty:NF \l__stex_notation_op_tl {
2365       \cs_set:cpx {
2366         stex_op_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2367         \__stex_notation_suffix_str
2368         _cs
2369       } {
2370         \stex_term_oms:nnn {
2371           \l_stex_get_symbol_uri_str \c_hash_str \__stex_notation_suffix_str
2372         }{
2373           \l_stex_get_symbol_uri_str
2374         }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2375       }
2376     }
2377
2378     \exp_args:Ne
2379     \stex_add_to_current_module:n {
2380       \cs_generate_from_arg_count:cNnn {
2381         stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2382         \__stex_notation_suffix_str
2383         _cs
2384       } \cs_set:Npn {\l__stex_notation_arity_str} {
2385         \exp_after:wN \exp_after:wN \exp_after:wN
2386         \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2387         { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2388       }
2389       \tl_if_empty:NF \l__stex_notation_op_tl {
2390         \cs_set:cpn {
2391           stex_op_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2392           \__stex_notation_suffix_str
2393           _cs

```

```

2394     } {
2395         \stex_term_oms:nnn {
2396             \l_stex_get_symbol_uri_str\c_hash_str \_stex_notation_suffix_str
2397         }{
2398             \l_stex_get_symbol_uri_str
2399         }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2400     }
2401 }
2402 }
2403 %\exp_args:Nx
2404 % \stex_do_up_to_module:n {
2405     \seq_put_right:cx {
2406         l_stex_symdecl_ \l_stex_get_symbol_uri_str
2407         _notations
2408     } {
2409         \_stex_notation_suffix_str
2410     }
2411 % }
2412
2413 \stex_debug:nn{symbols}{
2414     Notation~\_stex_notation_suffix_str
2415     ~for~\l_stex_get_symbol_uri_str^^J
2416     Operator~precedence:~\l__stex_notation_opprec_tl^^J
2417     Argument~precedences:~
2418     \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2419     Notation: \cs_meaning:c {
2420         stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2421         \_stex_notation_suffix_str
2422         _cs
2423     }
2424 }
2425
2426 \exp_args:Ne
2427 \stex_add_to_current_module:n {
2428     \seq_put_right:cn {
2429         l_stex_symdecl_\l_stex_get_symbol_uri_str
2430         _notations
2431     } { \_stex_notation_suffix_str }
2432 }
2433
2434 \stex_if_smsmode:F {
2435
2436     % HTML annotations
2437     \stex_if_do_html:T {
2438         \stex_annotate_invisible:nnn { notation }
2439         { \l_stex_get_symbol_uri_str } {
2440             \stex_annotate_invisible:nnn { notationfragment }
2441             { \_stex_notation_suffix_str }{}
2442             \stex_annotate_invisible:nnn { precedence }
2443             { \l__stex_notation_prec_str }{}
2444
2445             \int_zero:N \l_tmpa_int
2446             \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2447             \tl_clear:N \l_tmpa_tl

```

```

2448 \int_step_inline:nn { \l__stex_notation_arity_str }{
2449 \int_incr:N \l_tmpa_int
2450 \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2451 \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_r
2452 \str_if_eq:VnTF \l_tmpb_str a {
2453 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2454 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2455 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2456 } }
2457 }{
2458 \str_if_eq:VnTF \l_tmpb_str B {
2459 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2460 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2461 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2462 } }
2463 }{
2464 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2465 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2466 } }
2467 }
2468 }
2469 }
2470 \stex_annotate_invisible:nnn { notationcomp }{}{
2471 \str_set:Nx \l_stex_current_symbol_str { \l_stex_get_symbol_uri_str }
2472 $ \exp_args:Nno \use:nn { \use:c {
2473 stex_notation_ \l_stex_current_symbol_str
2474 \c_hash_str \__stex_notation_suffix_str _cs
2475 } } { \l_tmpa_tl } $
2476 }
2477 }
2478 }
2479 }
2480 }

```

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

\setnotation

```

2481 \keys_define:nn { stex / setnotation } {
2482 lang .tl_set_x:N = \l__stex_notation_lang_str ,
2483 variant .tl_set_x:N = \l__stex_notation_variant_str ,
2484 unknown .code:n = \str_set:Nx
2485 \l__stex_notation_variant_str \l_keys_key_str
2486 }
2487
2488 \cs_new_protected:Nn \stex_setnotation_args:n {
2489 \str_clear:N \l__stex_notation_lang_str
2490 \str_clear:N \l__stex_notation_variant_str
2491 \keys_set:nn { stex / setnotation } { #1 }
2492 }
2493
2494 \cs_new_protected:Nn \stex_setnotation:n {
2495 \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1 _notations }
2496 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2497 \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }

```

```

2498     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2499 \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2500     { \c_hash_str }
2501 \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_#1 _notations }
2502     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2503 \exp_args:Nx \stex_add_to_current_module:n {
2504     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2505     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2506     \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_#1 _notations }
2507     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2508     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2509     { \c_hash_str }
2510 }
2511 \stex_debug:nn {notations}{
2512     Setting~default~notation~
2513     {\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str}~for~
2514     #1 \\
2515     \expandafter\meaning\csname
2516     l_stex_symdecl_#1 _notations\endcsname
2517 }
2518 }{
2519     % todo throw error
2520 }
2521 }
2522
2523 \NewDocumentCommand \setnotation {m m} {
2524     \stex_get_symbol:n { #1 }
2525     \_stex_setnotation_args:n { #2 }
2526     \stex_setnotation:n{\l_stex_get_symbol_uri_str}
2527     \stex_smsmode_do:
2528 }
2529
2530 \cs_new_protected:Nn \stex_copy_notations:nn {
2531     \stex_debug:nn {notations}{
2532         Copying~notations~from~#2~to~#1\\
2533         \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
2534     }
2535     \tl_clear:N \l_tmpa_tl
2536     \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2537         \tl_put_right:Nn \l_tmpa_tl { {##} ##1 }
2538     }
2539     \seq_map_inline:cn {l_stex_symdecl_#2_notations}{
2540         \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2541         \edef \l_tmpa_tl {
2542             \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2543             \exp_after:wN\exp_after:wN\exp_after:wN {
2544                 \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2545             }
2546         }
2547         \exp_args:Nx
2548         \stex_do_up_to_module:n {
2549             \seq_put_right:cn{l_stex_symdecl_#1_notations}{##1}
2550             \cs_generate_from_arg_count:cNnn {
2551                 stex_notation_ #1 \c_hash_str ##1 _cs

```

```

2552     } \cs_set:Npn { \prop_item:cn {l_stex_symdecl_#2_prop}}{ arity } }{
2553     \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl}
2554   }
2555 }
2556 }
2557 }
2558
2559 \NewDocumentCommand \copynotation {m m} {
2560   \stex_get_symbol:n { #1 }
2561   \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2562   \stex_get_symbol:n { #2 }
2563   \exp_args:Noo
2564   \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2565   \exp_args:Nx \stex_add_import_to_current_module:n{
2566     \stex_copy_notations:nn {\l_tmpa_str} {\l_stex_get_symbol_uri_str}
2567   }
2568   \stex_smsmode_do:
2569 }
2570

```

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

## \symdef

```

2571 \keys_define:nn { stex / symdef } {
2572   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2573   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2574   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2575   type      .tl_set:N   = \l_stex_symdecl_type_tl ,
2576   def       .tl_set:N   = \l_stex_symdecl_definiens_tl ,
2577   op        .tl_set:N   = \l__stex_notation_op_tl ,
2578   lang      .str_set_x:N = \l__stex_notation_lang_str ,
2579   variant   .str_set_x:N = \l__stex_notation_variant_str ,
2580   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2581   assoc     .choices:nn =
2582     {bin,binl,binr,pre,conj,pwconj}
2583     {\str_set:Nx \l_stex_symdecl_assoc_type_str {\l_keys_choice_tl}},
2584   unknown   .code:n      = \str_set:Nx
2585     \l__stex_notation_variant_str \l_keys_key_str
2586 }
2587
2588 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2589   \str_clear:N \l_stex_symdecl_name_str
2590   \str_clear:N \l_stex_symdecl_args_str
2591   \str_clear:N \l_stex_symdecl_assoc_type_str
2592   \bool_set_false:N \l_stex_symdecl_local_bool
2593   \tl_clear:N \l_stex_symdecl_type_tl
2594   \tl_clear:N \l_stex_symdecl_definiens_tl
2595   \str_clear:N \l__stex_notation_lang_str
2596   \str_clear:N \l__stex_notation_variant_str
2597   \str_clear:N \l__stex_notation_prec_str
2598   \tl_clear:N \l__stex_notation_op_tl
2599
2600   \keys_set:nn { stex / symdef } { #1 }
2601 }

```



```

2602
2603 \NewDocumentCommand \symdef { m O{} } {
2604   \__stex_notation_symdef_args:n { #2 }
2605   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2606   \stex_symdecl_do:n { #1 }
2607   \tl_set:Nn \l_stex_notation_after_do_tl {
2608     \__stex_notation_final:
2609     \stex_smsmode_do:
2610   }
2611   \str_set:Nx \l_stex_get_symbol_uri_str {
2612     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2613   }
2614   \exp_args:Nx \stex_notation_do:nnnn
2615     { \prop_item:cn {l_stex_symdecl_l_stex_get_symbol_uri_str_prop } { args } }
2616     { \prop_item:cn { l_stex_symdecl_l_stex_get_symbol_uri_str_prop } { arity } }
2617     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2618   }
2619   \stex_deactivate_macro:Nn \symdef {module~environments}

```

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

### 30.3 Variables

```

2620 <@@=stex_variables>
2621
2622 \keys_define:nn { stex / vardef } {
2623   name      .str_set_x:N = \l__stex_variables_name_str ,
2624   args      .str_set_x:N = \l__stex_variables_args_str ,
2625   type      .tl_set:N   = \l__stex_variables_type_tl ,
2626   def       .tl_set:N   = \l__stex_variables_def_tl ,
2627   op        .tl_set:N   = \l__stex_variables_op_tl ,
2628   prec      .str_set_x:N = \l__stex_variables_prec_str ,
2629   assoc     .choices:nn =
2630     {bin,binl,binr,pre,conj,pwconj}
2631     {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
2632   bind      .choices:nn =
2633     {forall,exists}
2634     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2635 }
2636
2637 \cs_new_protected:Nn \__stex_variables_args:n {
2638   \str_clear:N \l__stex_variables_name_str
2639   \str_clear:N \l__stex_variables_args_str
2640   \str_clear:N \l__stex_variables_prec_str
2641   \str_clear:N \l__stex_variables_assoctype_str
2642   \str_clear:N \l__stex_variables_bind_str
2643   \tl_clear:N \l__stex_variables_type_tl
2644   \tl_clear:N \l__stex_variables_def_tl
2645   \tl_clear:N \l__stex_variables_op_tl
2646
2647   \keys_set:nn { stex / vardef } { #1 }
2648 }
2649
2650 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{} } {

```

```

2651 \__stex_variables_args:n {#2}
2652 \str_if_empty:NT \l__stex_variables_name_str {
2653   \str_set:Nx \l__stex_variables_name_str { #1 }
2654 }
2655 \prop_clear:N \l_tmpa_prop
2656 \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
2657
2658 \int_zero:N \l_tmpb_int
2659 \bool_set_true:N \l_tmpa_bool
2660 \str_map_inline:Nn \l__stex_variables_args_str {
2661   \token_case_meaning:NnF ##1 {
2662     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2663     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2664     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2665     {\tl_to_str:n a} {
2666       \bool_set_false:N \l_tmpa_bool
2667       \int_incr:N \l_tmpb_int
2668     }
2669     {\tl_to_str:n B} {
2670       \bool_set_false:N \l_tmpa_bool
2671       \int_incr:N \l_tmpb_int
2672     }
2673   }{
2674     \msg_error:nnxx{stex}{error/wrongargs}{
2675       variable~\l__stex_variables_name_str
2676     }{##1}
2677   }
2678 }
2679 \bool_if:NTF \l_tmpa_bool {
2680   % possibly numeric
2681   \str_if_empty:NTF \l__stex_variables_args_str {
2682     \prop_put:Nnn \l_tmpa_prop { args } {}
2683     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2684   }{
2685     \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
2686     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2687     \str_clear:N \l_tmpa_str
2688     \int_step_inline:nn \l_tmpa_int {
2689       \str_put_right:Nn \l_tmpa_str i
2690     }
2691     \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
2692     \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2693   }
2694 } {
2695   \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2696   \prop_put:Nnx \l_tmpa_prop { arity }
2697   { \str_count:N \l__stex_variables_args_str }
2698 }
2699 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
2700 \tl_set:cx { #1 }{ \stex_invoke_variable:n { \l__stex_variables_name_str } }
2701
2702 \prop_set_eq:cN { l_stex_variable_\l__stex_variables_name_str _prop } \l_tmpa_prop
2703
2704 \tl_if_empty:NF \l__stex_variables_op_tl {

```

```

2705 \cs_set:cpx {
2706   stex_var_op_notation_ \l__stex_variables_name_str _cs
2707 } {
2708   \stex_term_omv:nn {
2709     var://\l__stex_variables_name_str
2710   }{ \comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
2711 }
2712 }
2713
2714 \tl_set:Nn \l_stex_notation_after_do_tl {
2715   \exp_args:Nne \use:nn {
2716     \cs_generate_from_arg_count:cNnn { stex_var_notation_ \l__stex_variables_name_str _cs }
2717     \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
2718   } {
2719     \exp_after:wN \exp_after:wN \exp_after:wN
2720     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2721     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_symbol
2722   }}
2723 \stex_if_do_html:T {
2724   \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
2725     \stex_annotate_invisible:nnn { precedence }
2726     { \l__stex_variables_prec_str }{
2727     \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{}}{\$ \l__
2728     \stex_annotate_invisible:nnn{args}{}}{ \l__stex_variables_args_str }
2729     \stex_annotate_invisible:nnn{macroname}{#1}{
2730     \tl_if_empty:NF \l__stex_variables_def_tl {
2731       \stex_annotate_invisible:nnn{definiens}{
2732         { \$ \l__stex_variables_def_tl }
2733       }
2734     \str_if_empty:NF \l__stex_variables_assoctype_str {
2735       \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{
2736     }
2737     \int_zero:N \l_tmpa_int
2738     \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
2739     \tl_clear:N \l_tmpa_tl
2740     \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
2741       \int_incr:N \l_tmpa_int
2742       \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
2743       \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables
2744       \str_if_eq:VnTF \l_tmpb_str a {
2745         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2746           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2747           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2748         } }
2749       }{
2750         \str_if_eq:VnTF \l_tmpb_str B {
2751           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2752             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2753             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2754           } }
2755         }{
2756           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2757             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2758           } }

```

```

2759         }
2760     }
2761 }
2762 \stex_annotate_invisible:nnn { notationcomp }{}{
2763     \str_set:Nx \l_stex_current_symbol_str {var://\l__stex_variables_name_str }
2764     $ \exp_args:Nno \use:nn { \use:c {
2765         stex_var_notation_\l__stex_variables_name_str _cs
2766     } } { \l_tmpa_tl } $
2767 }
2768 }
2769 }
2770 }
2771
2772 \stex_notation_do:nnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { arit
2773 }
2774
2775 \cs_new:Nn \__stex_variables_reset:N {
2776     \tl_if_exist:NTF #1 {
2777         \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
2778     }{
2779         \let \exp_not:N #1 \exp_not:N \undefined
2780     }
2781 }
2782
2783 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
2784     \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
2785     \exp_args:Nnx \use:nn {
2786         % TODO
2787         \stex_annotate_invisible:nnn {vardecls}{\clist_use:Nn\l__stex_variables_names,}{
2788             #2
2789         }
2790     }{
2791         \__stex_variables_reset:N \varnot
2792         \__stex_variables_reset:N \vartype
2793         \__stex_variables_reset:N \vardef
2794     }
2795 }
2796
2797 \NewDocumentCommand \vardef { s } {
2798     \IfBooleanTF#1 {
2799         \__stex_variables_do_complex:nn
2800     }{
2801         \__stex_variables_do_simple:nnn
2802     }
2803 }
2804
2805 \NewDocumentCommand \svar { O{} m }{
2806     \tl_if_empty:nTF {#1}{
2807         \str_set:Nn \l_tmpa_str { #2 }
2808     }{
2809         \str_set:Nn \l_tmpa_str { #1 }
2810     }
2811     \stex_term_omv:nn {
2812         var://\l_tmpa_str

```

```
2813     }{ \comp{ #2 } }  
2814 }  
2815  
2816 </package>
```

## Chapter 31

# STEX -Terms Implementation

```
2817 <*package>
2818
2819 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2820
2821 <@@=stex_terms>
2822
2823   Warnings and error messages
2824   \msg_new:nnn{stex}{error/nonotation}{
2825     Symbol~#1~invoked,~but~has~no~notation~#2!
2826   }
2827   \msg_new:nnn{stex}{error/notationarg}{
2828     Error~in~parsing~notation~#1
2829   }
2830   \msg_new:nnn{stex}{error/noop}{
2831     Symbol~#1~has~no~operator~notation~for~notation~#2
2832   }
2833   \msg_new:nnn{stex}{error/notallowed}{
2834     Symbol~invocation~#1~not~allowed~in~notation~component~of~#2
2835   }
2836
```

### 31.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
2835 \keys_define:nn { stex / terms } {
2836   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2837   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2838   unknown .code:n = \str_set:Nx
2839     \l__stex_terms_variant_str \l_keys_key_str
2840 }
2841
2842 \cs_new_protected:Nn \__stex_terms_args:n {
2843   \str_clear:N \l__stex_terms_lang_str
2844   \str_clear:N \l__stex_terms_variant_str
2845
```

```

2846 \keys_set:nn { stex / terms } { #1 }
2847 }
2848
2849 \cs_new:Nn \__stex_terms_reset:N {
2850 \tl_if_exist:NTF #1 {
2851 \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
2852 }{
2853 \let \exp_not:N #1 \exp_not:N \undefined
2854 }
2855 }
2856
2857 \bool_new:N \l__stex_terms_allow_semantic_bool
2858 \bool_set_true:N \l__stex_terms_allow_semantic_bool
2859
2860 \cs_new_protected:Nn \stex_invoke_symbol:n {
2861 \bool_if:NTF \l__stex_terms_allow_semantic_bool {
2862 \str_if_eq:eeF {
2863 \prop_item:cn {
2864 l_stex_symdecl_#1_prop
2865 }{ deprecate }
2866 }{}{
2867 \msg_warning:nxxx{stex}{warning/deprecated}{
2868 Symbol~#1
2869 }{
2870 \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
2871 }
2872 }
2873 \if_mode_math:
2874 \exp_after:wN \__stex_terms_invoke_math:n
2875 \else:
2876 \exp_after:wN \__stex_terms_invoke_text:n
2877 \fi: { #1 }
2878 }{
2879 \msg_error:nxxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
2880 }
2881 }
2882
2883 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
2884 \peek_charcode_remove:NTF ! {
2885 \__stex_terms_invoke_op_custom:nn {#1}
2886 }{
2887 \__stex_terms_invoke_custom:nn {#1}
2888 }
2889 }
2890
2891 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2892 \peek_charcode_remove:NTF ! {
2893 % operator
2894 \peek_charcode_remove:NTF * {
2895 % custom op
2896 \__stex_terms_invoke_op_custom:nn {#1}
2897 }{
2898 % op notation
2899 \peek_charcode:NTF [ {

```

```

2900     \__stex_terms_invoke_op_notation:nw {#1}
2901   }{
2902     \__stex_terms_invoke_op_notation:nw {#1}[]
2903   }
2904 }
2905 }{
2906   \peek_charcode_remove:NTF * {
2907     \__stex_terms_invoke_custom:nn {#1}
2908     % custom
2909   }{
2910     % normal
2911     \peek_charcode:NTF [ {
2912       \__stex_terms_invoke_notation:nw {#1}
2913     }{
2914       \__stex_terms_invoke_notation:nw {#1}[]
2915     }
2916   }
2917 }
2918 }
2919
2920
2921 \cs_new_protected:Nn \__stex_terms_invoke_op_custom:nn {
2922   \exp_args:Nnx \use:nn {
2923     \str_set:Nn \l_stex_current_symbol_str { #1 }
2924     \bool_set_false:N \l__stex_terms_allow_semantic_bool
2925     \stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
2926       \comp{ #2 }
2927     }
2928   }{
2929     \__stex_terms_reset:N \l_stex_current_symbol_str
2930     \bool_set_true:N \l__stex_terms_allow_semantic_bool
2931   }
2932 }
2933
2934 \cs_new_protected:Nn \__stex_terms_find_notation:nn {
2935   \str_set:Nn \l_stex_current_symbol_str { #1 }
2936   \__stex_terms_args:n { #2 }
2937   \seq_if_empty:cTF {
2938     l_stex_symdecl_ #1 _notations
2939   } {
2940     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
2941   } {
2942     \bool_lazy_all:nTF {
2943       {\str_if_empty_p:N \l__stex_terms_variant_str}
2944       {\str_if_empty_p:N \l__stex_terms_lang_str}
2945     }{
2946       \seq_get_left:cN {l_stex_symdecl_#1_notations}\l__stex_terms_variant_str
2947     }{
2948       \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
2949         \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2950       }{
2951         \str_set:Nx \l__stex_terms_variant_str { \l__stex_terms_variant_str \c_hash_str \l__
2952       }{
2953         \msg_error:nnxx{stex}{error/nonotation}{#1}{

```



```

2954         ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2955     }
2956 }
2957 }
2958 }
2959 }
2960
2961 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
2962     \__stex_terms_find_notation:nn { #1 }{ #2 }
2963     \bool_set_false:N \l__stex_terms_allow_semantic_bool
2964     \cs_if_exist:cTF {
2965         stex_op_notation_ #1 \c_hash_str \l__stex_terms_variant_str _cs
2966     }{
2967         \use:c{stex_op_notation_ #1 \c_hash_str \l__stex_terms_variant_str _cs}
2968     }{
2969         \msg_error:nnxx{stex}{error/noop}{#1}{\l__stex_terms_variant_str}
2970     }
2971     \bool_set_true:N \l__stex_terms_allow_semantic_bool
2972 }
2973
2974 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {
2975     \__stex_terms_find_notation:nn { #1 }{ #2 }
2976     \cs_if_exist:cTF {
2977         stex_notation_ #1 \c_hash_str \l__stex_terms_variant_str _cs
2978     }{
2979         \tl_set:Nx \stex_symbol_after_invocation_tl {
2980             \__stex_terms_reset:N \stex_symbol_after_invocation_tl
2981             \__stex_terms_reset:N \l_stex_current_symbol_str
2982             \bool_set_true:N \l__stex_terms_allow_semantic_bool
2983         }
2984         \bool_set_false:N \l__stex_terms_allow_semantic_bool
2985         \use:c{stex_notation_ #1 \c_hash_str \l__stex_terms_variant_str _cs}
2986     }{
2987         \msg_error:nnxx{stex}{error/nonotation}{#1}{
2988             ~\l__stex_terms_variant_str
2989         }
2990     }
2991 }
2992
2993 \prop_new:N \l__stex_terms_custom_args_prop
2994
2995 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
2996     \exp_args:Nnx \use:nn {
2997         \bool_set_false:N \l__stex_terms_allow_semantic_bool
2998         \str_set:Nn \l_stex_current_symbol_str { #1 }
2999         \prop_clear:N \l__stex_terms_custom_args_prop
3000         \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3001         \prop_get:cnN {
3002             l_stex_symdecl_#1 _prop
3003         }{ args } \l_tmpa_str
3004         \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3005         \tl_set:Nn \arg { \__stex_terms_arg: }
3006         \str_if_empty:NTF \l_tmpa_str {
3007             \_stex_term_oms:nnn {#1}{#1}{#2}

```

```

3008     }{
3009         \str_if_in:NnTF \l_tmpa_str b {
3010             \stex_term_ombind:nnn {#1}{#1}{#2}
3011         }{
3012             \str_if_in:NnTF \l_tmpa_str B {
3013                 \stex_term_ombind:nnn {#1}{#1}{#2}
3014             }{
3015                 \stex_term_oma:nnn {#1}{#1}{#2}
3016             }
3017         }
3018     }
3019     % TODO check that all arguments exist
3020 }{
3021     \__stex_terms_reset:N \l_stex_current_symbol_str
3022     \__stex_terms_reset:N \arg
3023     \__stex_terms_reset:N \l__stex_terms_custom_args_prop
3024     \bool_set_true:N \l__stex_terms_allow_semantic_bool
3025 }
3026 }
3027
3028 \NewDocumentCommand \__stex_terms_arg: { s O{} m }{
3029     \tl_if_empty:nTF {#2}{
3030         \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
3031         \bool_set_true:N \l_tmpa_bool
3032         \bool_do_while:Nn \l_tmpa_bool {
3033             \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int}
3034             \int_incr:N \l_tmpa_int
3035         }{
3036             \bool_set_false:N \l_tmpa_bool
3037         }
3038     }
3039 }{
3040     \int_set:Nn \l_tmpa_int { #2 }
3041     \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
3042         % TODO throw error
3043     }
3044 }
3045 \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
3046 \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3047     % TODO throw error
3048 }
3049 \IfBooleanTF#1{
3050     \stex_annotate_invisible:n {
3051         \exp_args:No \stex_term_arg:nn {\l_stex_current_symbol_str}{#3}
3052     }
3053 }{
3054     \exp_args:No \stex_term_arg:nn {\l_stex_current_symbol_str}{#3}
3055 }
3056 }
3057
3058
3059 \cs_new_protected:Nn \stex_term_arg:nn {
3060     \exp_args:Nnx \use:nn {
3061         \bool_set_true:N \l__stex_terms_allow_semantic_bool

```

```

3062 \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3063 }{
3064 \bool_set_false:N \l__stex_terms_allow_semantic_bool
3065 }
3066 }
3067
3068 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
3069 \exp_args:Nnx \use:nn
3070 { \int_set:Nn \l__stex_terms_downprec { #2 }
3071 \_stex_term_arg:nn { #1 }{ #3 }
3072 }
3073 { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3074 }
3075
3076

```

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

## 31.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
3077 \tl_const:Nx \infprec {\int_use:N \c_max_int}
3078 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
3079 \int_new:N \l__stex_terms_downprec
3080 \int_set_eq:NN \l__stex_terms_downprec \infprec

```

(End definition for `\infprec`, `\neginfprec`, and `\l__stex_terms_downprec`. These variables are documented on page 37.)

Bracketing:

```

\l__stex_terms_left_bracket_str
\l__stex_terms_right_bracket_str
3081 \tl_set:Nn \l__stex_terms_left_bracket_str (
3082 \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

```

3083 \cs_new_protected:Nn \_stex_terms_maybe_brackets:nn {
3084 \bool_if:NTF \l__stex_terms_brackets_done_bool {
3085 \bool_set_false:N \l__stex_terms_brackets_done_bool
3086 #2
3087 } {
3088 \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
3089 \bool_if:NTF \l_stex_inarray_bool { #2 }{
3090 \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
3091 \dobrackets { #2 }
3092 }
3093 }{ #2 }
3094 }
3095 }

```

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

### **\dobrackets**

```
3096 \bool_new:N \l__stex_terms_brackets_done_bool
3097 %\RequirePackage{scalerel}
3098 \cs_new_protected:Npn \dobrackets #1 {
3099   %\ThisStyle{\if D\m@switch
3100   %   \exp_args:Nnx \use:nn
3101   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
3102   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
3103   % \else
3104   %   \exp_args:Nnx \use:nn
3105   %   {
3106   %     \bool_set_true:N \l__stex_terms_brackets_done_bool
3107   %     \int_set:Nn \l__stex_terms_downprec \infpref
3108   %     \l__stex_terms_left_bracket_str
3109   %     #1
3110   %   }
3111   %   {
3112   %     \bool_set_false:N \l__stex_terms_brackets_done_bool
3113   %     \l__stex_terms_right_bracket_str
3114   %     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3115   %   }
3116   %\fi}
3117 }
```

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

### **\withbrackets**

```
3118 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
3119   \exp_args:Nnx \use:nn
3120   {
3121     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
3122     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
3123     #3
3124   }
3125   {
3126     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
3127     {\l__stex_terms_left_bracket_str}
3128     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
3129     {\l__stex_terms_right_bracket_str}
3130   }
3131 }
```

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

### **\STEXinvisible**

```
3132 \cs_new_protected:Npn \STEXinvisible #1 {
3133   \stex_annotate_invisible:n { #1 }
3134 }
```

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

OMDoc terms:

`\_stex_term_math_oms:nnnn`

```
3135 \cs_new_protected:Nn \_stex_term_oms:nnn {
3136   \stex_annotate:nnn{ OMID }{ #2 }{
3137     \stex_highlight_term:nn { #1 } { #3 }
3138   }
3139 }
3140
3141 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
3142   \__stex_terms_maybe_brackets:nn { #3 }{
3143     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3144   }
3145 }
```

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

`\_stex_term_math_omv:nn`

```
3146 \cs_new_protected:Nn \_stex_term_omv:nn {
3147   \stex_annotate:nnn{ OMID }{ #1 }{
3148     \stex_highlight_term:nn { #1 } { #2 }
3149   }
3150 }
```

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

`\_stex_term_math_oma:nnnn`

```
3151 \cs_new_protected:Nn \_stex_term_oma:nnn {
3152   \stex_annotate:nnn{ OMA }{ #2 }{
3153     \stex_highlight_term:nn { #1 } { #3 }
3154   }
3155 }
3156
3157 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
3158   \__stex_terms_maybe_brackets:nn { #3 }{
3159     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3160   }
3161 }
```

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

`\_stex_term_math_omb:nnnn`

```
3162 \cs_new_protected:Nn \_stex_term_ombind:nnn {
3163   \stex_annotate:nnn{ OMBIND }{ #2 }{
3164     \stex_highlight_term:nn { #1 } { #3 }
3165   }
3166 }
3167
3168 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
3169   \__stex_terms_maybe_brackets:nn { #3 }{
3170     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3171   }
3172 }
```

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

`\stex_term_math_assoc_arg:nnnn`

```

3173 \cs_new_protected:Nn \stex_term_math_assoc_arg:nnnn {
3174   % TODO sequences
3175   \clist_set:Nn \l_tmpa_clist{ #3 }
3176   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3177     \tl_set:Nn \l_tmpa_tl { #3 }
3178   }{
3179     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3180     \clist_reverse:N \l_tmpa_clist
3181     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3182
3183     \clist_map_inline:Nn \l_tmpa_clist {
3184       \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3185         \exp_args:Nno
3186         \l_tmpa_cs { ##1 } \l_tmpa_tl
3187       }
3188     }
3189   }
3190   \exp_args:Nnno
3191   \stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
3192 }

```

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

`\stex_term_custom:nn`

```

3193 \cs_new_protected:Nn \stex_term_custom:nn {
3194   \str_set:Nn \l__stex_terms_custom_uri { #1 }
3195   \str_set:Nn \l_tmpa_str { #2 }
3196   \tl_clear:N \l_tmpa_tl
3197   \int_zero:N \l_tmpa_int
3198   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
3199   \__stex_terms_custom_loop:
3200 }

```

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

`\__stex_terms_custom_loop:`

```

3201 \cs_new_protected:Nn \__stex_terms_custom_loop: {
3202   \bool_set_false:N \l_tmpa_bool
3203   \bool_while_do:nn {
3204     \str_if_eq_p:ee X {
3205       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
3206     }
3207   }{
3208     \int_incr:N \l_tmpa_int
3209   }
3210
3211   \peek_charcode:NNTF [ {
3212     % notation/text component
3213     \__stex_terms_custom_component:w
3214   } {
3215     \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
3216       % all arguments read => finish
3217       \__stex_terms_custom_final:

```

```

3218 } {
3219   % arguments missing
3220   \peek_charcode_remove:NTF * {
3221     % invisible, specific argument position or both
3222     \peek_charcode:NTF [ {
3223       % visible specific argument position
3224       \__stex_terms_custom_arg:wn
3225     } {
3226       % invisible
3227       \peek_charcode_remove:NTF * {
3228         % invisible specific argument position
3229         \__stex_terms_custom_arg_inv:wn
3230       } {
3231         % invisible next argument
3232         \__stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
3233       }
3234     }
3235   } {
3236     % next normal argument
3237     \__stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
3238   }
3239 }
3240 }
3241 }

```

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

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

```

3242 \cs_new_protected:Npn \__stex_terms_custom_arg_inv:wn [ #1 ] #2 {
3243   \bool_set_true:N \l_tmpa_bool
3244   \__stex_terms_custom_arg:wn [ #1 ] { #2 }
3245 }

```

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

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

```

3246 \cs_new_protected:Npn \__stex_terms_custom_arg:wn [ #1 ] #2 {
3247   \str_set:Nx \l_tmpb_str {
3248     \str_item:Nn \l_tmpa_str { #1 }
3249   }
3250   \str_case:VnTF \l_tmpb_str {
3251     { X } {
3252       \msg_error:nnx{stex}{error/notationarg}{\l__stex_terms_custom_uri}
3253     }
3254     { i } { \__stex_terms_custom_set_X:n { #1 } }
3255     { b } { \__stex_terms_custom_set_X:n { #1 } }
3256     { a } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
3257     { B } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
3258   }{}{
3259     \msg_error:nnx{stex}{error/notationarg}{\l__stex_terms_custom_uri}
3260   }
3261
3262   \bool_if:nTF \l_tmpa_bool {
3263     \tl_put_right:Nx \l_tmpa_tl {
3264       \stex_annotate_invisible:n {

```

```

3265         \stex_term_arg:nn { \int_eval:n { #1 } }
3266         \exp_not:n { { #2 } }
3267     }
3268 }
3269 } {
3270     \tl_put_right:Nx \l_tmpa_tl {
3271         \stex_term_arg:nn { \int_eval:n { #1 } }
3272         \exp_not:n { { #2 } }
3273     }
3274 }
3275
3276 \__stex_terms_custom_loop:
3277 }

```

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

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

```

3278 \cs_new_protected:Nn \__stex_terms_custom_set_X:n {
3279     \str_set:Nx \l_tmpa_str {
3280         \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
3281         X
3282         \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
3283     }
3284 }

```

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

\\_\_stex\_terms\_custom\_component:

```

3285 \cs_new_protected:Npn \__stex_terms_custom_component:w [ #1 ] {
3286     \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
3287     \__stex_terms_custom_loop:
3288 }

```

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

\\_\_stex\_terms\_custom\_final:

```

3289 \cs_new_protected:Nn \__stex_terms_custom_final: {
3290     \int_compare:nNnTF \l_tmpb_int = 0 {
3291         \exp_args:Nnno \stex_term_oms:nnn
3292     }{
3293         \str_if_in:NnTF \l_tmpa_str {b} {
3294             \exp_args:Nnno \stex_term_ombind:nnn
3295         } {
3296             \exp_args:Nnno \stex_term_oma:nnn
3297         }
3298     }
3299     { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
3300 }

```

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

**\symref**

**\symname**

```

3301 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3302
3303 \keys_define:nn { stex / symname } {

```



```

3304 pre .tl_set_x:N = \l__stex_terms_pre_tl ,
3305 post .tl_set_x:N = \l__stex_terms_post_tl ,
3306 root .tl_set_x:N = \l__stex_terms_root_tl
3307 }
3308
3309 \cs_new_protected:Nn \stex_symname_args:n {
3310 \tl_clear:N \l__stex_terms_post_tl
3311 \tl_clear:N \l__stex_terms_pre_tl
3312 \tl_clear:N \l__stex_terms_root_str
3313 \keys_set:nn { stex / symname } { #1 }
3314 }
3315
3316 \NewDocumentCommand \symref { m m }{
3317 \let\compemph_uri_prev:\compemph@uri
3318 \let\compemph@uri\symrefemph@uri
3319 \STEXsymbol{#1}!{ #2 }
3320 \let\compemph@uri\compemph_uri_prev:
3321 }
3322
3323 \NewDocumentCommand \synonym { 0{ } m m }{
3324 \stex_symname_args:n { #1 }
3325 \let\compemph_uri_prev:\compemph@uri
3326 \let\compemph@uri\symrefemph@uri
3327 % TODO
3328 \STEXsymbol{#2}!\{ \l__stex_terms_pre_tl #3 \l__stex_terms_post_tl }
3329 \let\compemph@uri\compemph_uri_prev:
3330 }
3331
3332 \NewDocumentCommand \symname { 0{ } m }{
3333 \stex_symname_args:n { #1 }
3334 \stex_get_symbol:n { #2 }
3335 \str_set:Nx \l_tmpa_str {
3336 \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3337 }
3338 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3339
3340 \let\compemph_uri_prev:\compemph@uri
3341 \let\compemph@uri\symrefemph@uri
3342 \exp_args:NNx \use:nn
3343 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!{
3344 \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
3345 } }
3346 \let\compemph@uri\compemph_uri_prev:
3347 }
3348
3349 \NewDocumentCommand \Symname { 0{ } m }{
3350 \stex_symname_args:n { #1 }
3351 \stex_get_symbol:n { #2 }
3352 \str_set:Nx \l_tmpa_str {
3353 \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3354 }
3355 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3356 \let\compemph_uri_prev:\compemph@uri
3357 \let\compemph@uri\symrefemph@uri

```

```

3358 \exp_args:Nnx \use:nn
3359 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!{
3360   \exp_after:wN \stex_capitalize:n \l_tmpa_str
3361   \l__stex_terms_post_tl
3362 } }
3363 \let\compemph@uri\compemph_uri_prev:
3364 }

```

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

## 31.3 Notation Components

```

3365 <@@=stex_notationcomps>

```

`\stex_highlight_term:nn`

```

3366
3367 \str_new:N \l_stex_current_symbol_str
3368 \cs_new_protected:Nn \stex_highlight_term:nn {
3369   \exp_args:Nnx
3370   \use:nn {
3371     \str_set:Nx \l_stex_current_symbol_str { #1 }
3372     #2
3373   } {
3374     \str_set:Nx \exp_not:N \l_stex_current_symbol_str
3375     { \l_stex_current_symbol_str }
3376   }
3377 }
3378
3379 \cs_new_protected:Nn \stex_unhighlight_term:n {
3380   % \latexml_if:TF {
3381   %   #1
3382   % } {
3383   %   \rustex_if:TF {
3384   %     #1
3385   %   } {
3386     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
3387   % }
3388   % }
3389 }

```

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

```

\comp
\compemph@uri
\compemph
\defemph
\defemph@uri
\symrefemph
\symrefemph@uri
3390 \cs_new_protected:Npn \comp #1 {
3391   \str_if_empty:NF \l_stex_current_symbol_str {
3392     \rustex_if:TF {
3393       \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
3394     }{
3395       \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3396     }
3397   }
3398 }
3399
3400 \cs_new_protected:Npn \compemph@uri #1 #2 {

```

```

3401     \compemph{ #1 }
3402 }
3403
3404
3405 \cs_new_protected:Npn \compemph #1 {
3406     #1
3407 }
3408
3409 \cs_new_protected:Npn \defemph@uri #1 #2 {
3410     \defemph{#1}
3411 }
3412
3413 \cs_new_protected:Npn \defemph #1 {
3414     \textbf{#1}
3415 }
3416
3417 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3418     \symrefemph{#1}
3419 }
3420
3421 \cs_new_protected:Npn \symrefemph #1 {
3422     \textbf{#1}
3423 }

```

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

## `\ellipses`

```

3424 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix
\parrayline
\parraylineh
\parraycell
3425 \bool_new:N \l_stex_inarray_bool
3426 \bool_set_false:N \l_stex_inarray_bool
3427 \NewDocumentCommand \parray { m m } {
3428     \begingroup
3429     \bool_set_true:N \l_stex_inarray_bool
3430     \begin{array}{#1}
3431         #2
3432     \end{array}
3433 \endgroup
3434 }
3435
3436 \NewDocumentCommand \prmatrix { m } {
3437     \begingroup
3438     \bool_set_true:N \l_stex_inarray_bool
3439     \begin{matrix}
3440         #1
3441     \end{matrix}
3442 \endgroup
3443 }
3444
3445 \def \maybepline {
3446     \bool_if:NT \l_stex_inarray_bool {\hline}
3447 }

```

```

3448
3449 \def \parrayline #1 #2 {
3450   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3451 }
3452
3453 \def \pmrow #1 { \parrayline{}{ #1 } }
3454
3455 \def \parraylineh #1 #2 {
3456   #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3457 }
3458
3459 \def \parraycell #1 {
3460   #1 \bool_if:NT \l_stex_inarray_bool {&}
3461 }

```

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

## 31.4 Variables

```

3462 <@@=stex_variables>

```

`\stex_invoke_variable:n` Invokes a variable

```

3463 \cs_new_protected:Nn \stex_invoke_variable:n {
3464   \if_mode_math:
3465     \exp_after:wN \__stex_variables_invoke_math:n
3466   \else:
3467     \exp_after:wN \__stex_variables_invoke_text:n
3468   \fi: {#1}
3469 }
3470
3471 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
3472   %TODO
3473 }
3474
3475
3476 \cs_new_protected:Nn \__stex_variables_invoke_math:n {
3477   \peek_charcode_remove:NTF ! {
3478     \peek_charcode_remove:NTF ! {
3479       \peek_charcode:NTF [ {
3480         \__stex_variables_invoke_op_custom:nw
3481       }{
3482         % TODO throw error
3483       }
3484     }{
3485       \__stex_variables_invoke_op:n { #1 }
3486     }
3487   }{
3488     \peek_charcode_remove:NTF * {
3489       \__stex_variables_invoke_text:n { #1 }
3490     }{
3491       \__stex_variables_invoke_math_ii:n { #1 }
3492     }
3493   }
3494 }

```

```

3495
3496 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
3497   \cs_if_exist:cTF {
3498     stex_var_op_notation_ #1 _cs
3499   }{
3500     \use:c{stex_var_op_notation_ #1 _cs }
3501   }{
3502     \msg_error:nnxx{stex}{error/noop}{variable~#1}{}
3503   }
3504 }
3505
3506 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n #1 {
3507   \cs_if_exist:cTF {
3508     stex_var_notation_#1_cs
3509   }{
3510     \str_set:Nn \l_stex_current_symbol_str { #1 }
3511     \use:c{stex_var_notation_#1_cs}
3512   }{
3513     \msg_error:nnxx{stex}{error/nonotation}{variable~#1}{s}
3514   }
3515 }

```

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

```

3516 \endpackage

```

## Chapter 32

# STEX -Structural Features Implementation

```
3517 ⟨*package⟩
3518
3519 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3520
3521 ⟨@@=stex_features⟩
3522
3523   Warnings and error messages
3524   \msg_new:nnn{stex}{error/copymodule/notallowed}{
3525     Symbol~#1~can~not~be~assigned~in~copymodule~#2
3526   }
3527   \msg_new:nnn{stex}{error/interpretmodule/noddefinens}{
3528     Symbol~#1~not~assigned~in~interpretmodule~#2
3529   }
```

### 32.1 Imports with modification

```
3529 \cs_new_protected:Nn \stex_get_symbol_in_copymodule:n {
3530   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3531     \__stex_features_get_symbol_from_cs:n { #1 }
3532   }{
3533     % argument is a string
3534     % is it a command name?
3535     \cs_if_exist:cTF { #1 }{
3536       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3537       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3538       \str_if_empty:NNTF \l_tmpa_str {
3539         \exp_args:Nx \cs_if_eq:NNTF {
3540           \tl_head:N \l_tmpa_tl
3541         } \stex_invoke_symbol:n {
3542           \exp_args:No \__stex_features_get_symbol_from_cs:n { \use:c { #1 } }
3543         }{
3544           \__stex_features_get_symbol_from_string:n { #1 }
```

```

3545     }
3546   } {
3547     \__stex_features_get_symbol_from_string:n { #1 }
3548   }
3549   ){
3550     % argument is not a command name
3551     \__stex_features_get_symbol_from_string:n { #1 }
3552     % \l_stex_all_symbols_seq
3553   }
3554 }
3555 }
3556
3557 \cs_new_protected:Nn \__stex_features_get_symbol_from_string:n {
3558   \str_set:Nn \l_tmpa_str { #1 }
3559   \bool_set_false:N \l_tmpa_bool
3560   \bool_if:NF \l_tmpa_bool {
3561     \tl_set:Nn \l_tmpa_tl {
3562       \msg_set:nnn{stex}{error/unknownsymbol}{
3563         No~symbol~#1~found!
3564       }
3565       \msg_error:nn{stex}{error/unknownsymbol}
3566     }
3567     \str_set:Nn \l_tmpa_str { #1 }
3568     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3569     \seq_map_inline:Nn \l__stex_features_copymodule_fields_seq {
3570       \str_set:Nn \l_tmpb_str { ##1 }
3571       \str_if_eq:eeT { \l_tmpa_str } {
3572         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3573       } {
3574         \seq_map_break:n {
3575           \tl_set:Nn \l_tmpa_tl {
3576             \str_set:Nn \l_stex_get_symbol_uri_str {
3577               ##1
3578             }
3579             \__stex_features_get_symbol_check:
3580           }
3581         }
3582       }
3583     }
3584     \l_tmpa_tl
3585   }
3586 }
3587
3588 \cs_new_protected:Nn \__stex_features_get_symbol_from_cs:n {
3589   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3590   { \tl_tail:N \l_tmpa_tl }
3591   \tl_if_single:NTF \l_tmpa_tl {
3592     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3593       \exp_after:wN \str_set:Nn \exp_after:wN
3594       \l_stex_get_symbol_uri_str \l_tmpa_tl
3595       \__stex_features_get_symbol_check:
3596     }{
3597       % TODO
3598       % tail is not a single group

```

```

3599     }
3600   }{
3601     % TODO
3602     % tail is not a single group
3603   }
3604 }
3605
3606 \cs_new_protected:Nn \__stex_features_get_symbol_check: {
3607   \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq {?} \l_stex_get_symbol_uri_str
3608   \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 3 {
3609     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
3610     \str_set:Nx \l_tmpa_str {\seq_use:Nn \l_tmpa_seq ?}
3611     \seq_if_in:NoF \l__stex_features_copymodule_modules_seq \l_tmpa_str {
3612       \msg_error:nxxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3613         \l_stex_current_copymodule_name_str\Allowed:~\seq_use:Nn \l__stex_features_copymodule_modules_seq \l_tmpa_str
3614       }
3615     }
3616   }{
3617     \msg_error:nxxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3618       \l_stex_current_copymodule_name_str~(inexplicably)
3619     }
3620   }
3621 }
3622
3623 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
3624   \stex_import_module_uri:nn { #1 } { #2 }
3625   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3626   \stex_import_require_module:nnnn
3627   { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3628   { \l_stex_import_path_str } { \l_stex_import_name_str }
3629   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3630   \seq_set_eq:NN \l__stex_features_copymodule_modules_seq \l_stex_collect_imports_seq
3631   \seq_clear:N \l__stex_features_copymodule_fields_seq
3632   \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3633     \seq_map_inline:cn {c_stex_module_###1_constants}{
3634       \exp_args:NNx \seq_put_right:Nn \l__stex_features_copymodule_fields_seq {
3635         ###1 ? #####1
3636       }
3637     }
3638   }
3639   \seq_clear:N \l_tmpa_seq
3640   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3641     name      = \l_stex_current_copymodule_name_str ,
3642     module    = \l_stex_current_module_str ,
3643     from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3644     includes  = \l_tmpa_seq ,
3645     fields    = \l_tmpa_seq
3646   }
3647   \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3648     as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3649   \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_features_copymodule_modules_seq
3650   \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_features_copymodule_fields_seq {,~}
3651   \stex_if_smsmode:F {
3652     \begin{stex_annotate_env} {#4} {

```



```

3653     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3654   }
3655   \stex_annotate_invisible:nnn{from}{\l_stex_import_ns_str ?\l_stex_import_name_str}{\}
3656 }
3657 \bool_set_eq:NN \l__stex_features_oldhtml_bool \stex_html_do_output_bool
3658 \bool_set_false:N \stex_html_do_output_bool
3659 }
3660 \cs_new_protected:Nn \stex_copymodule_end:n {
3661   \def \l_tmpa_cs ##1 ##2 {#1}
3662   \bool_set_eq:NN \stex_html_do_output_bool \l__stex_features_oldhtml_bool
3663   \tl_clear:N \l_tmpa_tl
3664   \tl_clear:N \l_tmpb_tl
3665   \prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3666   \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3667     \seq_map_inline:cn {c_stex_module_##1_constants}{
3668       \tl_clear:N \l_tmpc_tl
3669       \l_tmpa_cs{##1}{####1}
3670       \str_if_exist:cTF {\l__stex_features_copymodule_##1?####1_name_str} {
3671         \tl_put_right:Nx \l_tmpa_tl {
3672           \prop_set_from_keyval:cn {
3673             l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_#
3674           }{
3675             \exp_after:wN \prop_to_keyval:N \csname
3676               l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_#
3677             \endcsname
3678           }
3679           \seq_clear:c {
3680             l_stex_symdecl_
3681             \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_name_s
3682             _notations
3683           }
3684         }
3685         \tl_put_right:Nx \l_tmpc_tl {
3686           \stex_copy_notations:nn {\l_stex_current_module_str ? \use:c{l__stex_features_copy
3687           \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_features_copymodule_##1?####1_n
3688         }
3689         \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{l__stex_features_
3690         \str_if_exist:cT {\l_stex_features_copymodule_##1?####1_macroname_str} {
3691           \tl_put_right:Nx \l_tmpc_tl {
3692             \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?#
3693           }
3694           \tl_put_right:Nx \l_tmpa_tl {
3695             \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3696             \stex_invoke_symbol:n {
3697               \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_na
3698             }
3699           }
3700         }
3701       }
3702     }{
3703       \tl_put_right:Nx \l_tmpc_tl {
3704         \stex_copy_notations:nn {\l_stex_current_module_str ? \l_stex_current_copymodule_n
3705       }
3706       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1_prop}

```

```

3707 \prop_put:Nnx \l_tmpa_prop { name }{ \l_stex_current_copymodule_name_str / #####1 }
3708 \prop_put:Nnx \l_tmpa_prop { module }{ \l_stex_current_module_str }
3709 \tl_put_right:Nx \l_tmpa_tl {
3710   \prop_set_from_keyval:cn {
3711     l_stex_symdecl\l_stex_current_module_str ? \l_stex_current_copymodule_name_str
3712   }{
3713     \prop_to_keyval:N \l_tmpa_prop
3714   }
3715   \seq_clear:c {
3716     l_stex_symdecl_
3717     \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / #####1
3718     _notations
3719   }
3720 }
3721 \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str}
3722 \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3723   \tl_put_right:Nx \l_tmpc_tl {
3724     \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}
3725   }
3726   \tl_put_right:Nx \l_tmpa_tl {
3727     \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3728       \stex_invoke_symbol:n {
3729         \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / #####1
3730       }
3731     }
3732   }
3733 }
3734 }
3735 \tl_if_exist:cT {l__stex_features_copymodule_##1?####1_def_tl}{
3736   \tl_put_right:Nx \l_tmpc_tl {
3737     \stex_annotate_invisible:nnn{definiens}{\use:c{l__stex_features_copymodule_##1?####1_def_tl}}
3738   }
3739 }
3740 \tl_put_right:Nx \l_tmpb_tl {
3741   \stex_annotate:nnn{assignment} {##1?####1} { \l_tmpc_tl }
3742 }
3743 }
3744 }
3745 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3746 \tl_put_left:Nx \l_tmpa_tl {
3747   \prop_set_from_keyval:cn {
3748     l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _prop
3749   }{
3750     \prop_to_keyval:N \l_stex_current_copymodule_prop
3751   }
3752 }
3753 \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl
3754 \stex_debug:nn{copymodule}{result:\meaning \l_tmpa_tl}
3755 \exp_args:Nx \stex_do_up_to_module:n {
3756   \exp_args:No \exp_not:n \l_tmpa_tl
3757 }
3758 \l_tmpb_tl
3759 \stex_if_smsmode:F {
3760   \end{stex_annotate_env}

```

```

3761 }
3762 }
3763
3764 \NewDocumentEnvironment {copymodule} { 0{} m m}{
3765   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ structure }
3766   \stex_deactivate_macro:Nn \symdecl {module~environments}
3767   \stex_deactivate_macro:Nn \symdef {module~environments}
3768   \stex_deactivate_macro:Nn \notation {module~environments}
3769   \stex_reactivate_macro:N \assign
3770   \stex_reactivate_macro:N \renamedec1
3771   \stex_reactivate_macro:N \donotcopy
3772   \stex_smsmode_do:
3773 }{
3774   \stex_copymodule_end:n {}
3775 }
3776
3777 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
3778   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ realization }
3779   \stex_deactivate_macro:Nn \symdecl {module~environments}
3780   \stex_deactivate_macro:Nn \symdef {module~environments}
3781   \stex_deactivate_macro:Nn \notation {module~environments}
3782   \stex_reactivate_macro:N \assign
3783   \stex_reactivate_macro:N \renamedec1
3784   \stex_reactivate_macro:N \donotcopy
3785   \stex_smsmode_do:
3786 }{
3787   \stex_copymodule_end:n {
3788     \tl_if_exist:cF {
3789       l__stex_features_copymodule_##1?##2_def_tl
3790     }{
3791       \msg_error:nxxx{stex}{error/interpretmodule/noddefinies}{
3792         ##1?##2
3793       }{\l_stex_current_copymodule_name_str}
3794     }
3795   }
3796 }
3797
3798 \NewDocumentCommand \donotcopy { 0{} m}{
3799   \stex_import_module_uri:nn { #1 } { #2 }
3800   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3801   \seq_map_inline:Nn \l_stex_collect_imports_seq {
3802     \seq_remove_all:Nn \l_stex_features_copymodule_modules_seq { ##1 }
3803     \seq_map_inline:cn {c_stex_module_##1_constants}{
3804       \seq_remove_all:Nn \l_stex_features_copymodule_fields_seq { ##1 ? #####1 }
3805       \bool_lazy_any_p:nT {
3806         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_name_str}}
3807         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_macroname_str}}
3808         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_def_tl}}
3809       }{
3810         % TODO throw error
3811       }
3812     }
3813   }
3814 }

```

```

3815 \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
3816 \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_ns_str ?\l_stex_import_name_str }
3817 \prop_put:Nnx \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
3818 }
3819
3820 \NewDocumentCommand \assign { m m }{
3821   \stex_get_symbol_in_copymodule:n {#1}
3822   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
3823   \tl_set:cn {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _def_tl}{#2}
3824 }
3825
3826 \keys_define:nn { stex / renamedec1 } {
3827   name .str_set_x:N = \l_stex_renamedec1_name_str
3828 }
3829 \cs_new_protected:Nn \__stex_features_renamedec1_args:n {
3830   \str_clear:N \l_stex_renamedec1_name_str
3831
3832   \keys_set:nn { stex / renamedec1 } { #1 }
3833 }
3834
3835 \NewDocumentCommand \renamedec1 { 0{} m m }{
3836   \__stex_features_renamedec1_args:n { #1 }
3837   \stex_get_symbol_in_copymodule:n {#2}
3838   \stex_debug:nn{renamedec1}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
3839   \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
3840   \str_if_empty:NTF \l_stex_renamedec1_name_str {
3841     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3842       \l_stex_get_symbol_uri_str
3843     } }
3844   } {
3845     \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex_r
3846     \stex_debug:nn{renamedec1}{@~\l_stex_current_module_str ? \l_stex_renamedec1_name_str}
3847     \prop_set_eq:cc {l_stex_symdecl_
3848       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3849     _prop
3850     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
3851     \seq_set_eq:cc {l_stex_symdecl_
3852       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3853     _notations
3854     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
3855     \prop_put:cnx {l_stex_symdecl_
3856       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3857     _prop
3858     }{ name }{ \l_stex_renamedec1_name_str }
3859     \prop_put:cnx {l_stex_symdecl_
3860       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3861     _prop
3862     }{ module }{ \l_stex_current_module_str }
3863     \exp_args:NNx \seq_put_left:Nn \l__stex_features_copymodule_fields_seq {
3864       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3865     }
3866     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3867       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3868     } }

```

```

3869 }
3870 }
3871 %\NewDocumentCommand \notation_in_copymodules: { 0{} m } {
3872 % \_stex_notation_args:n { #1 }
3873 % \tl_clear:N \l_stex_symdecl_definiens_tl
3874 % \stex_get_symbol_in_copymodule:n { #2 }
3875 % \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
3876 % % todo
3877 %}
3878 \stex_deactivate_macro:Nn \assign {copymodules}
3879 \stex_deactivate_macro:Nn \renamedekl {copymodules}
3880 \stex_deactivate_macro:Nn \donotcopy {copymodules}
3881
3882
3883 \seq_new:N \l_stex_implicit_morphisms_seq
3884 \NewDocumentCommand \implicitmorphism { 0{} m m }{
3885 \stex_import_module_uri:nn { #1 } { #2 }
3886 \stex_debug:nn{implicits}{
3887 Implicit~morphism:~
3888 \l_stex_module_ns_str ? \l__stex_features_name_str
3889 }
3890 \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
3891 \l_stex_module_ns_str ? \l__stex_features_name_str
3892 }{
3893 \msg_error:nnn{stex}{error/conflictingmodules}{
3894 \l_stex_module_ns_str ? \l__stex_features_name_str
3895 }
3896 }
3897
3898 % TODO
3899
3900
3901
3902 \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
3903 \l_stex_module_ns_str ? \l__stex_features_name_str
3904 }
3905 }
3906

```

## 32.2 The feature environment

structural@feature

```

3907
3908 \NewDocumentEnvironment{structural@feature}{ m m m }{
3909 \stex_if_in_module:F {
3910 \msg_set:nnn{stex}{error/nomodule}{
3911 Structural~Feature~has~to~occur~in~a~module:\\
3912 Feature~#2~of~type~#1\\
3913 In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
3914 }
3915 \msg_error:nn{stex}{error/nomodule}
3916 }
3917

```

```

3918 \str_set:Nx \l_stex_module_name_str {
3919   \prop_item:Nn \l_stex_current_module_prop
3920     { name } / #2 - feature
3921 }
3922
3923 \str_set:Nx \l_stex_module_ns_str {
3924   \prop_item:Nn \l_stex_current_module_prop
3925     { ns }
3926 }
3927
3928
3929 \str_clear:N \l_tmpa_str
3930 \seq_clear:N \l_tmpa_seq
3931 \tl_clear:N \l_tmpa_tl
3932 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
3933   origname = #2,
3934   name     = \l_stex_module_name_str ,
3935   ns       = \l_stex_module_ns_str ,
3936   imports  = \exp_not:o { \l_tmpa_seq } ,
3937   constants = \exp_not:o { \l_tmpa_seq } ,
3938   content  = \exp_not:o { \l_tmpa_tl } ,
3939   file     = \exp_not:o { \g_stex_currentfile_seq } ,
3940   lang     = \l_stex_module_lang_str ,
3941   sig      = \l_tmpa_str ,
3942   meta     = \l_tmpa_str ,
3943   feature  = #1 ,
3944 }
3945
3946 \stex_if_smsmode:F {
3947   \begin{stex_annotate_env}{ feature:#1 }{}
3948     \stex_annotate_invisible:nnn{header}{}{ #3 }
3949 }
3950 }{
3951   \str_set:Nx \l_tmpa_str {
3952     c_stex_feature_
3953     \prop_item:Nn \l_stex_current_module_prop { ns } ?
3954     \prop_item:Nn \l_stex_current_module_prop { name }
3955     _prop
3956   }
3957   \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
3958   \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
3959   \stex_if_smsmode:F {
3960     \end{stex_annotate_env}
3961   }
3962 }
3963

```

## 32.3 Features

structure

```

3964
3965 \prop_new:N \l_stex_all_structures_prop
3966

```

```

3967 \keys_define:nn { stex / features / structure } {
3968   name          .str_set_x:N = \l__stex_features_structure_name_str ,
3969 }
3970
3971 \cs_new_protected:Nn \__stex_features_structure_args:n {
3972   \str_clear:N \l__stex_features_structure_name_str
3973   \keys_set:nn { stex / features / structure } { #1 }
3974 }
3975
3976 %\stex_new_feature:nnnn { structure } { 0{ } m } {
3977 %   \__stex_features_structure_args:n { ##1 }
3978 %   \str_if_empty:NT \l__stex_features_structure_name_str {
3979 %     \str_set:Nx \l__stex_features_structure_name_str { ##2 }
3980 %   }
3981 % } {
3982 %
3983 %}
3984
3985 \NewDocumentEnvironment{mathstructure}{ 0{ } m }{
3986   \__stex_features_structure_args:n { #1 }
3987   \str_if_empty:NT \l__stex_features_structure_name_str {
3988     \str_set:Nx \l__stex_features_structure_name_str { #2 }
3989   }
3990   \exp_args:Nnnx
3991   \begin{structural@feature}{ structure }
3992     { \l__stex_features_structure_name_str }{ }
3993     \seq_clear:N \l_tmpa_seq
3994     \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
3995     \stex_smsmode_do:
3996   }{
3997     \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
3998     \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
3999     \str_set:Nx \l_tmpa_str {
4000       \prop_item:Nn \l_stex_current_module_prop { ns } ?
4001       \prop_item:Nn \l_stex_current_module_prop { name }
4002     }
4003     \seq_map_inline:Nn \l_tmpa_seq {
4004       \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
4005     }
4006     \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
4007     \exp_args:Nnx
4008     \AddToHookNext { env / mathstructure / after }{
4009       \symdecl{ #2 }[type = \exp_not:N\collection,def={\STEXsymbol{module-type}}{
4010         \stex_term_math_oms:nnnn { \l_tmpa_str }{ }{0}{ }
4011       }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]
4012       \STEXexport {
4013         \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
4014           { \prop_item:Nn \l_stex_current_module_prop { origname } }
4015           { \l_tmpa_str }
4016         \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
4017           { #2 }{ \l_tmpa_str }
4018       %   \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
4019       %     \prop_item:Nn \l_stex_current_module_prop { origname },
4020       %     \l_tmpa_str

```

```

4021 %     }
4022 %     \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
4023 %         #2,\l_tmpa_str
4024 %     }
4025 %     \tl_set:cx { #2 } {
4026 %         \stex_invoke_structure:n { \l_tmpa_str }
4027 %     }
4028 % }
4029
4030 \end{structural@feature}
4031 % \g_stex_last_feature_prop
4032 }

```

\instantiate

```

4033 \seq_new:N \l__stex_features_structure_field_seq
4034 \str_new:N \l__stex_features_structure_field_str
4035 \str_new:N \l__stex_features_structure_def_tl
4036 \prop_new:N \l__stex_features_structure_prop
4037 \NewDocumentCommand \instantiate { m O{} m }{
4038   \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
4039   \prop_set_eq:Nc \l__stex_features_structure_prop {
4040     c_stex_feature_\l_tmpa_str _prop
4041   }
4042   \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
4043   \seq_map_inline:Nn \l__stex_features_structure_field_seq {
4044     \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
4045     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
4046       \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
4047       \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
4048         {!} \l_tmpa_tl
4049       \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
4050         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
4051         \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
4052         \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
4053       }{
4054         \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
4055         \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
4056         \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
4057           \l_tmpa_tl
4058         \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
4059           \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
4060           \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
4061         }{
4062           \tl_clear:N \l_tmpb_tl
4063         }
4064       }
4065     }{
4066       \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
4067       \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
4068         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
4069         \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
4070         \tl_clear:N \l_tmpa_tl
4071       }{
4072         % TODO throw error

```



```

4073     }
4074   }
4075   % \l_tmpa_str: name
4076   % \l_tmpa_tl: definiens
4077   % \l_tmpb_tl: notation
4078   \tl_if_empty:NT \l__stex_features_structure_field_str {
4079     % TODO throw error
4080   }
4081   \str_clear:N \l_tmpb_str
4082
4083   \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
4084   \seq_map_inline:Nn \l_tmpa_seq {
4085     \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
4086     \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
4087     \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
4088       \seq_map_break:n {
4089         \str_set:Nn \l_tmpb_str { ####1 }
4090       }
4091     }
4092   }
4093   \prop_get:cnN { l_stex_symdecl_ \l_tmpb_str _prop } {args}
4094   \l_tmpb_str
4095
4096   \tl_if_empty:NTF \l_tmpb_tl {
4097     \tl_if_empty:NF \l_tmpa_tl {
4098       \exp_args:Nx \use:n {
4099         \symdecl{#3/\l__stex_features_structure_field_str}[args=\l_tmpb_str,def={\exp_args:
4100       }
4101     }
4102   }{
4103     \tl_if_empty:NTF \l_tmpa_tl {
4104       \exp_args:Nx \use:n {
4105         \symdef{#3/\l__stex_features_structure_field_str}[args=\l_tmpb_str]\exp_after:wN\
4106       }
4107     }{
4108       \exp_args:Nx \use:n {
4109         \symdef{#3/\l__stex_features_structure_field_str}[args=\l_tmpb_str,def={\exp_args:
4110         \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
4111       }
4112     }
4113   }
4114 }
4115 % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
4116 % \prop_item:Nn \l_stex_current_module_prop {name} ?
4117 % #3/\l__stex_features_structure_field_str
4118 % \par
4119 % \expandafter\present\csname
4120 %   l_stex_symdecl_
4121 %   \prop_item:Nn \l_stex_current_module_prop {ns} ?
4122 %   \prop_item:Nn \l_stex_current_module_prop {name} ?
4123 %   #3/\l__stex_features_structure_field_str
4124 %   _prop
4125 % \endcsname
4126 }

```

```

4127
4128 \tl_clear:N \l__stex_features_structure_def_tl
4129
4130 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
4131 \seq_map_inline:Nn \l_tmpa_seq {
4132   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
4133   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
4134   \exp_args:Nx \use:n {
4135     \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
4136
4137     }
4138   }
4139
4140   \prop_if_exist:cF {
4141     l_stex_symdecl_
4142     \prop_item:Nn \l_stex_current_module_prop {ns} ?
4143     \prop_item:Nn \l_stex_current_module_prop {name} ?
4144     #3/\l_tmpa_str
4145     _prop
4146   }{
4147     \prop_get:cnN { l_stex_symdecl_ ##1 _prop } {args}
4148     \l_tmpb_str
4149     \exp_args:Nx \use:n {
4150       \symdecl{#3/\l_tmpa_str}[args=\l_tmpb_str]
4151     }
4152   }
4153 }
4154
4155 \symdecl*{#3}[type={\STEXsymbol{module-type}}{
4156   \stex_term_math_oms:nnnn {
4157     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
4158     \prop_item:Nn \l__stex_features_structure_prop {name}
4159   }{0}{0}}
4160 }]}
4161
4162 % TODO: -> sms file
4163
4164 \tl_set:cx{ #3 }{
4165   \stex_invoke_structure:nnn {
4166     \prop_item:Nn \l_stex_current_module_prop {ns} ?
4167     \prop_item:Nn \l_stex_current_module_prop {name} ? #3
4168   } {
4169     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
4170     \prop_item:Nn \l__stex_features_structure_prop {name}
4171   }
4172 }
4173 \stex_smsmode_do:
4174 }

```

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

\stex\_invoke\_structure:nnn

```

4175 % #1: URI of the instance
4176 % #2: URI of the instantiated module

```

```

4177 \cs_new_protected:Nn \stex_invoke_structure:nnn {
4178   \tl_if_empty:nTF{ #3 }{
4179     \prop_set_eq:Nc \l__stex_features_structure_prop {
4180       c_stex_feature_ #2 _prop
4181     }
4182     \tl_clear:N \l_tmpa_tl
4183     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
4184     \seq_map_inline:Nn \l_tmpa_seq {
4185       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
4186       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
4187       \cs_if_exist:cT {
4188         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
4189       }{
4190         \tl_if_empty:NF \l_tmpa_tl {
4191           \tl_put_right:Nn \l_tmpa_tl {,}
4192         }
4193         \tl_put_right:Nx \l_tmpa_tl {
4194           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
4195         }
4196       }
4197     }
4198     \exp_args:No \mathstruct \l_tmpa_tl
4199   }{
4200     \stex_invoke_symbol:n{#1/#3}
4201   }
4202 }

```

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

```

4203 </package>

```

## Chapter 33

# STEX -Statements Implementation

```
4204 <*package>
4205
4206 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4207
4208 <@@=stex_statements>
      Warnings and error messages
4209

\titleemph
4210 \def\titleemph#1{\textbf{#1}}

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

### 33.1 Definitions

```
definiendum

4211 \keys_define:nn {stex / definiendum }{
4212   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
4213   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
4214   root     .str_set_x:N    = \l__stex_statements_definiendum_root_str,
4215   gfa      .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
4216 }
4217 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
4218   \str_clear:N \l__stex_statements_definiendum_root_str
4219   \tl_clear:N \l__stex_statements_definiendum_post_tl
4220   \str_clear:N \l__stex_statements_definiendum_gfa_str
4221   \keys_set:nn { stex / definiendum }{ #1 }
4222 }
4223 \NewDocumentCommand \definiendum { O{} m m } {
4224   \__stex_statements_definiendum_args:n { #1 }
4225   \stex_get_symbol:n { #2 }
4226   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4227   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
4228     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```

4229     \tl_set:Nn \l_tmpa_tl { #3 }
4230   } {
4231     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
4232     \tl_set:Nn \l_tmpa_tl {
4233       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
4234     }
4235   }
4236 } {
4237   \tl_set:Nn \l_tmpa_tl { #3 }
4238 }
4239
4240 % TODO root
4241 \rustex_if:TF {
4242   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
4243 } {
4244   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
4245 }
4246 }
4247 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

#### definame

```

4248
4249 \NewDocumentCommand \definame { 0{ } m } {
4250   \__stex_statements_definiendum_args:n { #1 }
4251   % TODO: root
4252   \stex_get_symbol:n { #2 }
4253   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4254   \str_set:Nx \l_tmpa_str {
4255     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4256   }
4257   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4258   \rustex_if:TF {
4259     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4260       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4261     }
4262   } {
4263     \defemph@uri {
4264       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4265     } { \l_stex_get_symbol_uri_str }
4266   }
4267 }
4268 \stex_deactivate_macro:Nn \definame {definition~environments}
4269
4270 \NewDocumentCommand \Definame { 0{ } m } {
4271   \__stex_statements_definiendum_args:n { #1 }
4272   \stex_get_symbol:n { #2 }
4273   \str_set:Nx \l_tmpa_str {
4274     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4275   }
4276   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4277   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4278   \rustex_if:TF {

```

```

4279 \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4280 \l_tmpa_str\l__stex_statements_definiendum_post_tl
4281 }
4282 } {
4283 \defemph@uri {
4284 \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
4285 } { \l_stex_get_symbol_uri_str }
4286 }
4287 }
4288 \stex_deactivate_macro:Nn \Definame {definition~environments}
4289
4290 \NewDocumentCommand \premise { m }{
4291 \stex_annotate:nnn{ premise }{}{ #1 }
4292 }
4293 \NewDocumentCommand \conclusion { m }{
4294 \stex_annotate:nnn{ conclusion }{}{ #1 }
4295 }
4296 \NewDocumentCommand \definiens { m }{
4297 \stex_annotate:nnn{ definiens }{}{ #1 }
4298 }
4299
4300 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
4301 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
4302 \stex_deactivate_macro:Nn \definiens {definition~environments}
4303

```

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

## sdefinition

```

4304
4305 \keys_define:nn {stex / sdefinition }{
4306 type .str_set_x:N = \sdefinitiontype,
4307 id .str_set_x:N = \sdefinitionid,
4308 name .str_set_x:N = \sdefinitionname,
4309 for .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4310 title .tl_set:N = \sdefinitiontitle
4311 }
4312 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
4313 \str_clear:N \sdefinitiontype
4314 \str_clear:N \sdefinitionid
4315 \str_clear:N \sdefinitionname
4316 \clist_clear:N \l__stex_statements_sdefinition_for_clist
4317 \tl_clear:N \sdefinitiontitle
4318 \keys_set:nn { stex / sdefinition }{}{ #1 }
4319 }
4320
4321 \NewDocumentEnvironment{sdefinition}{0{}}{
4322 \__stex_statements_sdefinition_args:n{ #1 }
4323 \stex_reactivate_macro:N \definiendum
4324 \stex_reactivate_macro:N \definame
4325 \stex_reactivate_macro:N \Definame
4326 \stex_reactivate_macro:N \premise
4327 \stex_reactivate_macro:N \definiens
4328 \stex_if_smsmode:F{

```

```

4329 \seq_clear:N \l_tmpa_seq
4330 \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4331   \tl_if_empty:nF{ ##1 }{
4332     \stex_get_symbol:n { ##1 }
4333     \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4334       \l_stex_get_symbol_uri_str
4335     }
4336   }
4337 }
4338 \exp_args:Nnnx
4339 \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {,}}
4340 \str_if_empty:NF \sdefinitiontype {
4341   \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{}
4342 }
4343 \clist_set:No \l_tmpa_clist \sdefinitiontype
4344 \tl_clear:N \l_tmpa_tl
4345 \clist_map_inline:Nn \l_tmpa_clist {
4346   \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
4347     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
4348   }
4349 }
4350 \tl_if_empty:NTF \l_tmpa_tl {
4351   \__stex_statements_sdefinition_start:
4352 }{
4353   \l_tmpa_tl
4354 }
4355 }
4356 \stex_ref_new_doc_target:n \sdefinitionid
4357 \stex_smsmode_do:
4358 }{
4359   \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
4360   \stex_if_smsmode:F {
4361     \clist_set:No \l_tmpa_clist \sdefinitiontype
4362     \tl_clear:N \l_tmpa_tl
4363     \clist_map_inline:Nn \l_tmpa_clist {
4364       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
4365         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
4366       }
4367     }
4368     \tl_if_empty:NTF \l_tmpa_tl {
4369       \__stex_statements_sdefinition_end:
4370     }{
4371       \l_tmpa_tl
4372     }
4373   } \end{stex_annotate_env}
4374 }
4375 }

```

\stexpatchdefinition

```

4376 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
4377   \par\noindent\titleemph{Definition}\tl_if_empty:NF \sdefinitiontitle {
4378     ~(\sdefinitiontitle)
4379   }~}
4380 }

```

```

4381 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\par\medskip}
4382
4383 \newcommand\stexpatchdefinition[3] [] {
4384   \str_set:Nx \l_tmpa_str{ #1 }
4385   \str_if_empty:NTF \l_tmpa_str {
4386     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
4387     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
4388   }{
4389     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
4390     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
4391   }
4392 }

```

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

\inlinedef inline:

```

4393 \keys_define:nn {stex / inlinedef }{
4394   type      .str_set_x:N = \sdefinitiontype,
4395   id        .str_set_x:N = \sdefinitionid,
4396   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4397   name      .str_set_x:N = \sdefinitionname
4398 }
4399 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
4400   \str_clear:N \sdefinitiontype
4401   \str_clear:N \sdefinitionid
4402   \str_clear:N \sdefinitionname
4403   \clist_clear:N \l__stex_statements_sdefinition_for_clist
4404   \keys_set:nn { stex / inlinedef }{ #1 }
4405 }
4406 \NewDocumentCommand \inlinedef { 0{} m } {
4407   \beginngroup
4408   \__stex_statements_inlinedef_args:n{ #1 }
4409   \stex_reactivate_macro:N \definiendum
4410   \stex_reactivate_macro:N \definame
4411   \stex_reactivate_macro:N \Definame
4412   \stex_reactivate_macro:N \premise
4413   \stex_reactivate_macro:N \definiens
4414   \stex_ref_new_doc_target:n \sdefinitionid
4415   \stex_if_smsmode:TF{
4416     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
4417   }{
4418     \seq_clear:N \l_tmpa_seq
4419     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4420       \tl_if_empty:nF{ ##1 }{
4421         \stex_get_symbol:n { ##1 }
4422         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4423           \l_stex_get_symbol_uri_str
4424         }
4425       }
4426     }
4427     \exp_args:Nnx
4428     \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
4429       \str_if_empty:NF \sdefinitiontype {
4430         \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{

```



```

4431     }
4432     #2
4433     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{ }\sdefinitionname } }
4434   }
4435 }
4436 \endgroup
4437 \stex_smsmode_do:
4438 }

```

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

## 33.2 Assertions

**sassertion**

```

4439
4440 \keys_define:nn {stex / sassertion }{
4441   type      .str_set_x:N = \sassertiontype,
4442   id        .str_set_x:N = \sassertionid,
4443   title     .tl_set:N    = \sassertiontitle ,
4444   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4445   name      .str_set_x:N = \sassertionname
4446 }
4447 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
4448   \str_clear:N \sassertiontype
4449   \str_clear:N \sassertionid
4450   \str_clear:N \sassertionname
4451   \clist_clear:N \l__stex_statements_sassertion_for_clist
4452   \tl_clear:N \sassertiontitle
4453   \keys_set:nn { stex / sassertion }{ #1 }
4454 }
4455
4456 %\tl_new:N \g__stex_statements_aftergroup_tl
4457
4458 \NewDocumentEnvironment{sassertion}{0{}}{
4459   \__stex_statements_sassertion_args:n{ #1 }
4460   \stex_reactivate_macro:N \premise
4461   \stex_reactivate_macro:N \conclusion
4462   \stex_if_smsmode:F {
4463     \seq_clear:N \l_tmpa_seq
4464     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4465       \tl_if_empty:nF{ ##1 }{
4466         \stex_get_symbol:n { ##1 }
4467         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4468           \l_stex_get_symbol_uri_str
4469         }
4470       }
4471     }
4472     \exp_args:Nnnx
4473     \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}
4474     \str_if_empty:NF \sassertiontype {
4475       \stex_annotate_invisible:nnn{type}{\sassertiontype}{ }
4476     }
4477     \clist_set:Nn \l_tmpa_clist \sassertiontype

```

```

4478 \tl_clear:N \l_tmpa_tl
4479 \clist_map_inline:Nn \l_tmpa_clist {
4480   \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
4481     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
4482   }
4483 }
4484 \tl_if_empty:NTF \l_tmpa_tl {
4485   \__stex_statements_sassertion_start:
4486 }{
4487   \l_tmpa_tl
4488 }
4489 }
4490 \str_if_empty:NTF \sassertionid {
4491   \str_if_empty:NF \sassertionname {
4492     \stex_ref_new_doc_target:n {}
4493   }
4494 } {
4495   \stex_ref_new_doc_target:n \sassertionid
4496 }
4497 \stex_smsmode_do:
4498 ){
4499   \str_if_empty:NF \sassertionname {
4500     \stex_symdecl_do:nn{ }\sassertionname}
4501   \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4502 }
4503 \stex_if_smsmode:F {
4504   \clist_set:Nn \l_tmpa_clist \sassertiontype
4505   \tl_clear:N \l_tmpa_tl
4506   \clist_map_inline:Nn \l_tmpa_clist {
4507     \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
4508       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
4509     }
4510   }
4511   \tl_if_empty:NTF \l_tmpa_tl {
4512     \__stex_statements_sassertion_end:
4513   }{
4514     \l_tmpa_tl
4515   }
4516   \end{stex_annotate_env}
4517 }
4518 }

```

\stexpatchassertion

```

4519
4520 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
4521   \par\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
4522     (\sassertiontitle)
4523   }~}
4524 }
4525 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
4526
4527 \newcommand\stexpatchassertion[3] [] {
4528   \str_set:Nx \l_tmpa_str{ #1 }
4529   \str_if_empty:NTF \l_tmpa_str {

```

```

4530     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
4531     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
4532   }{
4533     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
4534     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
4535   }
4536 }

```

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

`\inlineass` inline:

```

4537 \keys_define:nn {stex / inlineass }{
4538   type      .str_set_x:N = \sassertiontype,
4539   id        .str_set_x:N = \sassertionid,
4540   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4541   name      .str_set_x:N = \sassertionname
4542 }
4543 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
4544   \str_clear:N \sassertiontype
4545   \str_clear:N \sassertionid
4546   \str_clear:N \sassertionname
4547   \clist_clear:N \l__stex_statements_sassertion_for_clist
4548   \keys_set:nn { stex / inlineass }{ #1 }
4549 }
4550 \NewDocumentCommand \inlineass { 0{} m } {
4551   \begin_group
4552   \stex_reactivate_macro:N \premise
4553   \stex_reactivate_macro:N \conclusion
4554   \__stex_statements_inlineass_args:n{ #1 }
4555   \str_if_empty:NTF \sassertionid {
4556     \str_if_empty:NF \sassertionname {
4557       \stex_ref_new_doc_target:n { }
4558     }
4559   } {
4560     \stex_ref_new_doc_target:n \sassertionid
4561   }
4562
4563   \stex_if_smsmode:TF{
4564     \str_if_empty:NF \sassertionname {
4565       \stex_symdecl_do:nn{}{\sassertionname}
4566       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4567     }
4568   }{
4569     \seq_clear:N \l_tmpa_seq
4570     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4571       \tl_if_empty:nF{ ##1 }{
4572         \stex_get_symbol:n { ##1 }
4573         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4574           \l_stex_get_symbol_uri_str
4575         }
4576       }
4577     }
4578     \exp_args:Nnx
4579     \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{

```

```

4580     \str_if_empty:NF \sassertiontype {
4581       \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
4582     }
4583     #2
4584     \str_if_empty:NF \sassertionname {
4585       \stex_symdecl_do:nn{}{\sassertionname}
4586       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4587     }
4588   }
4589 }
4590 \endgroup
4591 \stex_smsmode_do:
4592 }

```

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

### 33.3 Examples

`sexample`

```

4593
4594 \keys_define:nn {stex / sexample }{
4595   type      .str_set_x:N = \exampletype,
4596   id        .str_set_x:N = \sexampleid,
4597   title     .tl_set:N    = \sexampletitle,
4598   for       .clist_set:N = \l__stex_statements_sexample_for_clist,
4599 }
4600 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
4601   \str_clear:N \sexampletype
4602   \str_clear:N \sexampleid
4603   \tl_clear:N \sexampletitle
4604   \clist_clear:N \l__stex_statements_sexample_for_clist
4605   \keys_set:nn { stex / sexample }{ #1 }
4606 }
4607
4608 \NewDocumentEnvironment{sexample}{0{}}{
4609   \__stex_statements_sexample_args:n{ #1 }
4610   \stex_reactivate_macro:N \premise
4611   \stex_reactivate_macro:N \conclusion
4612   \stex_if_smsmode:F {
4613     \seq_clear:N \l_tmpa_seq
4614     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4615       \tl_if_empty:nF{ ##1 }{
4616         \stex_get_symbol:n { ##1 }
4617         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4618           \l_stex_get_symbol_uri_str
4619         }
4620       }
4621     }
4622     \exp_args:Nnnx
4623     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
4624     \str_if_empty:NF \sexampletype {
4625       \stex_annotate_invisible:nnn{type}{\sexampletype}{}
4626     }

```

```

4627 \clist_set:No \l_tmpa_clist \sexamplotype
4628 \tl_clear:N \l_tmpa_tl
4629 \clist_map_inline:Nn \l_tmpa_clist {
4630   \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
4631     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
4632   }
4633 }
4634 \tl_if_empty:NTF \l_tmpa_tl {
4635   \__stex_statements_sexample_start:
4636 }{
4637   \l_tmpa_tl
4638 }
4639 }
4640 \str_if_empty:NF \sexampleid {
4641   \stex_ref_new_doc_target:n \sexampleid
4642 }
4643 \stex_smsmode_do:
4644 }{
4645   \str_if_empty:NF \sexamplename { \stex_symdecl_do:nn{}{\sexamplename} }
4646   \stex_if_smsmode:F {
4647     \clist_set:No \l_tmpa_clist \sexamplotype
4648     \tl_clear:N \l_tmpa_tl
4649     \clist_map_inline:Nn \l_tmpa_clist {
4650       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
4651         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
4652       }
4653     }
4654     \tl_if_empty:NTF \l_tmpa_tl {
4655       \__stex_statements_sexample_end:
4656     }{
4657       \l_tmpa_tl
4658     }
4659     \end{stex_annotate_env}
4660   }
4661 }

```

\stexpatchexample

```

4662 \cs_new_protected:Nn \__stex_statements_sexample_start: {
4663   \par\noindent\titllemph{Example~\tl_if_empty:NF \sexamplitle {
4664     (\sexamplitle)
4665   }~}
4666 }
4667 }
4668 \cs_new_protected:Nn \__stex_statements_sexample_end: { \par\medskip}
4669
4670 \newcommand\stexpatchexample[3]{} {
4671   \str_set:Nx \l_tmpa_str{ #1 }
4672   \str_if_empty:NTF \l_tmpa_str {
4673     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
4674     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
4675   }{
4676     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
4677     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
4678   }

```

4679 }

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

`\inlineex` inline:

```

4680 \keys_define:nn {stex / inlineex }{
4681   type      .str_set_x:N = \sexamplotype,
4682   id        .str_set_x:N = \sexampleid,
4683   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
4684   name      .str_set_x:N = \sexamplename
4685 }
4686 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
4687   \str_clear:N \sexamplotype
4688   \str_clear:N \sexampleid
4689   \str_clear:N \sexamplename
4690   \clist_clear:N \l__stex_statements_sexample_for_clist
4691   \keys_set:nn { stex / inlineex }{ #1 }
4692 }
4693 \NewDocumentCommand \inlineex { 0{} m } {
4694   \begingroup
4695   \stex_reactivate_macro:N \premise
4696   \stex_reactivate_macro:N \conclusion
4697   \__stex_statements_inlineex_args:n{ #1 }
4698   \str_if_empty:NF \sexampleid {
4699     \stex_ref_new_doc_target:n \sexampleid
4700   }
4701   \stex_if_smsmode:TF{
4702     \str_if_empty:NF \sexamplename { \stex_symdecl_do:nn{ }\sexamplename } }
4703   }{
4704     \seq_clear:N \l_tmpa_seq
4705     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4706       \tl_if_empty:nF{ ##1 }{
4707         \stex_get_symbol:n { ##1 }
4708         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4709           \l_stex_get_symbol_uri_str
4710         }
4711       }
4712     }
4713     \exp_args:Nnx
4714     \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpa_seq {,}}{
4715       \str_if_empty:NF \sexamplotype {
4716         \stex_annotate_invisible:nnn{type}{\sexamplotype}{ }
4717       }
4718       #2
4719       \str_if_empty:NF \sexamplename { \stex_symdecl_do:nn{ }\sexamplename } }
4720     }
4721   }
4722   \endgroup
4723   \stex_smsmode_do:
4724 }
```

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

## 33.4 Logical Paragraphs

sparagraph

```

4725 \keys_define:nn { stex / sparagraph } {
4726   id       .str_set_x:N = \sparagraphid ,
4727   title    .tl_set:N    = \l_stex_sparagraph_title_tl ,
4728   type     .str_set_x:N = \sparagraphtype ,
4729   for      .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
4730   from     .tl_set:N    = \sparagraphfrom ,
4731   to       .tl_set:N    = \sparagraphto ,
4732   start    .tl_set:N    = \l_stex_sparagraph_start_tl ,
4733   name     .str_set:N   = \sparagraphname
4734 }
4735
4736 \cs_new_protected:Nn \stex_sparagraph_args:n {
4737   \tl_clear:N \l_stex_sparagraph_title_tl
4738   \tl_clear:N \sparagraphfrom
4739   \tl_clear:N \sparagraphto
4740   \tl_clear:N \l_stex_sparagraph_start_tl
4741   \str_clear:N \sparagraphid
4742   \str_clear:N \sparagraphtype
4743   \clist_clear:N \l__stex_statements_sparagraph_for_clist
4744   \str_clear:N \sparagraphname
4745   \keys_set:nn { stex / sparagraph } { #1 }
4746 }
4747 \newif\if@in@omtext\@in@omtextfalse
4748
4749 \NewDocumentEnvironment {sparagraph} { 0{} } {
4750   \stex_sparagraph_args:n { #1 }
4751   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4752     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
4753   }{
4754     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
4755   }
4756   \@in@omtexttrue
4757   \stex_if_smsmode:F {
4758     \seq_clear:N \l_tmpa_seq
4759     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
4760       \tl_if_empty:NF{ ##1 }{
4761         \stex_get_symbol:n { ##1 }
4762         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4763           \l_stex_get_symbol_uri_str
4764         }
4765       }
4766     }
4767     \exp_args:Nnnx
4768     \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
4769     \str_if_empty:NF \sparagraphtype {
4770       \stex_annotate_invisible:nnn{type}{\sparagraphtype}{ }
4771     }
4772     \str_if_empty:NF \sparagraphfrom {
4773       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{ }
4774     }
4775     \str_if_empty:NF \sparagraphto {

```

```

4776     \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
4777 }
4778 \clist_set:No \l_tmpa_clist \sparagraphtype
4779 \tl_clear:N \l_tmpa_tl
4780 \clist_map_inline:Nn \sparagraphtype {
4781     \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
4782         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
4783     }
4784 }
4785 \tl_if_empty:NTF \l_tmpa_tl {
4786     \__stex_statements_sparagraph_start:
4787 }{
4788     \l_tmpa_tl
4789 }
4790 }
4791 \clist_set:No \l_tmpa_clist \sparagraphtype
4792 \str_if_empty:NTF \sparagraphid {
4793     \str_if_empty:NTF \sparagraphname {
4794         \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
4795             \stex_ref_new_doc_target:n {}
4796         }
4797     } {
4798         \stex_ref_new_doc_target:n {}
4799     }
4800 } {
4801     \stex_ref_new_doc_target:n \sparagraphid
4802 }
4803 \exp_args:NNx
4804 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
4805     \clist_map_inline:Nn \__stex_statements_sparagraph_for_clist {
4806         \tl_if_empty:nF{ ##1 }{
4807             \stex_get_symbol:n { ##1 }
4808             \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4809         }
4810     }
4811 }
4812 \stex_smsmode_do:
4813 \ignorespacesandpars
4814 }{
4815     \str_if_empty:NF \sparagraphname {
4816         \stex_symdecl_do:nn{}{\sparagraphname}
4817         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
4818     }
4819     \stex_if_smsmode:F {
4820         \clist_set:No \l_tmpa_clist \sparagraphtype
4821         \tl_clear:N \l_tmpa_tl
4822         \clist_map_inline:Nn \l_tmpa_clist {
4823             \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
4824                 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
4825             }
4826         }
4827         \tl_if_empty:NTF \l_tmpa_tl {
4828             \__stex_statements_sparagraph_end:
4829         }{

```



```

4830     \l_tmpa_tl
4831   }
4832   \end{stex_annotate_env}
4833 }
4834 }

```

# \stexpatchparagraph

```

4835
4836 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
4837   \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4838     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
4839       \titleemph{\l_stex_sparagraph_title_tl}:~
4840     }
4841   }{
4842     \titleemph{\l_stex_sparagraph_start_tl}~
4843   }
4844 }
4845 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
4846
4847 \newcommand\stexpatchparagraph[3] [] {
4848   \str_set:Nx \l_tmpa_str{ #1 }
4849   \str_if_empty:NTF \l_tmpa_str {
4850     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
4851     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
4852   }{
4853     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
4854     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
4855   }
4856 }
4857
4858 \keys_define:nn { stex / inlinepara} {
4859   id      .str_set_x:N = \sparagraphid ,
4860   type    .str_set_x:N = \sparagraphtype ,
4861   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
4862   from    .tl_set:N    = \sparagraphfrom ,
4863   to      .tl_set:N    = \sparagraphto ,
4864   name    .str_set:N   = \sparagraphname
4865 }
4866 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
4867   \tl_clear:N \sparagraphfrom
4868   \tl_clear:N \sparagraphto
4869   \str_clear:N \sparagraphid
4870   \str_clear:N \sparagraphtype
4871   \clist_clear:N \l__stex_statements_sparagraph_for_clist
4872   \str_clear:N \sparagraphname
4873   \keys_set:nn { stex / inlinepara }{ #1 }
4874 }
4875 \NewDocumentCommand \inlinepara { 0{} m } {
4876   \begingroup
4877   \__stex_statements_inlinepara_args:n{ #1 }
4878   \clist_set:No \l_tmpa_clist \sparagraphtype
4879   \str_if_empty:NTF \sparagraphid {
4880     \str_if_empty:NTF \sparagraphname {
4881       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{

```

```

4882     \stex_ref_new_doc_target:n {}
4883   }
4884 } {
4885   \stex_ref_new_doc_target:n {}
4886 }
4887 } {
4888   \stex_ref_new_doc_target:n \sparagraphid
4889 }
4890 \stex_if_smsmode:TF{
4891   \str_if_empty:NF \sparagraphname {
4892     \stex_symdecl_do:nn{ }\sparagraphname}
4893   \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
4894 }
4895 }{
4896   \seq_clear:N \l_tmpa_seq
4897   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
4898     \tl_if_empty:nF{ ##1 }{
4899       \stex_get_symbol:n { ##1 }
4900       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4901         \l_stex_get_symbol_uri_str
4902       }
4903     }
4904   }
4905   \exp_args:Nnx
4906   \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
4907     \str_if_empty:NF \sparagraphtype {
4908       \stex_annotate_invisible:nnn{type}{\sparagraphtype}{ }
4909     }
4910     \str_if_empty:NF \sparagraphfrom {
4911       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{ }
4912     }
4913     \str_if_empty:NF \sparagraphto {
4914       \stex_annotate_invisible:nnn{to}{\sparagraphto}{ }
4915     }
4916     \str_if_empty:NF \sparagraphname {
4917       \stex_symdecl_do:nn{ }\sparagraphname}
4918     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
4919   }
4920   \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
4921     \clist_map_inline:Nn \l_tmpa_seq {
4922       \stex_ref_new_sym_target:n {##1}
4923     }
4924   }
4925   #2
4926 }
4927 }
4928 \endgroup
4929 \stex_smsmode_do:
4930 }
4931
4932 (End definition for \stexpatchparagraph. This function is documented on page ??.)
4932 \</package>

```

# Chapter 34

## The Implementation

### 34.1 Package Options

We declare some switches which will modify the behavior according to the package options. Generally, an option `xxx` will just set the appropriate switches to true (otherwise they stay false).<sup>13</sup>

```
4933 <*package>
4934 <@@=stex_sproof>
4935
4936 %%%%%%%%%%% sproof.dtx %%%%%%%%%%%
4937
```

### 34.2 Proofs

We first define some keys for the proof environment.

```
4938 \keys_define:nn { stex / spf } {
4939   id          .str_set_x:N = \spfid,
4940   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,
4941   from        .tl_set:N    = \l__stex_sproof_spf_from_tl ,
4942   proofend    .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
4943   type        .str_set_x:N = \spftype,
4944   title       .tl_set:N    = \spftitle,
4945   continues   .tl_set:N    = \l__stex_sproof_spf_continues_tl,
4946   functions   .tl_set:N    = \l__stex_sproof_spf_functions_tl,
4947   method      .tl_set:N    = \l__stex_sproof_spf_method_tl
4948 }
4949 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
4950   \str_clear:N \spfid
4951   \tl_clear:N \l__stex_sproof_spf_for_tl
4952   \tl_clear:N \l__stex_sproof_spf_from_tl
4953   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
4954   \str_clear:N \spftype
4955   \tl_clear:N \spftitle
4956   \tl_clear:N \l__stex_sproof_spf_continues_tl
4957   \tl_clear:N \l__stex_sproof_spf_functions_tl
```

---

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

```

4958 \tl_clear:N \l__stex_sproof_spf_method_tl
4959 \keys_set:nn { stex / spf }{ #1 }
4960 }

```

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

```

4961 \str_set:Nn\c__stex_sproof_flow_str{inline}

```

(End definition for `\c__stex_sproof_flow_str`.)

For proofs, we will have to have deeply nested structures of enumerated list-like environments. However, L<sup>A</sup>T<sub>E</sub>X only allows `enumerate` environments up to nesting depth 4 and general list environments up to listing depth 6. This is not enough for us. Therefore we have decided to go along the route proposed by Leslie Lamport to use a single top-level list with dotted sequences of numbers to identify the position in the proof tree. Unfortunately, we could not use his `pf.sty` package directly, since it does not do automatic numbering, and we have to add keyword arguments all over the place, to accomodate semantic information.

`pst@with@label` This environment manages<sup>6</sup> the path labeling of the proof steps in the description environment of the outermost `proof` environment. The argument is the label prefix up to now; which we cache in `\pst@label` (we need evaluate it first, since are in the right place now!). Then we increment the proof depth which is stored in `\count10` (lower counters are used by T<sub>E</sub>X for page numbering) and initialize the next level counter `\count\count10` with 1. In the end call for this environment, we just decrease the proof depth counter by 1 again.

```

4962 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
4963 \cs_new_protected:Npn \sproofnumber {
4964   \int_set:Nn \l_tmpa_int {1}
4965   \bool_while_do:nn {
4966     \int_compare_p:nNn {
4967       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
4968     } > 0
4969   }{
4970     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
4971     \int_incr:N \l_tmpa_int
4972   }
4973 }
4974 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
4975   \int_set:Nn \l_tmpa_int {1}
4976   \bool_while_do:nn {
4977     \int_compare_p:nNn {
4978       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
4979     } > 0
4980   }{
4981     \int_incr:N \l_tmpa_int
4982   }
4983   \int_compare:nNnF \l_tmpa_int = 1 {
4984     \int_decr:N \l_tmpa_int
4985   }
4986   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
4987     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1
4988   }

```

---

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



```

5033 \clist_if_in:NnT \l_tmpa_clist {finnish}{
5034 \input{sproof-finnish.ldf}
5035 }
5036 \clist_if_in:NnT \l_tmpa_clist {french}{
5037 \input{sproof-french.ldf}
5038 }
5039 \clist_if_in:NnT \l_tmpa_clist {russian}{
5040 \input{sproof-russian.ldf}
5041 }
5042 \makeatother
5043 }{}
5044 }

```

spfsketch

```

5045 \newcommand\spfsketch[2] [] {
5046 \beginingroup
5047 \let \premise \stex_proof_premise:
5048 \_stex_sproof_spf_args:n{#1}
5049 \stex_if_smsmode:TF {
5050 \str_if_empty:NF \spfid {
5051 \stex_ref_new_doc_target:n \spfid
5052 }
5053 }{
5054 \seq_clear:N \l_tmpa_seq
5055 \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5056 \tl_if_empty:NF{ ##1 }{
5057 \stex_get_symbol:n { ##1 }
5058 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5059 \l_stex_get_symbol_uri_str
5060 }
5061 }
5062 }
5063 \exp_args:Nnx
5064 \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
5065 \str_if_empty:NF \spftype {
5066 \stex_annotate_invisible:nnn{type}{\spftype}{}}
5067 }
5068 \clist_set:No \l_tmpa_clist \spftype
5069 \tl_set:Nn \l_tmpa_tl {
5070 \titleemph{
5071 \tl_if_empty:NTF \spftitle {
5072 \spf@proofsketch@kw
5073 }{
5074 \spftitle
5075 }
5076 }::~
5077 }
5078 \clist_map_inline:Nn \l_tmpa_clist {
5079 \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5080 \tl_clear:N \l_tmpa_tl
5081 }
5082 }
5083 \str_if_empty:NF \spfid {
5084 \stex_ref_new_doc_target:n \spfid

```

```

5085     }
5086     \l_tmpa_tl #2 \sproofend
5087   }
5088 }
5089 \endgroup
5090 \stex_smsmode_do:
5091 }
5092

```

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

**spfeq** This is very similar to `\spfsketch`, but uses a computation array<sup>1415</sup>

```

5093 \newenvironment{spfeq}[2][]{
5094   \__stex_sproof_spf_args:n{#1}
5095   \let \premise \stex_proof_premise:
5096   \stex_if_smsmode:TF {
5097     \str_if_empty:NF \spfid {
5098       \stex_ref_new_doc_target:n \spfid
5099     }
5100   }{
5101     \seq_clear:N \l_tmpa_seq
5102     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5103       \tl_if_empty:nF{ ##1 }{
5104         \stex_get_symbol:n { ##1 }
5105         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5106           \l_stex_get_symbol_uri_str
5107         }
5108       }
5109     }
5110     \exp_args:Nnnx
5111     \begin{stex_annotate_env}{spfeq}{\seq_use:Nn \l_tmpa_seq {,}}
5112     \str_if_empty:NF \spftype {
5113       \stex_annotate_invisible:nnn{type}{\spftype}{ }
5114     }
5115
5116     \clist_set:No \l_tmpa_clist \spftype
5117     \tl_clear:N \l_tmpa_tl
5118     \clist_map_inline:Nn \l_tmpa_clist {
5119       \tl_if_exist:cT {__stex_sproof_spfeq_##1_start:}{
5120         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_start:}}
5121       }
5122       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5123         \tl_set:Nn \l_tmpa_tl {\use:n{ }}
5124       }
5125     }
5126     \tl_if_empty:NTF \l_tmpa_tl {
5127       \__stex_sproof_spfeq_start:
5128     }{
5129       \l_tmpa_tl
5130     }{-#2}
5131     \str_if_empty:NF \spfid {

```

<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

```

5132     \stex_ref_new_doc_target:n \spfid
5133   }
5134   \begin{displaymath}\begin{array}{rc1l}
5135   }
5136   \stex_smsmode_do:
5137 }{
5138   \stex_if_smsmode:F {
5139     \end{array}\end{displaymath}
5140     \clist_set:No \l_tmpa_clist \spftype
5141     \tl_clear:N \l_tmpa_tl
5142     \clist_map_inline:Nn \l_tmpa_clist {
5143       \tl_if_exist:cT {__stex_sproof_spfeq_##1_end:}{
5144         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_end:}}
5145       }
5146     }
5147     \tl_if_empty:NTF \l_tmpa_tl {
5148       \__stex_sproof_spfeq_end:
5149     }{
5150       \l_tmpa_tl
5151     }
5152     \end{stex_annotate_env}
5153   }
5154 }
5155
5156 \cs_new_protected:Nn \__stex_sproof_spfeq_start: {
5157   \titleemph{
5158     \tl_if_empty:NTF \spftitle {
5159       \spf@proof@kw
5160     }{
5161       \spftitle
5162     }
5163   }:
5164 }
5165 \cs_new_protected:Nn \__stex_sproof_spfeq_end: {\sproofend}
5166
5167 \newcommand\stexpatchspfeq[3] [] {
5168   \str_set:Nx \l_tmpa_str{ #1 }
5169   \str_if_empty:NTF \l_tmpa_str {
5170     \tl_set:Nn \__stex_sproof_spfeq_start: { #2 }
5171     \tl_set:Nn \__stex_sproof_spfeq_end: { #3 }
5172   }{
5173     \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_start:\endcsname{ #2 }
5174     \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_end:\endcsname{ #3 }
5175   }
5176 }
5177

```

(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.

```

5178 \newenvironment{sproof}[2] []{
5179   \let \premise \stex_proof_premise:

```



```

5180 \intarray_gzero:N \l__stex_sproof_counter_intarray
5181 \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
5182 \__stex_sproof_spf_args:n{#1}
5183 \stex_if_smsmode:TF {
5184   \str_if_empty:NF \spfid {
5185     \stex_ref_new_doc_target:n \spfid
5186   }
5187 }{
5188   \seq_clear:N \l_tmpa_seq
5189   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5190     \tl_if_empty:nF{ #1 }{
5191       \stex_get_symbol:n { #1 }
5192       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5193         \l_stex_get_symbol_uri_str
5194       }
5195     }
5196   }
5197   \exp_args:Nnnx
5198   \begin{stex_annotate_env}{sproof}{\seq_use:Nn \l_tmpa_seq {},}}
5199   \str_if_empty:NF \spftype {
5200     \stex_annotate_invisible:nnn{type}{\spftype}{}
5201   }
5202
5203   \clist_set:No \l_tmpa_clist \spftype
5204   \tl_clear:N \l_tmpa_tl
5205   \clist_map_inline:Nn \l_tmpa_clist {
5206     \tl_if_exist:cT {__stex_sproof_sproof_#1_start:}{
5207       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_#1_start:}}
5208     }
5209     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {#1} {
5210       \tl_set:Nn \l_tmpa_tl {\use:n{}}
5211     }
5212   }
5213   \tl_if_empty:NTF \l_tmpa_tl {
5214     \__stex_sproof_sproof_start:
5215   }{
5216     \l_tmpa_tl
5217   }{~#2}
5218   \str_if_empty:NF \spfid {
5219     \stex_ref_new_doc_target:n \spfid
5220   }
5221   \begin{description}
5222 }
5223 \stex_smsmode_do:
5224 }{
5225   \stex_if_smsmode:F{
5226     \end{description}
5227     \clist_set:No \l_tmpa_clist \spftype
5228     \tl_clear:N \l_tmpa_tl
5229     \clist_map_inline:Nn \l_tmpa_clist {
5230       \tl_if_exist:cT {__stex_sproof_sproof_#1_end:}{
5231         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_#1_end:}}
5232       }
5233     }

```

```

5234 \tl_if_empty:NTF \l_tmpa_tl {
5235   \__stex_sproof_sproof_end:
5236 }{
5237   \l_tmpa_tl
5238 }
5239 \end{stex_annotate_env}
5240 }
5241 }
5242
5243 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
5244   \par\noindent\titleemph{
5245     \tl_if_empty:NTF \spftype {
5246       \spf@proof@kw
5247     }{
5248       \spftype
5249     }
5250   }:
5251 }
5252 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
5253
5254 \newcommand\stexpatchsproof[3] [] {
5255   \str_set:Nx \l_tmpa_str{ #1 }
5256   \str_if_empty:NTF \l_tmpa_str {
5257     \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
5258     \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
5259   }{
5260     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
5261     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
5262   }
5263 }

```

**\spfidea**

```

5264 \newcommand\spfidea[2] []{
5265   \__stex_sproof_spf_args:n{#1}
5266   \titleemph{
5267     \tl_if_empty:NTF \spftype {Proof-Idea}{
5268       \spftype
5269     }:
5270   }~#2
5271   \sproofend
5272 }

```

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

The next two environments (proof steps) and comments, are mostly semantical, they take **KeyVal** arguments that specify their semantic role. In draft mode, they read these values and show them. If the surrounding proof had **display=flow**, then no new **\item** is generated, otherwise it is. In any case, the proof step number (at the current level) is incremented.

**spfstep**

```

5273 \newenvironment{spfstep}[1] []{
5274   \__stex_sproof_spf_args:n{#1}
5275   \stex_if_smsmode:TF {
5276     \str_if_empty:NF \spfid {

```

```

5277     \stex_ref_new_doc_target:n \spfid
5278   }
5279 }{
5280   \@in@omtexttrue
5281   \seq_clear:N \l_tmpa_seq
5282   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5283     \tl_if_empty:nF{ ##1 }{
5284       \stex_get_symbol:n { ##1 }
5285       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5286         \l_stex_get_symbol_uri_str
5287       }
5288     }
5289   }
5290   \exp_args:Nnnx
5291   \begin{stex_annotate_env}{spfstep}{\seq_use:Nn \l_tmpa_seq {,}}
5292   \str_if_empty:NF \spftype {
5293     \stex_annotate_invisible:nnn{type}{\spftype}{}
5294   }
5295   \clist_set:Nn \l_tmpa_clist \spftype
5296   \tl_set:Nn \l_tmpa_tl {
5297     \item[\sproofnumber]
5298   }
5299   \clist_map_inline:Nn \l_tmpa_clist {
5300     \exp_args:Nn \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5301       \tl_clear:N \l_tmpa_tl
5302     }
5303   }
5304   \l_tmpa_tl
5305   \tl_if_empty:NF \spftitle {
5306     {(\titleemph{\spftitle})\enspace}
5307   }
5308   \str_if_empty:NF \spfid {
5309     \stex_ref_new_doc_target:n \spfid
5310   }
5311 }
5312 \__stex_sproof_inc_counter:
5313 \stex_smsmode_do:
5314 \ignorespacesandpars
5315 }{
5316   \stex_if_smsmode:F {
5317     \end{stex_annotate_env}
5318   }
5319 }

```

#### sproofcomment

```

5320 \newenvironment{sproofcomment}[1][]{
5321   \__stex_sproof_spf_args:n{#1}
5322   \clist_set:Nn \l_tmpa_clist \spftype
5323   \tl_set:Nn \l_tmpa_tl {
5324     \item[\sproofnumber]
5325   }
5326   \clist_map_inline:Nn \l_tmpa_clist {
5327     \exp_args:Nn \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5328       \tl_clear:N \l_tmpa_tl

```

```

5329     }
5330   }
5331   \l_tmpa_tl
5332 }{
5333 }

```

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.

```

5334 \newenvironment{subproof}[2][]{
5335   \_stex_sproof_spf_args:n{#1}
5336   \stex_if_smsmode:TF{
5337     \str_if_empty:NF \spfid {
5338       \stex_ref_new_doc_target:n \spfid
5339     }
5340   }{
5341     \seq_clear:N \l_tmpa_seq
5342     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5343       \tl_if_empty:NF{ ##1 }{
5344         \stex_get_symbol:n { ##1 }
5345         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5346           \l_stex_get_symbol_uri_str
5347         }
5348       }
5349     }
5350     \exp_args:Nnnx
5351     \begin{stex_annotate_env}{subproof}{\seq_use:Nn \l_tmpa_seq {,}}
5352     \str_if_empty:NF \spftype {
5353       \stex_annotate_invisible:nnn{type}{\spftype}{ }
5354     }
5355
5356     \clist_set:No \l_tmpa_clist \spftype
5357     \tl_set:Nn \l_tmpa_tl {
5358       \item[\sproofnumber]
5359     }
5360     \clist_map_inline:Nn \l_tmpa_clist {
5361       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5362         \tl_clear:N \l_tmpa_tl
5363       }
5364     }
5365     \l_tmpa_tl
5366     \tl_if_empty:NF \spftitle {
5367       {(\titleemph{\spftitle})\enspace}
5368     }
5369     {~#2}
5370     \str_if_empty:NF \spfid {
5371       \stex_ref_new_doc_target:n \spfid
5372     }
5373   }
5374   \_stex_sproof_add_counter:
5375   \stex_smsmode_do:
5376 }{
5377   \_stex_sproof_remove_counter:

```

```

5378 \stex_sproof_inc_counter:
5379 \stex_if_smsmode:F{
5380   \end{stex_annotate_env}
5381 }
5382 }

```

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

```

5383 \newenvironment{spfcases}[2][]{
5384   \tl_if_empty:nTF{#1}{
5385     \begin{subproof}[method=by-cases]{#2}
5386   }{
5387     \begin{subproof}[#1,method=by-cases]{#2}
5388   }
5389 }{
5390   \end{subproof}
5391 }

```

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

```

5392 \newenvironment{spfcase}[2][]{
5393   \stex_sproof_spf_args:n{#1}
5394   \stex_if_smsmode:TF {
5395     \str_if_empty:NF \spfid {
5396       \stex_ref_new_doc_target:n \spfid
5397     }
5398   }{
5399     \seq_clear:N \l_tmpa_seq
5400     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5401       \tl_if_empty:nF{ ##1 }{
5402         \stex_get_symbol:n { ##1 }
5403         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5404           \l_stex_get_symbol_uri_str
5405         }
5406       }
5407     }
5408     \exp_args:Nnnx
5409     \begin{stex_annotate_env}{spfcase}{\seq_use:Nn \l_tmpa_seq {,}}
5410     \str_if_empty:NF \spftype {
5411       \stex_annotate_invisible:nnn{type}{\spftype}{ }
5412     }
5413     \clist_set:Nn \l_tmpa_clist \spftype
5414     \tl_set:Nn \l_tmpa_tl {
5415       \item[\sproofnumber]
5416     }
5417     \clist_map_inline:Nn \l_tmpa_clist {
5418       \exp_args:Nn \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5419         \tl_clear:N \l_tmpa_tl
5420       }
5421     }
5422     \l_tmpa_tl
5423     \tl_if_empty:nF{#2}{
5424       \titleemph{#2}:~
5425     }
5426   }

```

```

5427 \__stex_sproof_inc_counter:
5428 \stex_smsmode_do:
5429 }{
5430 \stex_if_smsmode:F{
5431 \clist_set:No \l_tmpa_clist \spftype
5432 \tl_set:Nn \l_tmpa_tl{\sproofend}
5433 \clist_map_inline:Nn \l_tmpa_clist {
5434 \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5435 \tl_clear:N \l_tmpa_tl
5436 }
5437 }
5438 \l_tmpa_tl
5439 \end{stex_annotate_env}
5440 }
5441 }

```

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

```

5442 \newcommand\spfcasesketch[3][]{
5443 \begin{spfcase}[#1]{#2}#3\end{spfcase}
5444 }

```

### 34.3 Justifications

We define the actions that are undertaken, when the keys for justifications are encountered. Here this is very simple, we just define an internal macro with the value, so that we can use it later.

```

5445 \keys_define:nn { stex / just }{
5446 id .str_set_x:N = \l__stex_sproof_just_id_str,
5447 method .tl_set:N = \l__stex_sproof_just_method_tl,
5448 premises .tl_set:N = \l__stex_sproof_just_premises_tl,
5449 args .tl_set:N = \l__stex_sproof_just_args_tl
5450 }

```

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>

**justification**

```

5451 \newenvironment{justification}[1]{}{}

```

**\premise**

```

5452 \newcommand\stex_proof_premise:[2]{}{#2}

```

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

**\justarg** the **\justarg** macro is purely semantic, so we ignore the keyval arguments for now and only display the content.

```

5453 \newcommand\justarg[2]{}{#2}
5454 \</package>

```

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

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

---

<sup>16</sup>EdNOTE: need to do something about the premise in draft mode.

## Chapter 35

# STEX -Others Implementation

```
5455 <*package>
5456
5457 %%%%%%%%%% others.dtx %%%%%%%%%%
5458
5459 <@@=stex_others>
    Warnings and error messages
5460 % None

\MSC Math subject classifier

5461 \NewDocumentCommand \MSC {m} {
5462 % TODO
5463 }

(End definition for \MSC. This function is documented on page ??.)
    Patching tikzinput, if loaded
5464 \@ifpackageloaded{tikzinput}{
5465 \RequirePackage{stex-tikzinput}
5466 }{}
5467 </package>
```

## Chapter 36

# STEX -Metatheory Implementation

```
5468 <*package>
5469 <@@=stex_modules>
5470
5471 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
5472
5473 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
5474 \begingroup
5475 \stex_module_setup:nn{
5476   ns=\c_stex_metatheory_ns_str,
5477   meta=NONE
5478 }{Metatheory}
5479 \stex_reactivate_macro:N \symdecl
5480 \stex_reactivate_macro:N \notation
5481 \stex_reactivate_macro:N \symdef
5482 \ExplSyntaxOff
5483 \csname stex_suppress_html:n\endcsname{
5484   % is-a (a:A, a \in A, a is an A, etc.)
5485   \symdecl{isa}[args=ai]
5486   \notation{isa}[typed]{#1 \comp{:} #2}{##1 \comp, ##2}
5487   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
5488   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
5489
5490   % bind (\forall, \Pi, \lambda etc.)
5491   \symdecl{bind}[args=Bi]
5492   \notation{bind}[forall]{\comp\forall #1.\;#2}{##1 \comp, ##2}
5493   \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
5494   \notation{bind}[depfun]{\comp( #1 \comp{}\;\to\;} #2}{##1 \comp, ##2}
5495
5496   % dummy variable
5497   \symdecl{dummyvar}
5498   \notation{dummyvar}[underscore]{\comp\_}
5499   \notation{dummyvar}[dot]{\comp\cdot}
5500   \notation{dummyvar}[dash]{\comp{\rm --}}
5501
5502   %fromto (function space, Hom-set, implication etc.)
```



```

5503 \symdecl{fromto}[args=ai]
5504 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
5505 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
5506
5507 % mapto (lambda etc.)
5508 \symdecl{mapto}[args=Bi]
5509 %\notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
5510 %\notation{mapto}[lambda]{\comp\lambda #1 \comp.\; #2}{#1 \comp, #2}
5511 %\notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.\; #2}{#1 \comp, #2}
5512
5513 % function/operator application
5514 \symdecl{apply}[args=ia]
5515 \notation{apply}[prec=0;0x\infpref,parens]{#1 \comp( #2 \comp)}{##1 \comp, ##2}
5516 \notation{apply}[prec=0;0x\infpref,lambda]{#1 \; #2 }{##1 \; ; ##2}
5517
5518 % ‘type’ of all collections (sets, classes, types, kinds)
5519 \symdecl{collection}
5520 \notation{collection}[U]{\comp{\mathcal{U}}}
5521 \notation{collection}[set]{\comp{\textsf{Set}}}
5522
5523 % collection of propositions/booleans/truth values
5524 \symdecl{prop}[name=proposition]
5525 \notation{prop}[prop]{\comp{\rm prop}}
5526 \notation{prop}[BOOL]{\comp{\rm BOOL}}
5527
5528 % sequences
5529 \symdecl{seqtype}[args=1]
5530 \notation{seqtype}[kleene]{#1^{\comp\ast}}
5531
5532 \symdef{sequence-index}[args=2,li,prec=nobrackets]{#{#1}_{#2}}
5533 \notation{sequence-index}[ui,prec=nobrackets]{#{#1}^{#2}}
5534
5535 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{\,ellipses}}{##1\comp,##2}
5536 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{\,ellipses,}#2}{##1\comp,##2}
5537 \symdef{aseqfromtovia}[args=aai,prec=nobrackets]{#1\comp{\,ellipses,}#2\comp{\,ellipses,}#3}
5538
5539 % letin (‘let’, local definitions, variable substitution)
5540 \symdecl{letin}[args=bii]
5541 \notation{letin}[let]{\comp{\rm let}}\;#1\comp{=}\;#2\; \comp{\rm in}}\;#3}
5542 \notation{letin}[subst]{#3 \comp[ #1 \comp/ #2 \comp]}
5543 \notation{letin}[frac]{#3 \comp[ \frac{#2}{#1} \comp]}
5544
5545 % structures
5546 \symdecl*{module-type}[args=1]
5547 \notation{module-type}{\mathtt{MOD} #1}
5548 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
5549 \notation{mathstruct}[angle,prec=nobrackets]{\comp\angle #1 \comp\rangle}{##1 \comp, ##2}
5550
5551 }
5552 \ExplSyntaxOn
5553 \stex_add_to_current_module:n{
5554   \let\appa\apply
5555   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
5556   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}

```

```

5557 \def\livar{\csname sequence-index\endcsname[li]}
5558 \def\uivar{\csname sequence-index\endcsname[ui]}
5559 \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
5560 \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
5561 \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
5562 }
5563 \__stex_modules_end_module:
5564 \endgroup
5565 \</package>

```

## Chapter 37

# Tikzinput Implementation

```
5566 <*package>
5567
5568 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
5569
5570 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
5571 \RequirePackage{l3keys2e}
5572
5573 \keys_define:nn { tikzinput } {
5574   image .bool_set:N = \c_tikzinput_image_bool,
5575   image .default:n = false ,
5576   unknown .code:n = {}
5577 }
5578
5579 \ProcessKeysOptions { tikzinput }
5580
5581 \bool_if:NTF \c_tikzinput_image_bool {
5582   \RequirePackage{graphicx}
5583
5584   \providecommand\usetikzlibrary[]{}
5585   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
5586 }{
5587   \RequirePackage{tikz}
5588   \RequirePackage{standalone}
5589
5590   \newcommand \tikzinput [2] [] {
5591     \setkeys{Gin}{#1}
5592     \ifx \Gin@ewidth \Gin@exclamation
5593       \ifx \Gin@eheight \Gin@exclamation
5594         \input { #2 }
5595       \else
5596         \resizebox{!}{ \Gin@eheight }{
5597           \input { #2 }
5598         }
5599       \fi
5600     \else
5601       \ifx \Gin@eheight \Gin@exclamation
5602         \resizebox{ \Gin@ewidth }{!}{
5603           \input { #2 }
```

```

5604     }
5605     \else
5606         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
5607             \input { #2 }
5608         }
5609     \fi
5610 \fi
5611 }
5612 }
5613
5614 \newcommand \ctikzinput [2] [] {
5615     \begin{center}
5616         \tikzinput [1] {#2}
5617     \end{center}
5618 }
5619
5620 \@ifpackageloaded{stex}{
5621     \RequirePackage{stex-tikzinput}
5622 }{}
5623
5624 </package>
5625 <*stex>
5626 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
5627 \RequirePackage{stex}
5628 \RequirePackage{tikzinput}
5629
5630 \newcommand\mhtikzinput[2] [] {%
5631     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
5632     \stex_in_repository:nn\Gin@mhrepos{
5633         \tikzinput[#1]{\mhpath{##1}{#2}}
5634     }
5635 }
5636 \newcommand\cmhtikzinput[2] [] {\begin{center}\mhtikzinput[#1]{#2}\end{center}}
5637 </stex>

```

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

## Chapter 38

# document-structure.sty Implementation

### 38.1 The document-structure Class

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

```
5638 \*cls)
5639 \@@=document_structure)
5640 \ProvidesExplClass{document-structure}{2022/02/10}{3.0}{Modular Document Structure Class}
5641 \RequirePackage{13keys2e}
```

### 38.2 Class Options

To initialize the `document-structure` class, we declare and process the necessary options using the `kvoptions` package for key/value options handling. For `omdoc.cls` this is quite simple. We have options `report` and `book`, which set the `\omdoc@cls@class` macro and pass on the macro to `omdoc.sty` for further processing.

`\omdoc@cls@class`

```
5642 \keys_define:nn{ document-structure / pkg }{
5643   class      .str_set_x:N = \c_document_structure_class_str,
5644   minimal    .bool_set:N = \c_document_structure_minimal_bool,
5645   report     .code:n      = {
5646     \ClassWarning{document-structure}{the option 'report' is deprecated, use 'class=report',
5647     \str_set:Nn \c_document_structure_class_str {report}
5648   },
5649   book       .code:n      = {
5650     \ClassWarning{document-structure}{the option 'book' is deprecated, use 'class=book', ins
5651     \str_set:Nn \c_document_structure_class_str {book}
5652   },
5653   bookpart   .code:n      = {
5654     \ClassWarning{document-structure}{the option 'bookpart' is deprecated, use 'class=book,t
5655     \str_set:Nn \c_document_structure_class_str {book}
5656     \str_set:Nn \c_document_structure_topsect_str {chapter}
5657   },
```

```

5658 docopt      .str_set_x:N = \c_document_structure_docopt_str,
5659 unknown     .code:n      = {
5660   \PassOptionsToPackage{ \CurrentOption }{ document-structure }
5661 }
5662 }
5663 \ProcessKeysOptions{ document-structure / pkg }
5664 \str_if_empty:NT \c_document_structure_class_str {
5665   \str_set:Nn \c_document_structure_class_str {article}
5666 }
5667 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
5668   {\c_document_structure_class_str}
5669

```

### 38.3 Beefing up the document environment

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

```

5670 \RequirePackage{document-structure}
5671 \bool_if:NF \c_document_structure_minimal_bool {

```

And define the environments we need. The top-level one is the `document` environment, which we redefined so that we can provide keyval arguments.

**document** For the moment we do not use them on the L<sup>A</sup>T<sub>E</sub>X level, but the document identifier is picked up by L<sup>A</sup>T<sub>E</sub>XML.<sup>17</sup>

```

5672 \keys_define:nn { document-structure / document }{
5673   id .str_set_x:N = \c_document_structure_document_id_str
5674 }
5675 \let\__document_structure_orig_document=\document
5676 \renewcommand{\document}[1][]{
5677   \keys_set:nn{ document-structure / document }{ #1 }
5678   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
5679   \__document_structure_orig_document
5680 }

```

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

```

5681 }
5682 \</cls>

```

### 38.4 Implementation: document-structure Package

```

5683 \<package>
5684 \ProvidesExplPackage{document-structure}{2022/02/10}{3.0}{Modular Document Structure}
5685 \RequirePackage{l3keys2e}

```

### 38.5 Package Options

We declare some switches which will modify the behavior according to the package options. Generally, an option `xxx` will just set the appropriate switches to true (otherwise they stay false).

---

<sup>17</sup>EDNOTE: faking documentkeys for now. @HANG, please implement

```

5686
5687 \keys_define:nn{ document-structure / pkg }{
5688   class      .str_set_x:N = \c_document_structure_class_str,
5689   topsect    .str_set_x:N = \c_document_structure_topsect_str,
5690   % showignores .bool_set:N = \c_document_structure_showignores_bool,
5691 }
5692 \ProcessKeysOptions{ document-structure / pkg }
5693 \str_if_empty:NT \c_document_structure_class_str {
5694   \str_set:Nn \c_document_structure_class_str {article}
5695 }
5696 \str_if_empty:NT \c_document_structure_topsect_str {
5697   \str_set:Nn \c_document_structure_topsect_str {section}
5698 }

```

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

```

5699 \RequirePackage{xspace}
5700 \RequirePackage{comment}
5701 \AddToHook{begindocument}{
5702   \ltx@ifpackageloaded{babel}{
5703     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5704     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5705       \makeatletter\input{omdoc-ngerman.ldf}\makeatother
5706     }
5707   }{}
5708 }

```

`\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.

```

5709 \int_new:N \l_document_structure_section_level_int
5710 \str_case:VnF \c_document_structure_topsect_str {
5711   {part}}{
5712     \int_set:Nn \l_document_structure_section_level_int {0}
5713   }
5714   {chapter}{
5715     \int_set:Nn \l_document_structure_section_level_int {1}
5716   }
5717 }{
5718   \str_case:VnF \c_document_structure_class_str {
5719     {book}}{
5720       \int_set:Nn \l_document_structure_section_level_int {0}
5721     }
5722     {report}}{
5723       \int_set:Nn \l_document_structure_section_level_int {0}
5724     }
5725   }{
5726     \int_set:Nn \l_document_structure_section_level_int {2}
5727   }
5728 }

```

## 38.6 Document Structure

The structure of the document is given by the `omgroup` environment just like in OMDoc. The hierarchy is adjusted automatically according to the  $\text{\LaTeX}$  class in effect.

`\currentsectionlevel` For the `\currentsectionlevel` and `\Currentsectionlevel` macros we use an internal macro `\current@section@level` that only contains the keyword (no markup). We initialize it with “document” as a default. In the generated OMDoc, we only generate a text element of class `omdoc_currentsectionlevel`, which will be instantiated by CSS later.<sup>18</sup>

EdN:18

```
5729 \def\current@section@level{document}%
5730 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
5731 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
5732 \cs_new_protected:Npn \skipomgroup {
5733   \ifcase\l_document_structure_section_level_int
5734   \or\stepcounter{part}
5735   \or\stepcounter{chapter}
5736   \or\stepcounter{section}
5737   \or\stepcounter{subsection}
5738   \or\stepcounter{subsubsection}
5739   \or\stepcounter{paragraph}
5740   \or\stepcounter{subparagraph}
5741   \fi
5742 }
```

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

`blindomgroup`

```
5743 \newcommand\at@begin@blindomgroup[1]{%
5744 \newenvironment{blindomgroup}
5745 {
5746   \int_incr:N\l_document_structure_section_level_int
5747   \at@begin@blindomgroup\l_document_structure_section_level_int
5748 }{}}
```

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

```
5749 \newcommand\omgroup@nonum[2]{
5750   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
5751   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
5752 }
```

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

`\omgroup@num` convenience macro: `\omgroup@num{<level>}{<title>}` makes numbered sectioning with title `<title>` at level `<level>`. We have to check the `short` key was given in the `omgroup` environment and – if it is use it. But how to do that depends on whether the `rdfmata` package has been loaded. In the end we call `\sref@label@id` to enable crossreferencing.

```
5753 \newcommand\omgroup@num[2]{
```

---

<sup>18</sup>EDNOTE: MK: we may have to experiment with the more powerful uppercasing macro from `mfirstuc.sty` once we internationalize.



```

5754 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
5755   \@nameuse{#1}{#2}
5756 }{
5757   \cs_if_exist:NTF\rdfmata@sectioning{
5758     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
5759   }{
5760     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
5761   }
5762 }
5763 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
5764 }

```

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

omgroup

```

5765 \keys_define:nn { document-structure / omgroup }{
5766   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
5767   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
5768   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
5769   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
5770   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
5771   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
5772   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
5773   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
5774   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
5775   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
5776 }
5777 \cs_new_protected:Nn \l__document_structure_omgroup_args:n {
5778   \str_clear:N \l__document_structure_omgroup_id_str
5779   \str_clear:N \l__document_structure_omgroup_date_str
5780   \clist_clear:N \l__document_structure_omgroup_creators_clist
5781   \clist_clear:N \l__document_structure_omgroup_contributors_clist
5782   \tl_clear:N \l__document_structure_omgroup_srccite_tl
5783   \tl_clear:N \l__document_structure_omgroup_type_tl
5784   \tl_clear:N \l__document_structure_omgroup_short_tl
5785   \tl_clear:N \l__document_structure_omgroup_display_tl
5786   \tl_clear:N \l__document_structure_omgroup_intro_tl
5787   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
5788   \keys_set:nn { document-structure / omgroup } { #1 }
5789 }

```

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

```

5790 \newif\if@mainmatter\@mainmattertrue
5791 \newcommand\at@begin@omgroup[3] [] {}

```

Then we define a helper macro that takes care of the sectioning magic. It comes with its own key/value interface for customization.

```

5792 \keys_define:nn { document-structure / sectioning }{
5793   name .str_set_x:N = \l__document_structure_sect_name_str ,
5794   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
5795   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
5796   clear .default:n = {true} ,
5797   num .bool_set:N = \l__document_structure_sect_num_bool ,

```

```

5798   num      .default:n      = {true}
5799 }
5800 \cs_new_protected:Nn \__document_structure_sect_args:n {
5801   \str_clear:N \l__document_structure_sect_name_str
5802   \str_clear:N \l__document_structure_sect_ref_str
5803   \bool_set_false:N \l__document_structure_sect_clear_bool
5804   \bool_set_false:N \l__document_structure_sect_num_bool
5805   \keys_set:nn { document-structure / sectioning } { #1 }
5806 }
5807 \newcommand\omdoc@sectioning[3][]{
5808   \__document_structure_sect_args:n {#1}
5809   \let\omdoc@sect@name\l__document_structure_sect_name_str
5810   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
5811   \if@mainmatter% numbering not overridden by frontmatter, etc.
5812     \bool_if:NTF \l__document_structure_sect_num_bool {
5813       \omgroup@num{#2}{#3}
5814     }{
5815       \omgroup@nonum{#2}{#3}
5816     }
5817     \def\current@section@level{\omdoc@sect@name}
5818   \else
5819     \omgroup@nonum{#2}{#3}
5820   \fi
5821 }% 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.

```

5822 \newcommand\omgroup@redefine@addtocontents[1]{%
5823 %\edef\__document_structureimport{#1}%
5824 %\@for\@I:=\__document_structureimport\do{%
5825 %\edef\@path{\csname module@\@I @path\endcsname}%
5826 %\@ifundefined{tf@toc}\relax%
5827 %   {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
5828 %\ifx\hyper@anchor\undefined% hyperref.sty loaded?
5829 %\def\addcontentsline##1##2##3{%
5830 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
5831 %\else% hyperref.sty not loaded
5832 %\def\addcontentsline##1##2##3{%
5833 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
5834 %\fi
5835 }% hypreref.sty loaded?

```

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

```

5836 \newenvironment{omgroup}[2][]{% keys, title
5837 {
5838   \__document_structure_omgroup_args:n { #1 }%\sref@target%

```

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

```

5839 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
5840   \omgroup@redefine@addtocontents{
5841     \@ifundefined{module@id}\used@modules%

```

```

5842     %{\@ifundefined{module@module@id @path}{\used@modules}\module@id}
5843   }
5844 }

```

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

```

5845 \int_incr:N\l_document_structure_section_level_int
5846 \ifcase\l_document_structure_section_level_int
5847   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
5848   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
5849   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
5850   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
5851   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
5852   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
5853   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
5854 \fi
5855 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
5856 \str_if_empty:NF \l__document_structure_omgroup_id_str {
5857   \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
5858 }
5859 }% for customization
5860 {}

```

and finally, we localize the sections

```

5861 \newcommand\omdoc@part@kw{Part}
5862 \newcommand\omdoc@chapter@kw{Chapter}
5863 \newcommand\omdoc@section@kw{Section}
5864 \newcommand\omdoc@subsection@kw{Subsection}
5865 \newcommand\omdoc@subsubsection@kw{Subsubsection}
5866 \newcommand\omdoc@paragraph@kw{paragraph}
5867 \newcommand\omdoc@subparagraph@kw{subparagraph}

```

## 38.7 Front and Backmatter

Index markup is provided by the `omtext` package [Koh20c], so in the `document-structure` package we only need to supply the corresponding `\printindex` command, if it is not already defined

`\printindex`

```

5868 \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).

```

5869 \cs_if_exist:NTF\frontmatter{
5870   \let\__document_structure_orig_frontmatter\frontmatter
5871   \let\frontmatter\relax
5872 }{
5873   \tl_set:Nn\__document_structure_orig_frontmatter{
5874     \clearpage
5875     \@mainmatterfalse
5876     \pagenumbering{roman}

```

```

5877 }
5878 }
5879 \cs_if_exist:NTF\backmatter{
5880   \let\__document_structure_orig_backmatter\backmatter
5881   \let\backmatter\relax
5882 }{
5883   \tl_set:Nn\__document_structure_orig_backmatter{
5884     \clearpage
5885     \@mainmatterfalse
5886     \pagenumbering{roman}
5887   }
5888 }

```

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.

```

5889 \newenvironment{frontmatter}{
5890   \__document_structure_orig_frontmatter
5891 }{
5892   \cs_if_exist:NTF\mainmatter{
5893     \mainmatter
5894   }{
5895     \clearpage
5896     \@mainmattertrue
5897     \pagenumbering{arabic}
5898   }
5899 }

```

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

```

5900 \newenvironment{backmatter}{
5901   \__document_structure_orig_backmatter
5902 }{
5903   \cs_if_exist:NTF\mainmatter{
5904     \mainmatter
5905   }{
5906     \clearpage
5907     \@mainmattertrue
5908     \pagenumbering{arabic}
5909   }
5910 }

```

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

```

5911 \@mainmattertrue\pagenumbering{arabic}

```

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

```

5912 \def \c__document_structure_document_str{document}
5913 \newcommand\afterprematurestop{}
5914 \def\prematurestop@endomgroup{
5915   \unless\ifx\@currenvir\c__document_structure_document_str
5916     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter\expandafter\expandafter
5917     \expandafter\prematurestop@endomgroup

```

```

5918 \fi
5919 }
5920 \providecommand\prematurestop{
5921 \message{Stopping~sTeX~processing~prematurely}
5922 \prematurestop@endomgroup
5923 \afterprematurestop
5924 \end{document}
5925 }

```

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

## 38.8 Global Variables

**\setSGvar** set a global variable

```

5926 \RequirePackage{etoolbox}
5927 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

**\useSGvar** use a global variable

```

5928 \newrobustcmd\useSGvar[1]{%
5929 \@ifundefined{sTeX@Gvar@#1}
5930 {\PackageError{document-structure}
5931 {The sTeX Global variable #1 is undefined}
5932 {set it with \protect\setSGvar}}
5933 \@nameuse{sTeX@Gvar@#1}}

```

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

**\ifSGvar** execute something conditionally based on the state of the global variable.

```

5934 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
5935 \@ifundefined{sTeX@Gvar@#1}
5936 {\PackageError{document-structure}
5937 {The sTeX Global variable #1 is undefined}
5938 {set it with \protect\setSGvar}}
5939 {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}

```

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

## Chapter 39

# NotesSlides – Implementation

### 39.1 Class and Package Options

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

```
5940 \*cls)
5941 \@@=notesslides)
5942 \ProvidesExplClass{notesslides}{2022/02/10}{3.0}{notesslides Class}
5943 \RequirePackage{13keys2e}
5944
5945 \keys_define:nn{notesslides / cls}{
5946   class .code:n = {
5947     \PassOptionsToClass{\CurrentOption}{document-structure}
5948     \str_if_eq:nnT{#1}{book}{
5949       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5950     }
5951     \str_if_eq:nnT{#1}{report}{
5952       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5953     }
5954   },
5955   notes .bool_set:N = \c__notesslides_notes_bool ,
5956   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
5957   unknown .code:n = {
5958     \PassOptionsToClass{\CurrentOption}{document-structure}
5959     \PassOptionsToClass{\CurrentOption}{beamer}
5960     \PassOptionsToPackage{\CurrentOption}{notesslides}
5961   }
5962 }
5963 \ProcessKeysOptions{ notesslides / cls }
5964 \bool_if:NTF \c__notesslides_notes_bool {
5965   \PassOptionsToPackage{notes=true}{notesslides}
5966 }{
5967   \PassOptionsToPackage{notes=false}{notesslides}
5968 }
5969 \</cls)
```

now we do the same for the notesslides package.

```

5970 <*package>
5971 \ProvidesExplPackage{notesslides}{2022/02/10}{3.0}{notesslides Package}
5972 \RequirePackage{13keys2e}
5973
5974 \keys_define:nn{notesslides / pkg}{
5975   topsect      .str_set_x:N = \c__notesslides_topsect_str,
5976   defaulttopsect .str_set_x:N = \c__notesslides_defaulttopsec_str,
5977   notes        .bool_set:N = \c__notesslides_notes_bool ,
5978   slides        .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
5979   sectocframes  .bool_set:N = \c__notesslides_sectocframes_bool ,
5980   frameimages   .bool_set:N = \c__notesslides_frameimages_bool ,
5981   fiboxed        .bool_set:N = \c__notesslides_fiboxed_bool ,
5982   noproblems     .bool_set:N = \c__notesslides_noproblems_bool,
5983   unknown        .code:n      = {
5984     \PassOptionsToClass{\CurrentOption}{stex}
5985     \PassOptionsToClass{\CurrentOption}{tikzinput}
5986   }
5987 }
5988 \ProcessKeysOptions{ notesslides / pkg }
5989 \newif\ifnotes
5990 \bool_if:NTF \c__notesslides_notes_bool {
5991   \notesttrue
5992 }{
5993   \notesfalse
5994 }
5995

```

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

```

5996 \str_if_empty:NTF \c__notesslides_topsect_str {
5997   \str_set_eq:NN \__notesslides_topsect \c__notesslides_defaulttopsec_str
5998 }{
5999   \str_set_eq:NN \__notesslides_topsect \c__notesslides_topsect_str
6000 }
6001 </package>

```

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

```

6002 <*cls>
6003 \bool_if:NTF \c__notesslides_notes_bool {
6004   \LoadClass{document-structure}
6005 }{
6006   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
6007   \newcounter{Item}
6008   \newcounter{paragraph}
6009   \newcounter{subparagraph}
6010   \newcounter{Hfootnote}
6011   \RequirePackage{document-structure}
6012 }

```

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

```

6013 \RequirePackage{notesslides}
6014 </cls>

```

In `notes` mode, we also have to make the `beamer`-specific things available to `article` via the `beamerarticle` package. We use options to avoid loading theorem-like environments, since we want to use our own from the `STEX` packages. The first batch of packages we want are loaded on `notesslides.sty`. These are the general ones, we will load the `STEX`-specific ones after we have done some work (e.g. defined the counters `m*`). Only the `stex-logo` package is already needed now for the default theme.

```

6015 \*package>
6016 \bool_if:NT \c__notesslides_notes_bool {
6017   \RequirePackage{a4wide}
6018   \RequirePackage{marginnote}
6019   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
6020   \RequirePackage{mdframed}
6021   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
6022   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
6023 }
6024 \RequirePackage{stex-tikzinput}
6025 \RequirePackage{etoolbox}
6026 \RequirePackage{amssymb}
6027 \RequirePackage{amsmath}
6028 \RequirePackage{comment}
6029 \RequirePackage{textcomp}
6030 \RequirePackage{url}
6031 \RequirePackage{graphicx}
6032 \RequirePackage{pgf}

```

## 39.2 Notes and Slides

For the lecture notes cases, we also provide the `\usetheme` macro that would otherwise come from the `beamer` class. While the latter loads `beamertheme<theme>.sty`, the notes version loads `beamernotestheme<theme>.sty`.<sup>19</sup>

```

6033 \bool_if:NT \c__notesslides_notes_bool {
6034   \renewcommand\usetheme[2][\usepackage[#1]{beamernotestheme#2}]
6035 }

```

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

```

6036 \newcounter{slide}
6037 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
6038 \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.

```

6039 \bool_if:NTF \c__notesslides_notes_bool {
6040   \renewenvironment{note}{\ignorespaces}{}
6041 }{
6042   \excludecomment{note}
6043 }

```

---

<sup>19</sup>EDNOTE: MK: This is not ideal, but I am not sure that I want to be able to provide the full theme functionality there.



We first set up the slide boxes in `article` mode. We set up sizes and provide a box register for the frames and a counter for the slides.

```
6044 \bool_if:NT \c__notesslides_notes_bool {
6045   \newlength{\slideframewidth}
6046   \setlength{\slideframewidth}{1.5pt}
```

**frame** We first define the keys.

```
6047 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
6048   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
6049     \bool_set_true:N #1
6050   }{
6051     \bool_set_false:N #1
6052   }
6053 }
6054 \keys_define:nn{notesslides / frame}{
6055   label .str_set_x:N = \l__notesslides_frame_label_str,
6056   allowframebreaks .code:n = {
6057     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
6058   },
6059   allowdisplaybreaks .code:n = {
6060     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
6061   },
6062   fragile .code:n = {
6063     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
6064   },
6065   shrink .code:n = {
6066     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
6067   },
6068   squeeze .code:n = {
6069     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
6070   },
6071   t .code:n = {
6072     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
6073   },
6074 }
6075 \cs_new_protected:Nn \__notesslides_frame_args:n {
6076   \str_clear:N \l__notesslides_frame_label_str
6077   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
6078   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
6079   \bool_set_true:N \l__notesslides_frame_fragile_bool
6080   \bool_set_true:N \l__notesslides_frame_shrink_bool
6081   \bool_set_true:N \l__notesslides_frame_squeeze_bool
6082   \bool_set_true:N \l__notesslides_frame_t_bool
6083   \keys_set:nn { notesslides / frame }{ #1 }
6084 }
```

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

```
6085 \renewenvironment{frame}[1][]{
6086   \__notesslides_frame_args:n{#1}
6087   \sffamily
6088   \stepcounter{slide}
6089   \def\@currentlabel{\theslide}
6090   \str_if_empty:NF \l__notesslides_frame_label_str {
6091     \label{\l__notesslides_frame_label_str}
```

6092 }

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

```

6093 \def\itemize@level{outer}
6094 \def\itemize@outer{outer}
6095 \def\itemize@inner{inner}
6096 \renewcommand\newpage{\addtocounter{framenum}{1}}
6097 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
6098 \renewenvironment{itemize}{
6099   \ifx\itemize@level\itemize@outer
6100     \def\itemize@label{$\rhd$}
6101   \fi
6102   \ifx\itemize@level\itemize@inner
6103     \def\itemize@label{$\scriptstyle\rhd$}
6104   \fi
6105   \begin{list}
6106     {\itemize@label}
6107     {\setlength{\labelsep}{.3em}
6108      \setlength{\labelwidth}{.5em}
6109      \setlength{\leftmargin}{1.5em}
6110     }
6111   \edef\itemize@level{\itemize@inner}
6112 }{
6113   \end{list}
6114 }

```

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

```

6115 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth]
6116 }{
6117   \medskip\miko@slidelabel\end{mdframed}
6118 }

```

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

`\frametitle`

```

6119 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
6120 }

```

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

EdN:20

`\pause` 20

```

6121 \bool_if:NT \c__notesslides_notes_bool {
6122   \newcommand\pause{}
6123 }

```

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

`nparagraph`

```

6124 \bool_if:NTF \c__notesslides_notes_bool {
6125   \newenvironment{nparagraph}[1][\begin{sparagraph}[#1]}\end{sparagraph}}
6126 }{
6127   \excludecomment{nparagraph}
6128 }

```

---

<sup>20</sup>EdNOTE: MK: fake it in notes mode for now

```

nomgroup
6129 \bool_if:NTF \c__notesslides_notes_bool {
6130 \newenvironment{nomgroup}[2] [] {\begin{omgroup}[#1]{#2}}{\end{omgroup}}
6131 }{
6132 \excludecomment{nomgroup}
6133 }

ntheorem
6134 \bool_if:NTF \c__notesslides_notes_bool {
6135 \newenvironment{ntheorem}[1] [] {\begin{sdefinition}[#1]}{\end{sdefinition}}
6136 }{
6137 \excludecomment{ntheorem}
6138 }

nassertion
6139 \bool_if:NTF \c__notesslides_notes_bool {
6140 \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}{\end{sassertion}}
6141 }{
6142 \excludecomment{nassertion}
6143 }

nsproof
6144 \bool_if:NTF \c__notesslides_notes_bool {
6145 \newenvironment{nsproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
6146 }{
6147 \excludecomment{nsproof}
6148 }

nexample
6149 \bool_if:NTF \c__notesslides_notes_bool {
6150 \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}{\end{sexample}}
6151 }{
6152 \excludecomment{nexample}
6153 }

\inputref@*skip We customize the hooks for in \inputref.
6154 \def\inputref@preskip{\smallskip}
6155 \def\inputref@postskip{\medskip}

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

\inputref*
6156 \let\orig@inputref\inputref
6157 \def\inputref{\@ifstar\ninputref\orig@inputref}
6158 \newcommand\ninputref[2] [] {
6159 \bool_if:NT \c__notesslides_notes_bool {
6160 \orig@inputref[#1]{#2}
6161 }
6162 }

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

```

## 39.3 Header and Footer Lines

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

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

```

6163 \newlength{\slidelogoheight}
6164
6165 \bool_if:NTF \c__notesslides_notes_bool {
6166   \setlength{\slidelogoheight}{.4cm}
6167 }{
6168   \setlength{\slidelogoheight}{1cm}
6169 }
6170 \newsavebox{\slidelogo}
6171 \sbox{\slidelogo}{\TeX}
6172 \newrobustcmd{\setslidelogo}[1]{
6173   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
6174 }
```

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

**\setsource** `\source` stores the writer's name. By default it is *Michael Kohlhase* since he is the main user and designer of this package. `\setsource{<name>}` can change the writer's name.

```

6175 \def\source{Michael Kohlhase}% customize locally
6176 \newrobustcmd{\setsource}[1]{\def\source{#1}}
```

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

**\setlicensing** Now, we set up the copyright and licensing. By default we use the Creative Commons Attribution-ShareAlike license to strengthen the public domain. If package `hyperref` is loaded, then we can attach a hyperlink to the license logo. `\setlicensing[<url>]{<logo name>}` is used for customization, where `<url>` is optional.

```

6177 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
6178 \newsavebox{\cclogo}
6179 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
6180 \newif\ifcchref\cchreffalse
6181 \AtBeginDocument{
6182   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
6183 }
6184 \def\licensing{
6185   \ifcchref
6186     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
6187   \else
6188     {\usebox{\cclogo}}
6189   \fi
6190 }
6191 \newrobustcmd{\setlicensing}[2][]{
6192   \def@url{#1}
6193   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
6194   \ifx@url@empty
6195     \def\licensing{{\usebox{\cclogo}}}
6196   \else
6197     \def\licensing{
```

```

6198     \ifcchref
6199     \href{#1}{\usebox{\cclogo}}
6200     \else
6201     {\usebox{\cclogo}}
6202     \fi
6203   }
6204 \fi
6205 }

```

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

EdN:21 `\slidelabel` Now, we set up the slide label for the article mode.<sup>21</sup>

```

6206 \newrobustcmd\miko@slidelabel{
6207   \vbox to \slidelogoheight{
6208     \vss\hbox to \slidewidth
6209     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
6210   }
6211 }

```

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

## 39.4 Frame Images

`\frameimage` We have to make sure that the width is overwritten, for that we check the `\Gin@ewidth` macro from the `graphicx` package. We also add the `label` key.

```

6212 \def\Gin@mhrepos{}
6213 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
6214 \define@key{Gin}{label}{\def\currentlabel{\arabic{slide}}\label{#1}}
6215 \newrobustcmd\frameimage[2][]{
6216   \stepcounter{slide}
6217   \bool_if:NT \c__notesslides_frameimages_bool {
6218     \def\Gin@ewidth{}\setkeys{Gin}{#1}
6219     \bool_if:NF \c__notesslides_notes_bool { \vfill }
6220     \begin{center}
6221       \bool_if:NTF \c__notesslides_fiboxed_bool {
6222         \fbox{
6223           \ifx\Gin@ewidth\@empty
6224             \ifx\Gin@mhrepos\@empty
6225               \mhgraphics[width=\slidewidth,#1]{#2}
6226             \else
6227               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
6228             \fi
6229           \else% Gin@ewidth empty
6230             \ifx\Gin@mhrepos\@empty
6231               \mhgraphics[#1]{#2}
6232             \else
6233               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
6234             \fi
6235           \fi% Gin@ewidth empty
6236         }
6237       }{
6238         \ifx\Gin@ewidth\@empty

```

---

<sup>21</sup>EdNOTE: see that we can use the themes for the slides some day. This is all fake.

```

6239         \ifx\Gin@mhrepos\empty
6240             \mhgraphics[width=\slidewidth,#1]{#2}
6241         \else
6242             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
6243         \fi
6244         \ifx\Gin@mhrepos\empty
6245             \mhgraphics[#1]{#2}
6246         \else
6247             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
6248         \fi
6249     \fi% Gin@ewidth empty
6250 }
6251 \end{center}
6252 \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
6253 \bool_if:NF \c__notesslides_notes_bool { \vfill }
6254 }
6255 } % ifmks@sty@frameimages

```

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

## 39.5 Colors and Highlighting

We first specify sans serif fonts as the default.

```

6256 \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

```

6257 \AddToHook{begindocument}{
6258     \definecolor{green}{rgb}{0,.5,0}
6259     \definecolor{purple}{cmyk}{.3,1,0,.17}
6260 }

```

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`.

```

6261 % \def\STpresent#1{\textcolor{blue}{#1}}
6262 \def\defemph#1{\textcolor{magenta}{#1}}
6263 \def\symrefemph#1{\textcolor{cyan}{#1}}
6264 \def\compemph#1{\textcolor{blue}{#1}}
6265 \def\titleemph#1{\textcolor{blue}{#1}}
6266 \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.

```

6267 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
6268 \def\smalltextwarning{
6269     \pgfuseimage{miko@small@dbend}
6270     \xspace
6271 }
6272 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}

```

```

6273 \newrobustcmd\textwarning{
6274   \raisebox{-0.05cm}{\pgfuseimage{miko@dbend}}
6275   \xspace
6276 }
6277 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}
6278 \newrobustcmd\bigtextwarning{
6279   \raisebox{-0.05cm}{\pgfuseimage{miko@big@dbend}}
6280   \xspace
6281 }

```

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

```

6282 \newrobustcmd\putgraphicsat[3]{
6283   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
6284 }
6285 \newrobustcmd\putat[2]{
6286   \begin{picture}(0,0)\put(#1){#2}\end{picture}
6287 }

```

## 39.6 Sectioning

If the `sectocframes` option is set, then we make section frames. We first define counters for `part` and `chapter`, which `beamer.cls` does not have and we make the `section` counter which it does dependent on `chapter`.

```

6288 \bool_if:NT \c__notesslides_sectocframes_bool {
6289   \str_if_eq:VnTF \__notesslidesstopsect{part}{
6290     \newcounter{chapter}\counterwithin*{section}{chapter}
6291   }{
6292     \str_if_eq:VnT\__notesslidesstopsect{chapter}{
6293       \newcounter{chapter}\counterwithin*{section}{chapter}
6294     }
6295   }
6296 }

```

`\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
6297 \def\part@prefix{}
6298 \@ifpackageloaded{document-structure}{
6299   \str_case:VnF \__notesslidesstopsect {
6300     {part}{
6301       \int_set:Nn \l_document_structure_section_level_int {0}
6302       \def\thesection{\arabic{chapter}.\arabic{section}}
6303       \def\part@prefix{\arabic{chapter}.}
6304     }
6305     {chapter}{
6306       \int_set:Nn \l_document_structure_section_level_int {1}
6307       \def\thesection{\arabic{chapter}.\arabic{section}}
6308       \def\part@prefix{\arabic{chapter}.}
6309     }
6310   }{
6311     \int_set:Nn \l_document_structure_section_level_int {2}
6312     \def\part@prefix{}

```

```

6313 }
6314 }
6315
6316 \bool_if:NF \c__notesslides_notes_bool { % only in slides

```

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

The new counters are used in the omgroup environment that choses the L<sup>A</sup>T<sub>E</sub>X sectioning macros according to \section@level.

omgroup

```

6317 \renewenvironment{omgroup}[2][]{
6318   \__document_structure_omgroup_args:n { #1 }
6319   \int_incr:N \l_document_structure_section_level_int
6320   \bool_if:NT \c__notesslides_sectocframes_bool {
6321     \stepcounter{slide}
6322     \begin{frame}[noframenumbering]
6323     \vfill\Large\centering
6324     \red{
6325       \ifcase\l_document_structure_section_level_int\or
6326         \stepcounter{part}
6327         \def\__notesslideslabel{\omdoc@part@kw~\Roman{part}}
6328         \def\currentsectionlevel{\omdoc@part@kw}
6329       \or
6330         \stepcounter{chapter}
6331         \def\__notesslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
6332         \def\currentsectionlevel{\omdoc@chapter@kw}
6333       \or
6334         \stepcounter{section}
6335         \def\__notesslideslabel{\part@prefix\arabic{section}}
6336         \def\currentsectionlevel{\omdoc@section@kw}
6337       \or
6338         \stepcounter{subsection}
6339         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
6340         \def\currentsectionlevel{\omdoc@subsection@kw}
6341       \or
6342         \stepcounter{subsubsection}
6343         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
6344         \def\currentsectionlevel{\omdoc@subsubsection@kw}
6345       \or
6346         \stepcounter{paragraph}
6347         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}.\arabic{paragraph}}
6348         \def\currentsectionlevel{\omdoc@paragraph@kw}
6349       \else
6350         \def\__notesslideslabel{}
6351         \def\currentsectionlevel{\omdoc@paragraph@kw}
6352       \fi% end ifcase
6353       \__notesslideslabel%\sref@label@id\__notesslideslabel
6354       \quad #2%
6355     }%
6356     \vfill%
6357     \end{frame}%
6358   }
6359   \str_if_empty:NF \l__document_structure_omgroup_id_str {
6360     \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str

```



```

6361     }
6362   }{}
6363 }

```

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

```

6364 \def\inserttheorembodyfont{\normalfont}
6365 %\bool_if:NF \c__notesslides_notes_bool {
6366 %   \defbeamertemplate{theorem begin}{miko}
6367 %   {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
6368 %     \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
6369 %     \inserttheorempunctuation\inserttheorembodyfont\xspace}
6370 %   \defbeamertemplate{theorem end}{miko}{}

```

and we set it as the default one.

```

6371 % \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.

```

6372 % \expandafter\def\csname Parent2\endcsname{}
6373 %}
6374
6375 \AddToHook{begindocument}{ % this does not work for some reasons
6376   \setbeamertemplate{theorems}[ams style]
6377 }
6378 \bool_if:NT \c__notesslides_notes_bool {
6379   \renewenvironment{columns}[1][1]{%
6380     \par\noindent%
6381     \begin{minipage}%
6382       \slidewidth\centering\leavevmode%
6383   }{%
6384     \end{minipage}\par\noindent%
6385   }%
6386   \newsavebox\columnbox%
6387   \renewenvironment<>{column}[2][1]{%
6388     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
6389   }{%
6390     \end{minipage}\end{lrbox}\usebox\columnbox%
6391   }%
6392 }
6393 \bool_if:NTF \c__notesslides_noproblems_bool {
6394   \newenvironment{problems}{}{}
6395 }{
6396   \excludecomment{problems}
6397 }

```

## 39.7 Excursions

`\excursion` The excursion macros are very simple, we define a new internal macro `\excursionref` and use it in `\excursion`, which is just an `\inputref` that checks if the new macro is defined before formatting the file in the argument.

```

6398 \gdef\printexcursions{}
6399 \newcommand\excursionref[2]{% label, text

```

```

6400 \bool_if:NT \c__notesslides_notes_bool {
6401   \begin{sparagraph}[title=Excursion]
6402     #2 \sref[fallback=the appendix]{#1}.
6403   \end{sparagraph}
6404 }
6405 }
6406 \newcommand\activate@excursion[2][] {
6407   \gappto\printexcursions{\inputref{#1}{#2}}
6408 }
6409 \newcommand\excursion[4][] {% repos, label, path, text
6410   \bool_if:NT \c__notesslides_notes_bool {
6411     \activate@excursion[#1]{#3}\excursionref{#2}{#4}
6412   }
6413 }

```

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

#### \excursiongroup

```

6414 \keys_define:nn{notesslides / excursiongroup }{
6415   id          .str_set_x:N = \l__notesslides_excursion_id_str,
6416   intro       .tl_set:N   = \l__notesslides_excursion_intro_tl,
6417   mhrepos     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
6418 }
6419 \cs_new_protected:Nn \__notesslides_excursion_args:n {
6420   \tl_clear:N \l__notesslides_excursion_intro_tl
6421   \str_clear:N \l__notesslides_excursion_id_str
6422   \str_clear:N \l__notesslides_excursion_mhrepos_str
6423   \keys_set:nn {notesslides / excursiongroup }{ #1 }
6424 }
6425 \newcommand\excursiongroup[1][] {
6426   \__notesslides_excursion_args:n{ #1 }
6427   \ifdefempty\printexcursions{}% only if there are excursions
6428   {\begin{note}
6429     \begin{omgroup}[#1]{Excursions}%
6430     \ifdefempty\l__notesslides_excursion_intro_tl{
6431       \inputref[\l__notesslides_excursion_mhrepos_str]{
6432         \l__notesslides_excursion_intro_tl
6433       }
6434     }
6435     \printexcursions%
6436     \end{omgroup}
6437   \end{note}}
6438 }
6439 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
6440 \end{package}

```

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

## Chapter 40

# The Implementation

### 40.1 Package Options

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

```
6441 <*package>
6442 <@@=problems>
6443 \ProvidesExplPackage{problem}{2019/03/20}{1.3}{Semantic Markup for Problems}
6444 \RequirePackage{l3keys2e}
6445
6446 \keys_define:nn { problem / pkg }{
6447   notes      .default:n    = { true },
6448   notes      .bool_set:N   = \c__problems_notes_bool,
6449   gnotes     .default:n    = { true },
6450   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
6451   hints      .default:n    = { true },
6452   hints      .bool_set:N   = \c__problems_hints_bool,
6453   solutions  .default:n    = { true },
6454   solutions  .bool_set:N   = \c__problems_solutions_bool,
6455   pts        .default:n    = { true },
6456   pts        .bool_set:N   = \c__problems_pts_bool,
6457   min        .default:n    = { true },
6458   min        .bool_set:N   = \c__problems_min_bool,
6459   boxed      .default:n    = { true },
6460   boxed      .bool_set:N   = \c__problems_boxed_bool,
6461   unknown    .code:n       = {}
6462 }
6463 \newif\ifsolutions
6464
6465 \ProcessKeysOptions{ problem / pkg }
6466 \bool_if:NTF \c__problems_solutions_bool {
6467   \solutionstrue
6468 }{
6469   \solutionsfalse
6470 }
```

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

```
6471 \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.

```
6472 \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.

```
6473 \def\prob@problem@kw{Problem}
6474 \def\prob@solution@kw{Solution}
6475 \def\prob@hint@kw{Hint}
6476 \def\prob@note@kw{Note}
6477 \def\prob@gnote@kw{Grading}
6478 \def\prob@pt@kw{pt}
6479 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
6480 \AddToHook{begindocument}{
6481   \ltx@ifpackageloaded{babel}{
6482     \makeatletter
6483     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6484     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6485       \input{problem-ngerman.ldf}
6486     }
6487     \clist_if_in:NnT \l_tmpa_clist {finnish}{
6488       \input{problem-finnish.ldf}
6489     }
6490     \clist_if_in:NnT \l_tmpa_clist {french}{
6491       \input{problem-french.ldf}
6492     }
6493     \clist_if_in:NnT \l_tmpa_clist {russian}{
6494       \input{problem-russian.ldf}
6495     }
6496     \makeatother
6497   }{ }
6498 }
```

## 40.2 Problems and Solutions

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

```
6499 \keys_define:nn{ problem / problem }{
6500   id      .str_set:x:N = \l__problems_prob_id_str,
6501   pts     .tl_set:N    = \l__problems_prob_pts_tl,
6502   min     .tl_set:N    = \l__problems_prob_min_tl,
6503   title   .tl_set:N    = \l__problems_prob_title_tl,
6504   type    .tl_set:N    = \l__problems_prob_type_tl,
6505   refnum  .int_set:N    = \l__problems_prob_refnum_int
6506 }
6507 \cs_new_protected:Nn \__problems_prob_args:n {
```

```

6508 \str_clear:N \l__problems_prob_id_str
6509 \tl_clear:N \l__problems_prob_pts_tl
6510 \tl_clear:N \l__problems_prob_min_tl
6511 \tl_clear:N \l__problems_prob_title_tl
6512 \tl_clear:N \l__problems_prob_type_tl
6513 \int_zero_new:N \l__problems_prob_refnum_int
6514 \keys_set:nn { problem / problem }{ #1 }
6515 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
6516   \let\l__problems_prob_refnum_int\undefined
6517 }
6518 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

6519 \newcounter{problem}
6520 \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.

```

6521 \newcommand\prob@label[1]{#1}

```

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

`\prob@number` We consolidate the problem number into a reusable internal macro

```

6522 \newcommand\prob@number{
6523   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
6524     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
6525   }{
6526     \int_if_exist:NTF \l__problems_prob_refnum_int {
6527       \prob@label{\int_use:N \l__problems_prob_refnum_int }
6528     }{
6529       \prob@label\theproblem
6530     }
6531   }
6532 }

```

(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.

```

6533 \newcommand\prob@title[3]{%
6534   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
6535     #2 \l__problems_inclprob_title_tl #3
6536   }{
6537     \tl_if_exist:NTF \l__problems_prob_title_tl {
6538       #2 \l__problems_prob_title_tl #3
6539     }{
6540       #1
6541     }
6542   }
6543 }

```

(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.

```

6544 \def\prob@heading{
6545   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
6546   %\sref@label{id}\prob@problem@kw~\prob@number}{~}
6547 }

```

(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`

```

6548 \newenvironment{sproblem}[1][{}{
6549   \__problems_prob_args:n{#1}%\sref@target%
6550   \@in@omtexttrue% we are in a statement (for inline definitions)
6551   \stepcounter{problem}\record@problem
6552   \def\current@section@level{\prob@problem@kw}
6553   \tl_if_exist:NTF \l__problems_inclprob_type_tl {
6554     \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
6555   }{
6556     \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
6557   }
6558   \str_if_exist:NTF \l__problems_inclprob_id_str {
6559     \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
6560   }{
6561     \str_set_eq:NN \sproblemid \l__problems_prob_id_str
6562   }
6563
6564
6565   \clist_set:No \l_tmpa_clist \sproblemtype
6566   \tl_clear:N \l_tmpa_tl
6567   \clist_map_inline:Nn \l_tmpa_clist {
6568     \tl_if_exist:cT {\__problems_sproblem_##1_start:}{
6569       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_start:}}
6570     }
6571   }
6572   \tl_if_empty:NTF \l_tmpa_tl {
6573     \__problems_sproblem_start:
6574   }{
6575     \l_tmpa_tl
6576   }
6577   \stex_ref_new_doc_target:n \sproblemid
6578 }{
6579   \clist_set:No \l_tmpa_clist \sproblemtype
6580   \tl_clear:N \l_tmpa_tl
6581   \clist_map_inline:Nn \l_tmpa_clist {
6582     \tl_if_exist:cT {\__problems_sproblem_##1_end:}{
6583       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_end:}}
6584     }

```

```

6585 }
6586 \tl_if_empty:NTF \l_tmpa_tl {
6587   \__problems_sproblem_end:
6588 }{
6589   \l_tmpa_tl
6590 }
6591
6592
6593 \smallskip
6594 }
6595
6596
6597 \cs_new_protected:Nn \__problems_sproblem_start: {
6598   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
6599 }
6600 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
6601
6602 \newcommand\stexpatchproblem[3][] {
6603   \str_set:Nx \l_tmpa_str{ #1 }
6604   \str_if_empty:NTF \l_tmpa_str {
6605     \tl_set:Nn \__problems_sproblem_start: { #2 }
6606     \tl_set:Nn \__problems_sproblem_end: { #3 }
6607   }{
6608     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
6609     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
6610   }
6611 }
6612
6613
6614 \bool_if:NT \c__problems_boxed_bool {
6615   \surroundwithmdframed{problem}
6616 }

```

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

```

6617 \def\record@problem{
6618   \protected@write\@auxout{}
6619   {
6620     \string\@problem{\prob@number}
6621     {
6622       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
6623         \l__problems_inclprob_pts_tl
6624       }{
6625         \l__problems_prob_pts_tl
6626       }
6627     }%
6628     {
6629       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
6630         \l__problems_inclprob_min_tl
6631       }{
6632         \l__problems_prob_min_tl
6633       }
6634     }
6635   }
6636 }

```

(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).

```
6637 \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.

```
6638 \keys_define:nn { problem / solution }{
6639   id                .str_set_x:N = \l__problems_solution_id_str ,
6640   for               .tl_set:N    = \l__problems_solution_for_tl ,
6641   height            .dim_set:N   = \l__problems_solution_height_dim ,
6642   creators          .clist_set:N = \l__problems_solution_creators_clist ,
6643   contributors      .clist_set:N = \l__problems_solution_contributors_clist ,
6644   srccite           .tl_set:N    = \l__problems_solution_srccite_tl
6645 }
6646 \cs_new_protected:Nn \__problems_solution_args:n {
6647   \str_clear:N \l__problems_solution_id_str
6648   \tl_clear:N \l__problems_solution_for_tl
6649   \tl_clear:N \l__problems_solution_srccite_tl
6650   \clist_clear:N \l__problems_solution_creators_clist
6651   \clist_clear:N \l__problems_solution_contributors_clist
6652   \dim_zero:N \l__problems_solution_height_dim
6653   \keys_set:nn { problem / solution }{ #1 }
6654 }
```

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

```
6655 \newcommand\@startsolution[1][{}]{
6656   \__problems_solution_args:n { #1 }
6657   \@in@omtexttrue% we are in a statement.
6658   \bool_if:NF \c__problems_boxed_bool { \hrule }
6659   \smallskip\noindent
6660   {\textbf\prob@solution@kw : \enspace}
6661   \begin{small}
6662   \def\current@section@level{\prob@solution@kw}
6663   \ignorespacesandpars
6664 }
```

**\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.

```
6665 \newcommand\startsolutions{
6666   \specialcomment{solution}{\@startsolution}{
6667     \bool_if:NF \c__problems_boxed_bool {
6668       \hrule\medskip
6669     }
6670     \end{small}%
6671   }
6672   \bool_if:NT \c__problems_boxed_bool {
6673     \surroundwithmdframed{solution}
6674   }
6675 }
```



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

\stopsolutions

```
6676 \newcommand\stopsolutions{\excludecomment{solution}}
```

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

so it only remains to start/stop solutions depending on what option was specified.

```
6677 \ifsolutions
6678 \startsolutions
6679 \else
6680 \stopsolutions
6681 \fi
```

exnote

```
6682 \bool_if:NTF \c__problems_notes_bool {
6683   \newenvironment{exnote}[1][]{
6684     \par\smallskip\hrule\smallskip
6685     \noindent\textbf{\prob@note@kw : }\small
6686   }{
6687     \smallskip\hrule
6688   }
6689 }{
6690   \excludecomment{exnote}
6691 }
```

hint

```
6692 \bool_if:NTF \c__problems_notes_bool {
6693   \newenvironment{hint}[1][]{
6694     \par\smallskip\hrule\smallskip
6695     \noindent\textbf{\prob@hint@kw :~ }\small
6696   }{
6697     \smallskip\hrule
6698   }
6699   \newenvironment{exhint}[1][]{
6700     \par\smallskip\hrule\smallskip
6701     \noindent\textbf{\prob@hint@kw :~ }\small
6702   }{
6703     \smallskip\hrule
6704   }
6705 }{
6706   \excludecomment{hint}
6707   \excludecomment{exhint}
6708 }
```

gnote

```
6709 \bool_if:NTF \c__problems_notes_bool {
6710   \newenvironment{gnote}[1][]{
6711     \par\smallskip\hrule\smallskip
6712     \noindent\textbf{\prob@gnote@kw : }\small
6713   }{
6714     \smallskip\hrule
6715   }
6716 }{
6717   \excludecomment{gnote}
6718 }
```

## 40.3 Multiple Choice Blocks

EdN:22

mcb 22

```
6719 \newenvironment{mcb}{
6720   \begin{enumerate}
6721 }{
6722   \end{enumerate}
6723 }
```

we define the keys for the mcc macro

```
6724 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
6725   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
6726     \bool_set_true:N #1
6727   }{
6728     \bool_set_false:N #1
6729   }
6730 }
6731 \keys_define:nn { problem / mcc }{
6732   id          .str_set_x:N = \l__problems_mcc_id_str ,
6733   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
6734   T           .default:n    = { true } ,
6735   T           .bool_set:N    = \l__problems_mcc_t_bool ,
6736   F           .default:n    = { true } ,
6737   F           .bool_set:N    = \l__problems_mcc_f_bool ,
6738   Ttext       .code:n       = {
6739     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
6740   } ,
6741   Ftext       .code:n       = {
6742     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
6743   }
6744 }
6745 \cs_new_protected:Nn \l__problems_mcc_args:n {
6746   \str_clear:N \l__problems_mcc_id_str
6747   \tl_clear:N \l__problems_mcc_feedback_tl
6748   \bool_set_true:N \l__problems_mcc_t_bool
6749   \bool_set_true:N \l__problems_mcc_f_bool
6750   \bool_set_true:N \l__problems_mcc_Ttext_bool
6751   \bool_set_false:N \l__problems_mcc_Ftext_bool
6752   \keys_set:nn { problem / mcc }{ #1 }
6753 }
```

\mcc

```
6754 \newcommand\mcc[2][] {
6755   \l__problems_mcc_args:n{ #1 }
6756   \item #2
6757   \ifsolutions
6758     \\\
6759     \bool_if:NT \l__problems_mcc_t_bool {
6760       % TODO!
6761       % \ifcsstring{mcc@T}{T}{\mcc@Ttext}%
6762     }
6763     \bool_if:NT \l__problems_mcc_f_bool {
```

---

<sup>22</sup>EdNOTE: MK: maybe import something better here from a dedicated MC package

```

6764         % TODO!
6765         % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
6766     }
6767     \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
6768         !
6769     }{
6770         \l__problems_mcc_feedback_tl
6771     }
6772     \fi
6773 } %solutions

```

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

## 40.4 Including Problems

`\includeproblem` The `\includeproblem` command is essentially a glorified `\input` statement, it sets some internal macros first that overwrite the local points. Importantly, it resets the `inclprob` keys after the input.

```

6774
6775 \keys_define:nn{ problem / inclproblem }{
6776   id      .str_set_x:N = \l__problems_inclprob_id_str,
6777   pts     .tl_set:N    = \l__problems_inclprob_pts_tl,
6778   min     .tl_set:N    = \l__problems_inclprob_min_tl,
6779   title   .tl_set:N    = \l__problems_inclprob_title_tl,
6780   refnum  .int_set:N    = \l__problems_inclprob_refnum_int,
6781   type    .tl_set:N    = \l__problems_inclprob_type_tl,
6782   mhrepos .str_set_x:N = \l__problems_inclprob_mhrepos_str
6783 }
6784 \cs_new_protected:Nn \l__problems_inclprob_args:n {
6785   \str_clear:N \l__problems_prob_id_str
6786   \tl_clear:N \l__problems_inclprob_pts_tl
6787   \tl_clear:N \l__problems_inclprob_min_tl
6788   \tl_clear:N \l__problems_inclprob_title_tl
6789   \tl_clear:N \l__problems_inclprob_type_tl
6790   \int_zero_new:N \l__problems_inclprob_refnum_int
6791   \str_clear:N \l__problems_inclprob_mhrepos_str
6792   \keys_set:nn { problem / inclproblem }{ #1 }
6793   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
6794     \let\l__problems_inclprob_pts_tl\undefined
6795   }
6796   \tl_if_empty:NT \l__problems_inclprob_min_tl {
6797     \let\l__problems_inclprob_min_tl\undefined
6798   }
6799   \tl_if_empty:NT \l__problems_inclprob_title_tl {
6800     \let\l__problems_inclprob_title_tl\undefined
6801   }
6802   \tl_if_empty:NT \l__problems_inclprob_type_tl {
6803     \let\l__problems_inclprob_type_tl\undefined
6804   }
6805   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
6806     \let\l__problems_inclprob_refnum_int\undefined
6807   }
6808 }

```

```

6809
6810 \cs_new_protected:Nn \__problems_inclprob_clear: {
6811   \let\l__problems_inclprob_id_str\undefined
6812   \let\l__problems_inclprob_pts_tl\undefined
6813   \let\l__problems_inclprob_min_tl\undefined
6814   \let\l__problems_inclprob_title_tl\undefined
6815   \let\l__problems_inclprob_type_tl\undefined
6816   \let\l__problems_inclprob_refnum_int\undefined
6817   \let\l__problems_inclprob_mhrepos_str\undefined
6818 }
6819 \__problems_inclprob_clear:
6820
6821 \newcommand\includeproblem[2][ ]{
6822   \__problems_inclprob_args:n{ #1 }
6823   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
6824     \input{#2}
6825   }{
6826     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
6827       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
6828     }
6829   }
6830   \__problems_inclprob_clear:
6831 }

```

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

## 40.5 Reporting Metadata

For messages it is OK to have them in English as the whole documentation is, and we can therefore assume authors can deal with it.

```

6832 \AddToHook{enddocument}{
6833   \bool_if:NT \c__problems_pts_bool {
6834     \message{Total:~\arabic{pts}~points}
6835   }
6836   \bool_if:NT \c__problems_min_bool {
6837     \message{Total:~\arabic{min}~minutes}
6838   }
6839 }

```

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

```

6840 \def\pts#1{
6841   \bool_if:NT \c__problems_pts_bool {
6842     \marginpar{#1~\prob@pt@kw}
6843   }
6844 }
6845 \def\min#1{
6846   \bool_if:NT \c__problems_min_bool {
6847     \marginpar{#1~\prob@min@kw}
6848   }
6849 }

```

`\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.

```

6850 \newcounter{pts}
6851 \def\show@pts{
6852   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
6853     \bool_if:NT \c__problems_pts_bool {
6854       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
6855       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
6856     }
6857   }{
6858     \tl_if_exist:NT \l__problems_prob_pts_tl {
6859       \bool_if:NT \c__problems_pts_bool {
6860         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
6861         \addtocounter{pts}{\l__problems_prob_pts_tl}
6862       }
6863     }
6864   }
6865 }

```

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

`\show@min`

```

6866 \newcounter{min}
6867 \def\show@min{
6868   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
6869     \bool_if:NT \c__problems_min_bool {
6870       \marginpar{\l__problems_inclprob_min_tl\ min}
6871       \addtocounter{min}{\l__problems_inclprob_min_tl}
6872     }
6873   }{
6874     \tl_if_exist:NT \l__problems_prob_min_tl {
6875       \bool_if:NT \c__problems_min_bool {
6876         \marginpar{\l__problems_prob_min_tl\ min}
6877         \addtocounter{min}{\l__problems_prob_min_tl}
6878       }
6879     }
6880   }
6881 }
6882 </package>

```

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

## Chapter 41

# Implementation: The hwexam Class

The functionality is spread over the `hwexam` class and package. The class provides the `document` environment and pre-loads some convenience packages, whereas the package provides the concrete functionality.

### 41.1 Class Options

To initialize the `hwexam` class, we declare and process the necessary options by passing them to the respective packages and classes they come from.

```
6883 \@@=hwexam>
6884 \*cls>
6885 \ProvidesExplClass{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
6886 \RequirePackage{l3keys2e}
6887 \DeclareOption*{
6888   \PassOptionsToClass{\CurrentOption}{document-structure}
6889   \PassOptionsToPackage{\CurrentOption}{stex}
6890   \PassOptionsToPackage{\CurrentOption}{hwexam}
6891   \PassOptionsToPackage{\CurrentOption}{tikzinput}
6892 }
6893 \ProcessOptions
```

We load `omdoc.cls`, and the desired packages. For the L<sup>A</sup>T<sub>E</sub>XML bindings, we make sure the right packages are loaded.

```
6894 \LoadClass{document-structure}
6895 \RequirePackage{stex}
6896 \RequirePackage{hwexam}
6897 \RequirePackage{tikzinput}
6898 \RequirePackage{graphicx}
6899 \RequirePackage{a4wide}
6900 \RequirePackage{amssymb}
6901 \RequirePackage{amstext}
6902 \RequirePackage{amsmath}
```

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

```

6903 \newcommand\assig@default@type{\hwexam@assignment@kw}
6904 \def\document@hwexamtype{\assig@default@type}
6905 <@@=document_structure>
6906 \keys_define:nn { document-structure / document }{
6907 id .str_set_x:N = \c_document_structure_document_id_str,
6908 hwexamtype .tl_set:N = \document@hwexamtype
6909 }
6910 <@@=hwexam>
6911 </cls>

```

## Chapter 42

# Implementation: The hwexam Package

### 42.1 Package Options

The first step is to declare (a few) package options that handle whether certain information is printed or not. Some come with their own conditionals that are set by the options, the rest is just passed on to the `problems` package.

```
6912 \*package>
6913 \ProvidesExplPackage{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
6914 \RequirePackage{l3keys2e}
6915
6916 \newif\iftest\testfalse
6917 \DeclareOption{test}{\testtrue}
6918 \newif\ifmultiple\multiplefalse
6919 \DeclareOption{multiple}{\multipletrue}
6920 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
6921 \ProcessOptions
```

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

```
6922 \RequirePackage{keyval}[1997/11/10]
6923 \RequirePackage{problem}
```

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

```
6924 \newcommand\hwexam@assignment@kw{Assignment}
6925 \newcommand\hwexam@given@kw{Given}
6926 \newcommand\hwexam@due@kw{Due}
6927 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
6928 blank~for~extra~space}
6929 \def\hwexam@minutes@kw{minutes}
6930 \newcommand\correction@probs@kw{prob.}
6931 \newcommand\correction@pts@kw{total}
6932 \newcommand\correction@reached@kw{reached}
6933 \newcommand\correction@sum@kw{Sum}
6934 \newcommand\correction@grade@kw{grade}
6935 \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

```

6936 \AddToHook{begindocument}{
6937 \ltx@ifpackageloaded{babel}{
6938 \makeatletter
6939 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6940 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6941 \input{hwexam-ngerman.ldf}
6942 }
6943 \clist_if_in:NnT \l_tmpa_clist {finnish}{
6944 \input{hwexam-finnish.ldf}
6945 }
6946 \clist_if_in:NnT \l_tmpa_clist {french}{
6947 \input{hwexam-french.ldf}
6948 }
6949 \clist_if_in:NnT \l_tmpa_clist {russian}{
6950 \input{hwexam-russian.ldf}
6951 }
6952 \makeatother
6953 }{}
6954 }
6955

```

## 42.2 Assignments

Then we set up a counter for problems and make the problem counter inherited from `problem.sty` depend on it. Furthermore, we specialize the `\prob@label` macro to take the assignment counter into account.

```

6956 \newcounter{assignment}
6957 \numberproblemsin{assignment}
6958 \renewcommand\prob@label[1]{\assignment@number.#1}

```

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

```

6959 \keys_define:nn { hwexam / assignment } {
6960 id .str_set:x:N = \l__hwexam_assign_id_str,
6961 number .int_set:N = \l__hwexam_assign_number_int,
6962 title .tl_set:N = \l__hwexam_assign_title_tl,
6963 type .tl_set:N = \l__hwexam_assign_type_tl,
6964 given .tl_set:N = \l__hwexam_assign_given_tl,
6965 due .tl_set:N = \l__hwexam_assign_due_tl,
6966 loadmodules .code:n = {
6967 \bool_set_true:N \l__hwexam_assign_loadmodules_bool
6968 }
6969 }
6970 \cs_new_protected:Nn \__hwexam_assignment_args:n {
6971 \str_clear:N \l__hwexam_assign_id_str
6972 \int_set:Nn \l__hwexam_assign_number_int {-1}
6973 \tl_clear:N \l__hwexam_assign_title_tl
6974 \tl_clear:N \l__hwexam_assign_type_tl
6975 \tl_clear:N \l__hwexam_assign_given_tl
6976 \tl_clear:N \l__hwexam_assign_due_tl
6977 \bool_set_false:N \l__hwexam_assign_loadmodules_bool

```

```

6978 \keys_set:nn { hwexam / assignment }{ #1 }
6979 }

```

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.

```

6980 \newcommand\given@due[2]{
6981 \bool_lazy_all:nF {
6982 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
6983 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
6984 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
6985 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
6986 }{ #1 }
6987
6988 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
6989 \tl_if_empty:NF \l__hwexam_assign_given_tl {
6990 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
6991 }
6992 }{
6993 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
6994 }
6995
6996 \bool_lazy_or:nnF {
6997 \bool_lazy_and_p:nn {
6998 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
6999 }{
7000 \tl_if_empty_p:V \l__hwexam_assign_due_tl
7001 }
7002 }{
7003 \bool_lazy_and_p:nn {
7004 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
7005 }{
7006 \tl_if_empty_p:V \l__hwexam_assign_due_tl
7007 }
7008 }{ ,~ }
7009
7010 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
7011 \tl_if_empty:NF \l__hwexam_assign_due_tl {
7012 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
7013 }
7014 }{
7015 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
7016 }
7017
7018 \bool_lazy_all:nF {
7019 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
7020 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
7021 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
7022 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
7023 }{ #2 }
7024 }

```

`\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.

```

7025 \newcommand\assignment@title[3]{
7026 \tl_if_empty:NTF \l__hwexam_inclassassign_title_tl {
7027 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
7028 #1
7029 }{
7030 #2\l__hwexam_assign_title_tl#3
7031 }
7032 }{
7033 #2\l__hwexam_inclassassign_title_tl#3
7034 }
7035 }

```

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

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

```

7036 \newcommand\assignment@number{
7037 \int_compare:nNnTF \l__hwexam_inclassassign_number_int = {-1} {
7038 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
7039 \arabic{assignment}
7040 } {
7041 \int_use:N \l__hwexam_assign_number_int
7042 }
7043 }{
7044 \int_use:N \l__hwexam_inclassassign_number_int
7045 }
7046 }

```

(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.

```

7047 \newenvironment{assignment}[1][ ]{
7048 \__hwexam_assignment_args:n { #1 }
7049 %\sref@target
7050 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
7051 \global\stepcounter{assignment}
7052 }{
7053 \global\setcounter{assignment}{\int_use:N\l__hwexam_assign_number_int}
7054 }
7055 \setcounter{problem}{0}
7056 \def\current@section@level{\document@hwexamtype}
7057 %\sref@label@id{\document@hwexamtype \thesection}
7058 \begin{@assignment}
7059 }{
7060 \end{@assignment}
7061 }

```

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

```

7062 \def\ass@title{
7063 \protect\document@hwexamtype~\arabic{assignment}
7064 \assignment@title{}\{;\}{} -- \given@due{}\}{}
7065 }
7066 \ifmultiple
7067 \newenvironment{@assignment}{
7068 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
7069 \begin{omgroup}[loadmodules]{\ass@title}
7070 }{
7071 \begin{omgroup}{\ass@title}
7072 }
7073 }{
7074 \end{omgroup}
7075 }

```

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

```

7076 \else
7077 \newenvironment{@assignment}{
7078 \begin{center}\bf
7079 \Large@title\strut\\
7080 \document@hwexamtype~\arabic{assignment}\assignment@title{}\{;\}{}{\}{}
7081 \large\given@due{--;\}{}{;\}{}
7082 \end{center}
7083 }{}
7084 \fi% multiple

```

## 42.3 Including Assignments

**\in\*assignment** This macro is essentially a glorified `\include` statement, it just sets some internal macros first that overwrite the local points. Importantly, it resets the `inclassig` keys after the input.

```

7085 \keys_define:nn { hwexam / inclassignment } {
7086 %id .str_set_x:N = \l__hwexam_assign_id_str,
7087 number .int_set:N = \l__hwexam_inclassign_number_int,
7088 title .tl_set:N = \l__hwexam_inclassign_title_tl,
7089 type .tl_set:N = \l__hwexam_inclassign_type_tl,
7090 given .tl_set:N = \l__hwexam_inclassign_given_tl,
7091 due .tl_set:N = \l__hwexam_inclassign_due_tl,
7092 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
7093 }
7094 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
7095 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
7096 \tl_clear:N \l__hwexam_inclassign_title_tl
7097 \tl_clear:N \l__hwexam_inclassign_type_tl
7098 \tl_clear:N \l__hwexam_inclassign_given_tl
7099 \tl_clear:N \l__hwexam_inclassign_due_tl
7100 \str_clear:N \l__hwexam_inclassign_mhrepos_str
7101 \keys_set:nn { hwexam / inclassignment }{ #1 }
7102 }
7103 \__hwexam_inclassignment_args:n {}
7104
7105 \newcommand\inputassignment[2][{}]{

```

```

7106 \_hwexam_inclassnment_args:n { #1 }
7107 \str_if_empty:NTF \l__hwexam_inclassn_mhrepos_str {
7108   \input{#2}
7109 }{
7110   \stex_in_repository:nn{\l__hwexam_inclassn_mhrepos_str}{
7111     \input{\mhp{path}\l__hwexam_inclassn_mhrepos_str}{#2}}
7112   }
7113 }
7114 \_hwexam_inclassnment_args:n {}
7115 }
7116 \newcommand\includeassignment[2][]{
7117   \newpage
7118   \inputassignment[#1]{#2}
7119 }

```

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

## 42.4 Typesetting Exams

\quizheading

```

7120 \ExplSyntaxOff
7121 \newcommand\quizheading[1]{%
7122   \def\@tas{#1}%
7123   \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
7124   \ifx\@tas\@empty\else%
7125     \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
7126   \fi%
7127 }
7128 \ExplSyntaxOn

```

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

\testheading

```

7129
7130 \def\hwexamheader{\input{hwexam-default.header}}
7131
7132 \def\hwexamminutes{
7133   \tl_if_empty:NTF \testheading@duration {
7134     {\testheading@min}~\hwexam@minutes@kw
7135   }{
7136     \testheading@duration
7137   }
7138 }
7139
7140 \keys_define:nn { hwexam / testheading } {
7141   min .tl_set:N = \testheading@min,
7142   duration .tl_set:N = \testheading@duration,
7143   reqpts .tl_set:N = \testheading@reqpts,
7144   tools .tl_set:N = \testheading@tools
7145 }
7146 \cs_new_protected:Nn \_hwexam_testheading_args:n {
7147   \tl_clear:N \testheading@min
7148   \tl_clear:N \testheading@duration

```

```

7149 \tl_clear:N \testheading@reqpts
7150 \tl_clear:N \testheading@tools
7151 \keys_set:nn { hwexam / testheading }{ #1 }
7152 }
7153 \newenvironment{testheading}[1][]{
7154   \_hwexam_testheading_args:n{ #1 }
7155   \newcount\check@time\check@time=\testheading@min
7156   \advance\check@time by -\theassignment@totalmin
7157   \newif\if@bonuspoints
7158   \tl_if_empty:NTF \testheading@reqpts {
7159     \@bonuspointsfalse
7160   }{
7161     \newcount\bonus@pts
7162     \bonus@pts=\theassignment@totalpts
7163     \advance\bonus@pts by -\testheading@reqpts
7164     \edef\bonus@pts{\the\bonus@pts}
7165     \@bonuspointstrue
7166   }
7167   \edef\check@time{\the\check@time}
7168
7169   \makeatletter\hwexamheader\makeatother
7170 }{
7171   \newpage
7172 }

```

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

**\testspace**

```

7173 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

**\testnewpage**

```

7174 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

**\testemptypage**

```

7175 \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.

```

7176 <@=problems>
7177 \renewcommand\@problem[3]{
7178   \stepcounter{assignment@probs}
7179   \def\__problemspts{#2}
7180   \ifx\__problemspts\@empty\else
7181     \addtocounter{assignment@totalpts}{#2}
7182   \fi
7183   \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
7184   \xdef\correction@probs{\correction@probs & #1}%
7185   \xdef\correction@pts{\correction@pts & #2}
7186   \xdef\correction@reached{\correction@reached &}

```

```

7187 }
7188 <@@=hwexam>

```

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

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

```

7189 \newcounter{assignment@probs}
7190 \newcounter{assignment@totalpts}
7191 \newcounter{assignment@totalmin}
7192 \def\correction@probs{\correction@probs@kw}
7193 \def\correction@pts{\correction@pts@kw}
7194 \def\correction@reached{\correction@reached@kw}
7195 \stepcounter{assignment@probs}
7196 \newcommand\correction@table{
7197 \resizebox{\textwidth}{!}{%
7198 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
7199 &\multicolumn{\theassignment@probs}{c|}|%|
7200 {\footnotesize\correction@forgrading@kw} &\\ \hline
7201 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
7202 \correction@pts & \theassignment@totalpts & \\ \hline
7203 \correction@reached & & \[.7cm]\hline
7204 \end{tabular}}
7205 \end{package}

```

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

## 42.5 Leftovers

at some point, we may want to reactivate the logos font, then we use

here we define the logos that characterize the assignment

```

\font\bierfont=../assignments/bierglas
\font\denkerfont=../assignments/denker
\font\uhrfont=../assignments/uhr
\font\warnschildfont=../assignments/achtung

\newcommand\bierglas{{\bierfont\char65}}
\newcommand\denker{{\denkerfont\char65}}
\newcommand\uhr{{\uhrfont\char65}}
\newcommand\warnschild{{\warnschildfont\char 65}}
\newcommand\hardA{\warnschild}
\newcommand\longA{\uhr}
\newcommand\thinkA{\denker}
\newcommand\discussA{\bierglas}

```