

# The sTeX3 Package Collection \*

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

2022-07-21

## Abstract

sTeX is a collection of L<sup>A</sup>T<sub>E</sub>X packages that allow to markup documents semantically without leaving the document format.

Running ‘pdflatex’ over sTeX-annotated documents formats them into normal-looking PDF. But sTeX also comes with a conversion pipeline into semantically annotated HTML5, which can host semantic added-value services that make the documents active (i.e. interactive and user-adaptive) and essentially turning L<sup>A</sup>T<sub>E</sub>X into a document format for (mathematical) knowledge management (MKM).

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

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

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

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

---

\*Version 3.1 (last revised 2022-07-21)

# Contents

<b>I</b>	<b>Manual</b>	<b>1</b>
<b>1</b>	<b>What is sTeX?</b>	<b>2</b>
<b>2</b>	<b>Quickstart</b>	<b>3</b>
2.1	Setup	3
2.1.1	Minimal Setup for the PDF-only Workflow	3
2.1.2	GIT-based Setup for the sTeX Development Version	3
2.1.3	sTeX Archives (Manual Setup)	4
2.1.4	The sTeX IDE	4
2.1.5	Manual Setup for Active Documents and Knowledge Management Services	4
2.2	A First sTeX Document	5
2.2.1	OMDoc/xhtml Conversion	8
2.2.2	MMT/OMDoc Conversion	9
<b>3</b>	<b>Creating sTeX Content</b>	<b>10</b>
3.1	How Knowledge is Organized in sTeX	10
3.2	sTeX Archives	11
3.2.1	The Local MathHub-Directory	11
3.2.2	The Structure of sTeX Archives	12
3.2.3	MANIFEST.MF-Files	12
3.2.4	Using Files in sTeX Archives Directly	13
3.3	Module, Symbol and Notation Declarations	14
3.3.1	The smodule-Environment	14
3.3.2	Declaring New Symbols and Notations	16
	Operator Notations	19
3.3.3	Argument Modes	20
	Mode-b Arguments	20
	Mode-a Arguments	21
	Mode-B Arguments	22
3.3.4	Type and Definiens Components	23
3.3.5	Precedences and Automated Bracketing	24
3.3.6	Variables	26
3.3.7	Variable Sequences	27
3.4	Module Inheritance and Structures	28
3.4.1	Multilinguality and Translations	28
3.4.2	Simple Inheritance and Namespaces	29
3.4.3	The mathstructure Environment	31
3.4.4	The copymodule Environment	34
3.4.5	The interpretmodule Environment	35
3.5	Primitive Symbols (The sTeX Metatheory)	36
<b>4</b>	<b>Using sTeX Symbols</b>	<b>37</b>
4.1	\symref and its variants	37
4.2	Marking Up Text and On-the-Fly Notations	38
4.3	Referencing Symbols and Statements	40

<b>5</b>	<b><code>sTeX</code> Statements</b>	<b>41</b>
5.1	Definitions, Theorems, Examples, Paragraphs . . . . .	41
5.2	Proofs . . . . .	44
5.3	Highlighting and Presentation Customizations . . . . .	49
<b>6</b>	<b>Additional Packages</b>	<b>51</b>
6.1	Tikzinput: Treating TIKZ code as images . . . . .	51
6.2	Modular Document Structuring . . . . .	52
6.2.1	Introduction . . . . .	52
6.2.2	Package Options . . . . .	52
6.2.3	Document Fragments . . . . .	52
6.2.4	Ending Documents Prematurely . . . . .	54
6.2.5	Global Document Variables . . . . .	54
6.3	Slides and Course Notes . . . . .	54
6.3.1	Introduction . . . . .	54
6.3.2	Package Options . . . . .	55
6.3.3	Notes and Slides . . . . .	55
6.3.4	Customizing Header and Footer Lines . . . . .	56
6.3.5	Frame Images . . . . .	57
6.3.6	Excursions . . . . .	58
6.4	Representing Problems and Solutions . . . . .	59
6.4.1	Introduction . . . . .	59
6.4.2	Problems and Solutions . . . . .	59
6.4.3	Markup for Added-Value Services . . . . .	61
	Multiple Choice Blocks . . . . .	61
	Filling-In Concrete Solutions . . . . .	62
6.4.4	Including Problems . . . . .	63
6.5	Homeworks, Quizzes and Exams . . . . .	64
6.5.1	Introduction . . . . .	64
6.5.2	Package Options . . . . .	64
6.5.3	Assignments . . . . .	64
6.5.4	Including Assignments . . . . .	65
6.5.5	Typesetting Exams . . . . .	65
<b>II</b>	<b>Documentation</b>	<b>67</b>
<b>7</b>	<b><code>sTeX</code>-Basics</b>	<b>68</b>
7.1	Macros and Environments . . . . .	68
7.1.1	HTML Annotations . . . . .	68
7.1.2	Babel Languages . . . . .	69
7.1.3	Auxiliary Methods . . . . .	69
<b>8</b>	<b><code>sTeX</code>-MathHub</b>	<b>70</b>
8.1	Macros and Environments . . . . .	70
8.1.1	Files, Paths, URIs . . . . .	70
8.1.2	MathHub Archives . . . . .	71
8.1.3	Using Content in Archives . . . . .	72

<b>9</b>	<b>sTeX-References</b>	<b>73</b>
9.1	Macros and Environments . . . . .	73
9.1.1	Setting Reference Targets . . . . .	73
9.1.2	Using References . . . . .	74
<b>10</b>	<b>sTeX-Modules</b>	<b>75</b>
10.1	Macros and Environments . . . . .	75
10.1.1	The <code>smodule</code> environment . . . . .	77
<b>11</b>	<b>sTeX-Module Inheritance</b>	<b>79</b>
11.1	Macros and Environments . . . . .	79
11.1.1	SMS Mode . . . . .	79
11.1.2	Imports and Inheritance . . . . .	80
<b>12</b>	<b>sTeX-Symbols</b>	<b>82</b>
12.1	Macros and Environments . . . . .	82
<b>13</b>	<b>sTeX-Terms</b>	<b>84</b>
13.1	Macros and Environments . . . . .	84
<b>14</b>	<b>sTeX-Structural Features</b>	<b>86</b>
14.1	Macros and Environments . . . . .	86
14.1.1	Structures . . . . .	86
<b>15</b>	<b>sTeX-Statements</b>	<b>87</b>
15.1	Macros and Environments . . . . .	87
<b>16</b>	<b>sTeX-Proofs: Structural Markup for Proofs</b>	<b>88</b>
<b>17</b>	<b>sTeX-Metatheory</b>	<b>89</b>
17.1	Symbols . . . . .	89
<b>III</b>	<b>Extensions</b>	<b>90</b>
<b>18</b>	<b>Tikzinput: Treating TIKZ code as images</b>	<b>91</b>
18.1	Macros and Environments . . . . .	91
<b>19</b>	<b>document-structure: Semantic Markup for Open Mathematical Documents in L<sup>A</sup>T<sub>E</sub>X</b>	<b>92</b>
<b>20</b>	<b>NotesSlides – Slides and Course Notes</b>	<b>93</b>
<b>21</b>	<b>problem.sty: An Infrastructure for formatting Problems</b>	<b>94</b>
<b>22</b>	<b>hwexam.sty/cls: An Infrastructure for formatting Assignments and Exams</b>	<b>95</b>
<b>IV</b>	<b>Implementation</b>	<b>96</b>

<b>23</b>	<b>STeX-Basics Implementation</b>	<b>97</b>
23.1	The STeXDocument Class . . . . .	97
23.2	Preliminaries . . . . .	98
23.3	Messages and logging . . . . .	98
23.4	HTML Annotations . . . . .	99
23.5	Babel Languages . . . . .	101
23.6	Persistence . . . . .	102
23.7	Auxiliary Methods . . . . .	103
<b>24</b>	<b>STeX-MathHub Implementation</b>	<b>106</b>
24.1	Generic Path Handling . . . . .	106
24.2	PWD and kpsewhich . . . . .	108
24.3	File Hooks and Tracking . . . . .	109
24.4	MathHub Repositories . . . . .	110
24.5	Using Content in Archives . . . . .	115
<b>25</b>	<b>STeX-References Implementation</b>	<b>119</b>
25.1	Document URIs and URLs . . . . .	119
25.2	Setting Reference Targets . . . . .	121
25.3	Using References . . . . .	123
<b>26</b>	<b>STeX-Modules Implementation</b>	<b>126</b>
26.1	The smodule environment . . . . .	130
26.2	Invoking modules . . . . .	136
<b>27</b>	<b>STeX-Module Inheritance Implementation</b>	<b>138</b>
27.1	SMS Mode . . . . .	138
27.2	Inheritance . . . . .	142
<b>28</b>	<b>STeX-Symbols Implementation</b>	<b>148</b>
28.1	Symbol Declarations . . . . .	148
28.2	Notations . . . . .	156
28.3	Variables . . . . .	164
<b>29</b>	<b>STeX-Terms Implementation</b>	<b>172</b>
29.1	Symbol Invocations . . . . .	172
29.2	Terms . . . . .	179
29.3	Notation Components . . . . .	183
29.4	Variables . . . . .	185
29.5	Sequences . . . . .	188
<b>30</b>	<b>STeX-Structural Features Implementation</b>	<b>189</b>
30.1	Imports with modification . . . . .	190
30.2	The feature environment . . . . .	198
30.3	Structure . . . . .	198
<b>31</b>	<b>STeX-Statements Implementation</b>	<b>209</b>
31.1	Definitions . . . . .	209
31.2	Assertions . . . . .	215
31.3	Examples . . . . .	218
31.4	Logical Paragraphs . . . . .	221

<b>32 The Implementation</b>	<b>226</b>
32.1 Proofs . . . . .	226
<b>33 <math>\text{\TeX}</math>-Others Implementation</b>	<b>235</b>
<b>34 <math>\text{\TeX}</math>-Metatheory Implementation</b>	<b>237</b>
<b>35 Tikzinput Implementation</b>	<b>240</b>
<b>36 document-structure.sty Implementation</b>	<b>243</b>
36.1 Package Options . . . . .	243
36.2 Document Structure . . . . .	244
36.3 Front and Backmatter . . . . .	248
36.4 Global Variables . . . . .	250
<b>37 NotesSlides – Implementation</b>	<b>251</b>
37.1 Class and Package Options . . . . .	251
37.2 Notes and Slides . . . . .	253
37.3 Header and Footer Lines . . . . .	257
37.4 Frame Images . . . . .	259
37.5 Sectioning . . . . .	260
37.6 Excursions . . . . .	263
<b>38 The Implementation</b>	<b>265</b>
38.1 Package Options . . . . .	265
38.2 Problems and Solutions . . . . .	266
38.3 Markup for Added Value Services . . . . .	273
38.4 Multiple Choice Blocks . . . . .	273
38.5 Filling in Concrete Solutions . . . . .	274
38.6 Including Problems . . . . .	274
38.7 Reporting Metadata . . . . .	276
<b>39 Implementation: The hwexam Package</b>	<b>278</b>
39.1 Package Options . . . . .	278
39.2 Assignments . . . . .	279
39.3 Including Assignments . . . . .	282
39.4 Typesetting Exams . . . . .	283
39.5 Leftovers . . . . .	285
<b>40 References</b>	<b>286</b>

# Part I

## Manual



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



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

# Chapter 1

## What is sTeX?

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

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

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

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

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

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



# Chapter 2

## Quickstart

### 2.1 Setup

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

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

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

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

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

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

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

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

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

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

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

### 2.1.3 $\text{\texttt{sTeX}}$ Archives (Manual Setup)

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

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

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

Note that  $\text{\texttt{sTeX}}$  archives often depend on other archives, thus you should be prepared to clone these as well – e.g. if `pdflatex` reports missing files. To make sure that  $\text{\texttt{sTeX}}$  too knows where to find its archives, we need to set a global system variable `MATHHUB`, that points to your local MathHub-directory (see [section 3.2](#)).

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

### 2.1.4 The $\text{\texttt{sTeX}}$ IDE

We are currently working on an  $\text{\texttt{sTeX}}$  IDE as an  $\text{\texttt{sTeX}}$  plugin for VScode; see [\[S1a\]](#). It will feature a setup procedure that automates the setup described above (and below). For additional functionality see the (now obsolete) plugin for  $\text{\texttt{sTeX}}$  1 [\[SLS; Stb\]](#).

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

Foregoing on the  $\text{\texttt{sTeX}}$  IDE, we will need several additional (on top of the minimal setup above) pieces of software; namely:

- **The Mmt System** available [here](#). 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  $\text{\texttt{sTeX}}$ /MMT content archives.

- **$\text{\texttt{sTeX}}$  Archives** If we only care about  $\text{\texttt{LATEX}}$  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)  $\text{\texttt{sTeX}}$  archives are cloned as well.

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

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

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

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

```

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

```

Compiling this document with `pdflatex` should yield the output

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

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

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

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

### Remark 2.2.1:

Note that all of the highlighting, tooltips, coloring and the environment headers come from `stexthm` – by default, the amount of additional packages loaded is kept to a minimum and all the presentations can be customized, see [section 5.3](#).

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

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

First, we open a new *module* called `GeometricSeries`. The main purpose of the `smodule` environment is to group the contents and associate it with a *globally unique* identifier (URI), which is computed from the name `GeometricSeries` and the document context.

(Depending on your pdf viewer), the URI should pop up in a tooltip if you hover over the word `geometric series`.

---

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

---

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

The `\importmodule`-statements make all  $\text{\TeX}$  symbols and associated semantic macros (e.g. `\infinitiesum`, `\realdive`, `\realpower`) in the imported module available to the current module `GeometricSeries`. The module `GeometricSeries` “exports” all of these symbols to all modules imports it via an `\importmodule{GeometricSeries}` instruction. Additionally it exports the local symbol `\geometricSeries`.

---

```
\usemodule
```

---

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

---

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

---

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

---

```
\comp
```

---

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

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

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

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

---

`\symname`

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

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

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

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

---

`\symref`

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

---

`\define`  
`\definiendum`

The `\define{geometricSeries} ...`

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

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

The next snippet – set in a math environment – uses several semantic macros imported from (or recursively via) `series` and `realarithmetics`, such as `\defeq`, `\infinitesum`, etc. In math mode, using a semantic macro inserts its (default) definition. A semantic macro can have several notations – in that case, we can explicitly choose a specific notation by providing its identifier as an optional argument; e.g. `\realdivide[frac]{a}{b}` will use the explicit notation named `frac` of the semantic macro `\realdivide`, which yields  $\frac{a}{b}$  instead of  $a/b$ .

<hr/> <code>\svar</code> <hr/>	The <code>\svar{n}</code> command marks up the <code>n</code> as a variable with name <code>n</code> and notation <code>n</code> .
<hr/> <code>\definiens</code> <hr/>	The <code>sdefinition</code> -environment additionally provides the <code>\definiens</code> -command, which allows for explicitly marking up its argument as the <i>definiens</i> of the symbol currently being defined.

### 2.2.1 OMDoc/xhtml Conversion

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

#### TODO VSCode Plugin

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

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

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

```
<OMBIND>
  <OMID name="...?series?infinitesum"/>
  <OMV name="n"/>
```

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

```

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

```

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

#### Remark 2.2.2:

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

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

## 2.2.2 Mmt/OMDoc Conversion

Another way to convert our document to *actual* MMT/OMDOC is to put it in an `TEX` archive (see [section 3.2](#)) and have MMT take care of everything.

Assuming the above file is `source/demo.tex` in an `TEX` archive `MyTest`, you can run MMT and do `build MyTest stex-omdoc demo.tex` to convert the document to both `xhtml` (which you will find in `xhtml/demo.xhtml` in the archive) and formal MMT/OMDOC, which you can subsequently view in the MMT browser (see <https://uniformal.github.io/doc/applications/server.html#the-mmt-web-site> for details).

## Chapter 3

# Creating sTeX Content

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

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

**lang** (*(⟨language⟩\*)*) Languages to load with the babel package.

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

**writesms** (*(⟨boolean⟩)*) with this package option, sTeX will write the contents of all external modules imported via `\importmodule` or `\usemodule` into a file `\jobname.sms` (analogously to the table of contents `.toc`-file).

**usesms** (*(⟨boolean⟩)*) subsequently tells sTeX to read the generated sms-file at the beginning of the document. This allows for e.g. collaborating on documents without all authors having to have all used archives and modules available – one author can load the modules with **writesms**, and the rest can use the modules with **usesms**. Furthermore, the sms file can be submitted alongside a `tex`-file, effectively making it “standalone”.

**image** (*(⟨boolean⟩)*) passed on to tikzinput.

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

### 3.1 How Knowledge is Organized in sTeX

sTeX content is organized on multiple levels:

1. sTeX **archives** (see [section 3.2](#)) contain individual `.tex`-files.
2. These may contain sTeX **modules**, introduced via `\begin{smodule}{ModuleName}`.



3. Modules contain  $\text{\S}\text{\TeX}$  **symbol declarations**, introduced via  $\text{\textbackslash symdecl}\{\text{symbolname}\}$ ,  $\text{\textbackslash symdef}\{\text{symbolname}\}$  and some other constructions. Most symbols have a *notation* that can be used via a *semantic macro*  $\text{\textbackslash symbolname}$  generated by symbol declarations.
4.  $\text{\S}\text{\TeX}$  **expressions** finally are built up from usages of semantic macros.

- $\text{\S}\text{\TeX}$  archives are simultaneously MMT archives, and the same directory structure is consequently used.
  - $\text{\S}\text{\TeX}$  modules correspond to OMDOC/MMT *theories*.  $\text{\textbackslash importmodules}$  (and similar constructions) induce MMT **includes** and other *theory morphisms*, thus giving rise to a *theory graph* in the OMDOC sense [RK13].
  - Symbol declarations induce OMDOC/MMT *constants*, with optional (formal) *type* and *definiens* components.
  - Finally,  $\text{\S}\text{\TeX}$  expressions are converted to OMDOC/MMT terms, which use the abstract syntax (and XML encoding) of OPENMATH [Bus+04].

## 3.2 $\text{\S}\text{\TeX}$ Archives

### 3.2.1 The Local MathHub-Directory

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

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

1. If the  $\text{\S}\text{\TeX}$  package is loaded with the option `mathhub=/path/to/mathhub`, then  $\text{\S}\text{\TeX}$  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  $\text{\S}\text{\TeX}$ -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,  $\text{\S}\text{\TeX}$  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. Finally, if all else fails,  $\text{\S}\text{\TeX}$  will look for a file `~/.stex/mathhub.path`. If this file exists,  $\text{\S}\text{\TeX}$  will assume that it contains the path to the local MathHub-directory. This method is recommended on systems where it is difficult to set environment variables.

### 3.2.2 The Structure of $\text{\TeX}$ Archives

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

Each such archive needs two subdirectories:

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

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

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

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

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

### 3.2.3 MANIFEST.MF-Files

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

```
id: smglom/calculus
source-base: http://mathhub.info/smgglom/calculus
narration-base: http://mathhub.info/smgglom/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`.

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

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

<div style="border-bottom: 1px solid black; padding-bottom: 5px;"><math>\backslash\text{mhinput}</math></div>	$\backslash\text{mhinput}$ [Some/Archive]{some/file} directly inputs the file some/file in the source-folder of Some/Archive.
---	---

<div style="border-bottom: 1px solid black; padding-bottom: 5px;"><math>\backslash\text{inputref}</math></div>	$\backslash\text{inputref}$ [Some/Archive]{some/file} behaves like $\backslash\text{mhinput}$ , but wraps the input in a <code>\begingroup ... \endgroup</code> . When converting to <code>xhtml</code> , the file is not input at all, and instead an html-annotation is inserted that references the file, e.g. for lazy loading.
--	---

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

<div style="border-bottom: 1px solid black; padding-bottom: 5px;"><math>\backslash\text{ifinput}</math></div>	Both $\backslash\text{mhinput}$ and $\backslash\text{inputref}$ set $\backslash\text{ifinput}$ to “true” during input. This allows for selectively including e.g. bibliographies only if the current file is not being currently included in a larger document.
---	---

<div style="border-bottom: 1px solid black; padding-bottom: 5px;"><math>\backslash\text{addmhbibresource}</math></div>	$\backslash\text{addmhbibresource}$ [Some/Archive]{some/file} searches for a file like $\backslash\text{mhinput}$ does, but calls $\backslash\text{addbibresource}$ to the result and looks for the file in the archive root directory directly, rather than the <code>source</code> directory. Typical invocations are
--	---

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

---

`\libinput` `\libinput{some/file}` searches for a file `some/file` in

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

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

`\libinput` will throw an error if *no* candidate for `some/file` is found.

---

`\libusepackage` `\libusepackage[package-options]{some/file}` searches for a file `some/file.sty` in the same way that `\libinput` does, but will call `\usepackage[package-options]{path/to/some/file}` instead of `\input`.

`\libusepackage` throws an error if not *exactly one* candidate for `some/file` is found.

**Remark 3.2.1:**

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

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

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

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

## 3.3 Module, Symbol and Notation Declarations

### 3.3.1 The `smodule`-Environment

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

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

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

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

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

`type` ( $\langle string \rangle$ \*) for use in customizations.  
`deprecate` ( $\langle module \rangle$ ) if set, will throw a warning when loaded, urging to use  $\langle module \rangle$  instead.  
`id` ( $\langle string \rangle$ ) for cross-referencing.  
`ns` ( $\langle URI \rangle$ ) the namespace to use. *Should not be used, unless you know precisely what you're doing.* If not explicitly set, is computed using `\stex_modules_current_namespace:`.  
`lang` ( $\langle language \rangle$ ) if not set, computed from the current file name (e.g. `foo.en.tex`).  
`sig` ( $\langle language \rangle$ ) if the current file is a translation of a file with the same base name but a different language suffix, setting `sig=<lang>` will preload the module from that language file. This helps ensuring that the (formal) content of both modules is (almost) identical across languages and avoids duplication.  
`creators` ( $\langle string \rangle$ \*) names of the creators.  
`contributors` ( $\langle string \rangle$ \*) names of contributors.  
`srccite` ( $\langle string \rangle$ ) a source citation for the content of this module.

$\hookrightarrow$  An  $\text{\texttt{sTeX}}$  module corresponds to an MMT/OMDOC *theory*. As such it  
 $\rightarrow$  gets assigned a module URI (*universal resource identifier*) of the form  
 $\rightsquigarrow$  `<namespace>?<module-name>`.

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

### Example 1

Input:

```

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

```

Output:

Hello World

## \stexpatchmodule

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

For example:

### Example 2

Input:

```

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

```

Output:

```

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

```

### 3.3.2 Declaring New Symbols and Notations

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

`\symdecl`

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

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

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

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

#### Example 3

Input:

```

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

```

Output:

```

Given a foo, we can...

```

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

#### Example 4

Input:

```

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

```

Output:

```

this is a symbol taking two arguments.

```

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

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

---

`\notation`

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

#### Example 5

Input:

```

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

```

Output:

```

First: a; Second: b

```

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

---

`\comp`

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

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

#### Example 6

Input:

```

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

```

Output:

First: *a*; Second: *b*



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



Note that it is required that

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

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

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

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

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

---

`\symdef`

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



### Example 7

Input:

```

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

```

Output:

```

1.: a; 2.: b

```

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

But one man’s meat is another man’s poison: it is very subjective what the “default notation” of an operator should be. Different communities have different practices. For instance, the complex unit is written as  $i$  in Mathematics and as  $j$  in electrical engineering. So to allow modular specification and facilitate re-use of document fragments  $\text{\S}$ TEX allows to re-set notation defaults.

#### \setnotation

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

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

#### \textsymdecl

In the less mathematical settings where we want a symbol and semantic macro for some concept with a notation *beyond* its mere name, but which should also be available in  $\text{\TeX}$ ’s text mode, the command `\textsymdecl` is useful. For example, we can declare a symbol `openmath` with the notation `\textsc{OpenMath}` using `\textsymdecl{openmath}[name=OpenMath]{\textsc{OpenMath}}`. The `\openmath` yields `OPENMATH` both in text and math mode.

### Operator Notations

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

We can do so by supplying the `\notation` (or `\symdef`) with an *operator notation*, indicated with the optional argument `op=`. We can then invoke the operator notation

using `\symbolname!`[notation-identifier]. Since operator notations never take arguments, we do not need to use `\comp` in it, the whole notation is wrapped in a `\comp` automatically:

#### Example 8

Input:

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

Output:

`newbinarysymbol` is also occasionally written `a: · ; b: ·`

$\hookrightarrow$  `\symbolname!` is translated to OMDOC/MMT as `<OMS name="...?symbolname"/>`  
 $\hookrightarrow$  directly.  
 $\rightsquigarrow$  `T`  $\rightsquigarrow$

### 3.3.3 Argument Modes

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

#### Mode-b Arguments

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

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

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

#### Example 9

Input:

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

Output:

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

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

## Mode-a Arguments

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

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

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

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

### Example 10

Input:

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

Output:

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

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

### Example 11

Input:

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

Output:

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

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

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

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

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

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

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

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

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

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

## Mode-B Arguments

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

### Example 12

Input:

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

Output:

$\forall x, y, z. P$

### 3.3.4 Type and Definiens Components

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

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

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

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

#### Example 13

Input:

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

Output:

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

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

#### Example 14

Input:

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

Output:

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

### 3.3.5 Precedences and Automated Bracketing

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

#### Example 15

Input:

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

Output:

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

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

#### Example 16

Input:

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

Output:

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

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

#### Example 17

Input:

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

Output:

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

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

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

We will investigate the precise meaning of `<opprec>` and the `<argprec>`s shortly – in the vast majority of cases, it is perfectly sufficient to think of `prec=` taking a single number and having that be *the* precedence of the notation, where lower precedences (somewhat

counterintuitively) bind stronger than higher precedences. So fixing our notations for `\addition` and `\multiplication`, we get:

### Example 18

Input:

```

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

```

Output:

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

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

---

`\infprec`  
`\neginfprec`

---

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



More precisely, each notation takes

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

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

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

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

In the example above:

1.  $\text{\texttt{S}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$  starts out with  $p_d = \text{\texttt{\textcode{\neginfprec}}}$ .
2.  $\text{\texttt{S}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$  encounters `\addition` with  $p_{op} = 100$ . Since  $100 \not> \text{\texttt{\textcode{\neginfprec}}}$ , it inserts no parentheses.
3. Next,  $\text{\texttt{S}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$  encounters the two arguments for `\addition`. Both have no specifically provided argument precedence, so  $\text{\texttt{S}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$  uses  $p_d = p_{op} = 100$  for both and recurses.



4. Next,  $\text{\S}\text{\TeX}$  encounters  $\text{\textcolor{teal}{multiplication}\{b,\dots\}}$ , whose notation has  $p_{op} = 50$ .
5. We compare to the current downward precedence  $p_d$  set by  $\text{\textcolor{teal}{addition}}$ , arriving at  $p_{op} = 50 \not> 100 = p_d$ , so  $\text{\S}\text{\TeX}$  again inserts no parentheses.
6. Since the notation of  $\text{\textcolor{teal}{multiplication}}$  has no explicitly set argument precedences,  $\text{\S}\text{\TeX}$  uses the operator precedence for all arguments of  $\text{\textcolor{teal}{multiplication}}$ , hence sets  $p_d = p_{op} = 50$  and recurses.
7. Next,  $\text{\S}\text{\TeX}$  encounters the inner  $\text{\textcolor{teal}{addition}\{c,\dots\}}$  whose notation has  $p_{op} = 100$ .
8. We compare to the current downward precedence  $p_d$  set by  $\text{\textcolor{teal}{multiplication}}$ , arriving at  $p_{op} = 100 > 50 = p_d$  – which finally prompts  $\text{\S}\text{\TeX}$  to insert parentheses, and we proceed as before.

### 3.3.6 Variables

All symbol and notation declarations require a module with which they are associated, hence the commands  $\text{\textcolor{blue}{symdecl}}$ ,  $\text{\textcolor{blue}{notation}}$ ,  $\text{\textcolor{blue}{symdef}}$  etc. are disabled outside of  $\text{\textcolor{black}{smodule}}$ -environments.

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

---

$\text{\textcolor{blue}{svar}}$

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

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

---

$\text{\textcolor{blue}{vardef}}$

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

#### Example 19

Input:

```

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

```



Output:

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

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

**TODO:** `bind=forall/exists`

### 3.3.7 Variable Sequences

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

`\varseq`

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

This is best shown by example:

#### Example 20

Input:

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

Output:

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

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

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

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

#### Example 21

Input:

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

Output:

$a_1 + \dots + a_n$

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

#### Example 22

Input:

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

Output:

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

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

#### Example 23

Input:

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

Output:

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

## 3.4 Module Inheritance and Structures

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

Before we get into theory morphisms in  $\text{\texttt{gT\TeX}}$  we will see a very simple application of modules: managing multilinguality modularly.

### 3.4.1 Multilinguality and Translations

If we load the  $\text{\texttt{gT\TeX}}$  document class or package with the option `lang=<lang>`,  $\text{\texttt{gT\TeX}}$  will load the appropriate `babel` language for you – e.g. `lang=de` will load the `babel` language

ngerman. Additionally, it makes  $\text{\TeX}$  aware of the current document being set in (in this example) *german*. This matters for reasons other than mere `babel`-purposes, though:

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

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

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

If a module `Foo` exists in e.g. *english* in a file `Foo.en.tex`, we can provide a file `Foo.de.tex` right next to it, and write `\begin{smodule}[sig=en]{Foo}`. The `sig`-key then signifies, that the “signature” of the module is contained in the *english* version of the module, which is immediately imported from there, just like `\importmodule` would.

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

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

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

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

### 3.4.2 Simple Inheritance and Namespaces

---

`\importmodule`  
`\usemodule`

---

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

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

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



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

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

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



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

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

- Similarly, in `\importmodule{some/path?Foo}` the path `some/path` refers to either the sub-directory and relative namespace path of the current directory and namespace outside of an archive, or relative to the current archive's top-level namespace and `source`-folder, respectively.

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

- Similarly, `\importmodule[Some/Archive]{some/path?Foo}` is resolved like the previous cases, but relative to the archive `Some/Archive` in the mathhub-directory.
- Finally, `\importmodule{full://uri?Foo}` naturally refers to the module `Foo` in the namespace `full://uri`. Since the file this module is declared



in can not be determined directly from the URI, the module must be in memory already, e.g. by being referenced earlier in the same document. Since this is less compatible with a modular development, using full URIs directly is strongly discouraged, unless the module is declared in the current file directly.

---

`\STEXexport`

---

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



For persistency reasons, everything in an `\STEXexport` is digested by T<sub>E</sub>X in the L<sup>A</sup>T<sub>E</sub>X3-category code scheme. This means that the characters `_` and `:` are considered *letters* and valid parts of control sequence names, and space characters are ignored entirely. For spaces, use the character `~` instead, and keep in mind, that if you want to use subscripts, you should use `\c_math_subscript_token` instead of `_`!

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

A safer alternative is to use macro definition principles, that are safe to use even if the macro being defined already exists, and ideally are local to the current T<sub>E</sub>X group, such as `\def` or `\let`.

### 3.4.3 The `mathstructure` Environment

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

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

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

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

How this works is again best demonstrated by example:

### Example 24

Input:

```

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

Output:

A **monoid** is...

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

### Example 25

Input:

```

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

Output:

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

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

### Example 26

Input:

```

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

```

Output:

```

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

```

### \instantiate

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

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

`\instantiate` and `mathstructure` make use of the *Theories-as-Types* paradigm (see [MRK18]):

- `mathstructure{<name>}` simply creates a nested theory with name `<name>-structure`. The *constant* `<name>` is defined as `Mod(<name>-structure)`
- `→M` – a *dependent record type with manifest fields*, the fields of which are generated from (and correspond to) the constants in `<name>-structure`.
- `→T` `\instantiate` generates a constant whose definiens is a record term of type `Mod(<name>-structure)`, with the fields assigned based on the respective key-value-list.

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

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

### \varinstantiate

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

This allows us to do things like:

#### Example 27

Input:

```

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

```

Output:

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

.

and

### Example 28

Input:

```

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

```

Output:

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

.

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

### 3.4.4 The copymodule Environment

TODO: explain

Given modules:

### Example 29

Input:

```

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

```

Output:



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

### Example 30

Input:

```

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

```

Output:

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

TODO: explain donotclone

### 3.4.5 The interpretmodule Environment

TODO: explain

### Example 31

Input:

```

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

```

Output:

---

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

The `stex-metattheory` package contains  $\text{sTeX}$  symbols so ubiquitous, that it is virtually impossible to describe any flexiformal content without them, or that are required to annotate even the most primitive symbols with meaningful (foundation-independent) “type”-annotations, or required for basic structuring principles (theorems, definitions). As such, it serves as the default meta theory for any  $\text{sTeX}$  module.

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

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

We make this theory part of the  $\text{sTeX}$  collection due to the obiquity of the symbols involved. Note however, that the metatheory is for all practical purposes a “normal”  $\text{sTeX}$  module, and the symbols contained “normal”  $\text{sTeX}$  symbols.

## Chapter 4

# Using $\text{\TeX}$ Symbols

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

### 4.1 `\symref` and its variants

---

`\symref`  
`\symname`

---

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

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

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

#### Example 32

Input:

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

Output:

A natural number is...

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

`\Symname`

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

### Example 33

Input:

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

Output:

Natural numbers are...



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

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

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

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

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

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

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

### Example 34

Input:

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

Output:

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

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

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



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

## \arg

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

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

### Example 35

Input:

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

Output:

```
Addition is...
```

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

`\arg` also allows us to switch the order of arguments around and “hide” arguments: For example, `\arg[3]{<code>}` signifies that `<code>` represents the *third* argument to the current operator, and `\arg*[i]{<code>}` signifies that `<code>` represents the *i*th argument, but it should not produce any output (it is exported in the xhtml however, so that MMT and other systems can pick up on it).<sup>1</sup>

### Example 36

Input:

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

Output:

<sup>1</sup>EdNOTE: MK: I do not understand why we have to/want to give the second `arg*`; I think this must be elaborated on.

adding  $k$  yields...

Note that since the second `\arg` has no explicit argument number, it automatically represents the first not-yet-given argument – i.e. in this case the first one.<sup>2</sup>

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

### Example 37

Input:

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

Output:

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

## 4.3 Referencing Symbols and Statements

TODO: references documentation

<sup>2</sup>EdNOTE: MK: I do not understand this at all.

## Chapter 5

# sTeX Statements

### 5.1 Definitions, Theorems, Examples, Paragraphs

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

The corresponding environments for that are:

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

The *presentation* of these environments can be customized to use e.g. predefined theorem-environments, see [section 5.3](#) for details.

All of these environments take optional arguments in the form of `key=value`-pairs. Common to all of them are the keys `id=` (for cross-referencing, see [section 4.3](#)), `type=` for customization (see [section 5.3](#)) and additional information (e.g. definition principles, “difficulty” etc), as well as `title=` (for giving the paragraph a title), and finally `for=`.

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

#### Example 38

Input:

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

Output:

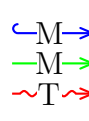

**Example 5.1.1** (An Example).  $2+3$  is 5,  $2\cdot 3$  is 6.

---

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

---

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

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

---

`\definiens`

---

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

All four statement environments – i.e. `sdefinition`, `sassertion`, `sexample`, and `sparagraph` – also take an optional parameter `name=` – if this one is given a value, the environment will generate a *symbol* by that name (but with no semantic macro). Not only does this allow for `\symref` et al, it allows us to resume our earlier example for monoids much more nicely:<sup>3</sup>

### Example 39

Input:

---

<sup>3</sup>EdNOTE: MK: we should reference the example explicitly here.



```

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

```

Output:

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

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

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

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

The main difference to before<sup>4</sup> is that the two **sassertions** now have **name=** attributes. Thus the **mathstructure monoid** now contains two additional symbols, namely the axioms for associativity and that  $e$  is a unit. Note that both symbols do not represent the mere *propositions* that e.g.  $\circ$  is associative, but *the assertion that it is actually true* that  $\circ$  is associative.

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

```

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

```

<sup>4</sup>EdNOTE: MK: reference

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

## 5.2 Proofs

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

Its central component is the `sproof`-environment, whose body consists of:

- *subproofs* via the `subproof`-environment,
- *proof steps* via the `\spfstep`, `\eqstep` `\assumption`, and `\conclude` macros, and
- *comments*, via normal text without special markup.

`sproof`, `subproof` and the various proof step macros take the following optional arguments:

`id` ( $\langle string \rangle$ ) for referencing,

`method` ( $\langle string \rangle$ ) the proof method (e.g. contradiction, induction,...)

`term` ( $\langle token list \rangle$ ) the (ideally semantically-marked up) proposition that is derived/proven by this proof/subproof/proof step.

Additionally, they take one mandatory argument for the document text to be annotated, or (in the case of the environments) as an introductory description of the proof itself. Since the latter often contains the `term` to be derived as text, alternatively to providing it as an optional argument, the mandatory argument can use the `\yield`-macro to mark it up in the text.

The `sproof` and `subproof` environments additionally take two optional arguments:

`for` the symbol identifier/name corresponding to the `sassertion` to be proven. This too subsumes `\yield` and the `term`-argument.

`hide` In the pdf, this only shows the mandatory argument text and hides the body of the environment. In the HTML (as served by MMT), the bodies of all `proof` and `subproof` environments are *collapsible*, and `hide` collapses the body by default.

```

1 \begin{sassertion}[type=theorem,name=sqrt2irr]
2   \conclusion{\irrational{\arg{\realroot{2}}$ is \comp{irrational}}}.
3 \end{sassertion}
4
5 \begin{sproof}[for=sqrt2irr,method=contradiction]{By contradiction}
6   \assumption{Assume \yield{\rational{\arg{\realroot{2}}$ is
7     \comp{rational}}}}
8   \begin{subproof}[method=straightforward]{Then
9     \yield{\$eq{\ratfrac{\intpow{\vara}{2}}{\intpow{\varb}{2}}{2}$
10       for some $\inset{\vara,\varb}\PosInt$ with
11       \coprime{\arg{\vara},\arg{\varb}$ \comp{coprime}}}}
```

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

```

12 \assumption{By assumption, \yield{there are
13 $\inset{\vara,\varb}\PosInt$ with
14 $\realroot{2}=\ratfrac{\vara}{\varb}$}}
15 \spfstep{wlog, we can assume \coprime{$\arg{\vara},\arg{\varb}$}
16 to be \comp{coprime}}
17 % a comment:
18 If not, reduce the fraction until numerator and denominator
19 are coprime, and let the resulting components be
20 $\vara$ and $\varb$
21 \spfstep{Then \yield{$\eq{\intpow{\ratfrac{\vara}{\varb}}{2}{2}$}}
22 \eqstep{\ratfrac{\intpow{\vara}{2}}{\intpow{\varb}{2}}}}
23 \end{subproof}
24 \begin{subproof}[term=\divides{2}{\vara},method=straightforward]{
25 Then $\vara$ is even}
26 \spfstep{Multiplying the equation by $\intpow{\varb}{2}$ yields
27 $\yield{\eq{\intpow{\vara}{2}}{\inttimes{2}{\intpow{\varb}{2}}}}$}
28 \spfstep[term=\divides{2}{\intpow{\vara}{2}}]{Hence
29 $\intpow{\vara}{2}$ is even}
30 \conclude[term=\divides{2}{\vara}]{Hence $\vara$ is even as well}
31 % another comment:
32 Hint: Think about the prime factorizations of $\vara$ and
33 $\intpow{\vara}{2}$
34 \end{subproof}
35 \begin{subproof}[term=\divides{2}{\varb},method=straightforward,]{
36 Then $\varb$ is also even}
37 \spfstep{Since $\vara$ is even, we have \yield{some $\varc$ $
38 such that $\eq{\inttimes{2}{\varc}}{\vara}$}}
39 \spfstep{Plugging into the above, we get
40 \yield{$\eq{\intpow{\inttimes{2}{\vara}}{2}
41 {\inttimes{2}{\intpow{\varb}{2}}}$}}
42 \eqstep{\inttimes{4}{\intpow{\vara}{2}}}
43 \spfstep{Dividing both sides by $2$ yields
44 \yield{$\eq{\intpow{\varb}{2}}{\inttimes{2}{\intpow{\vara}{2}}}$}}
45 \spfstep[term=\divides{2}{\intpow{\varb}{2}}]{Hence
46 $\intpow{\varb}{2}$ is even}
47 \conclude[term=\divides{2}{\varb}]{Hence $\varb$ is even}
48 % one more comment:
49 By the same argument as above
50 \end{subproof}
51 \conclude[term=\contradiction]{Contradiction to $\vara,\varb$ being
52 \symname{coprime}.}
53 \end{sproof}

```

which will produce:

**Theorem 5.2.1.**  $\sqrt{2}$  is *irrational*.

**Proof:** By contradiction

1. Assume  $\sqrt{2}$  is *rational*
2. Then  $(\frac{a}{b})^2=2$  for some  $a,b \in \mathbb{Z}^+$  with  $a,b$  *coprime*
  - 2.1. By assumption, there are  $a,b \in \mathbb{Z}^+$  with  $\sqrt{2} = \frac{a}{b}$
  - 2.2. wlog, we can assume  $a,b$  to be *coprime*

*If not, reduce the fraction until numerator and denominator are coprime, and let the re-*

sulting components be  $a$  and  $b$

2.3. Then  $(\frac{a}{b})^2=2$

$$= \frac{a^2}{b^2}$$

3. Then  $a$  is even

3.1. Multiplying the equation by  $b^2$  yields  $a^2=2b^2$

3.2. Hence  $a^2$  is even

$\Rightarrow$  Hence  $a$  is even as well

*Hint: Think about the prime factorizations of  $a$  and  $a^2$*

4. Then  $b$  is also even

4.1. Since  $a$  is even, we have some  $c$  such that  $2c=a$

4.2. Plugging into the above, we get  $(2a)^2=2b^2$

$$= 4a^2$$

4.3. Dividing both sides by 2 yields  $b^2=2a^2$

4.4. Hence  $b^2$  is even

$\Rightarrow$  Hence  $b$  is even

*By the same argument as above*

$\Rightarrow$  Contradiction to  $a, b$  being coprime.

□

If we mark all subproofs with `hide`, we will obtain the following instead:

**Theorem 5.2.2.**  $\sqrt{2}$  is irrational.

**Proof:** By contradiction

1. Assume  $\sqrt{2}$  is rational

2. Then  $(\frac{a}{b})^2=2$  for some  $a, b \in \mathbb{Z}^+$  with  $a, b$  coprime

3. Then  $a$  is even

4. Then  $b$  is also even

$\Rightarrow$  Contradiction to  $a, b$  being coprime.

□

However, the hidden subproofs will still be shown in the HTML, only in an expandable section which is collapsed by default.

The above style of writing proofs is usually called *structured proofs*. They have a huge advantage over the traditional purely prosaic style, in that (as the name suggests) the actual *structure* of the proof is made explicit, which almost always makes it considerably more comprehensible. We, among many others, encourage the general use of structured proofs.

Alas, most proofs are not written in this style, and we would do users a disservice by insisting on this style. For that reason, the `spfblock` environment turns all subproofs and proof step macros into presentationally neutral *inline* annotations, as in the induction step of the following example:

```
1 \begin{sproof}[id=simple-proof,method=induction]
2   {We prove that  $\sum_{i=1}^{n-1} 2i-1 = n^2$  by induction over  $n$ }
```

```

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

```

This yields the following result:

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

*For the induction we have to consider three cases:*

1.  $n = 1$

then we compute  $1 = 1^2$

2.  $n = 2$

*This case is not really necessary, but we do it for the fun of it (and to get more intuition).*

We compute  $1 + 3 = 2^2 = 4$ .

3.  $n > 1$

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

We have to show that we can derive the assertion for  $n = k + 1$  from this assumption,

i.e.  $\sum_{i=1}^{k+1} (2i - 1) = (k + 1)^2$ .  
 We obtain  $\sum_{i=1}^{k+1} 2i - 1 = \sum_{i=1}^k 2i - 1 + 2(k + 1) - 1$  by [splitting the sum](#). Thus  
 we have  $\sum_{i=1}^{k+1} (2i - 1) = k^2 + 2k + 1$  by [induction hypothesis](#). We can [simplify](#) the  
 right-hand side to  $k + 1^2$ , which proves the assertion.  
 $\Rightarrow$  We have considered all the cases, so we have proven the assertion. □

**proof** 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 **spfstep**, **spfcomment**, and **spfcases** environments that are used to markup the proof steps.

---

**\spfname** The **\spfname** 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 same optional argument as **proof** and another one: a natural language text that sketches the proof.

---

**\spfstep** Regular proof steps are marked up with the **\spfstep** macro, 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.

---

**\yield** See above

---

**\spfjust** This evidence is marked up with the **\spfjust** macro in the **stex-proofs** package. This environment totally invisible to the formatted result; it wraps the text in the proof step that corresponds to the evidence (ideally, a semantically marked-up term).

---

**\assumption** The **\assumption** macro allows to mark up a (justified) assumption.

---

**\justarg**

**subproof** The **subproof** environment is used to mark up a subproof. This environment takes an optional **KeyVal** argument for semantic annotations and a second argument that allows to specify an introductory comment (just like in the **proof** environment). The **method** key can be used to give the name of the proof method executed to make this subproof.

---

---

**`\sproofend`**

---

---

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

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

---

---

**`\sProofEndSymbol`**

---

---

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

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

## 5.3 Highlighting and Presentation Customizations

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

The `stexthm` package defines some default customizations that can be used, but of course many existing L<sup>A</sup>T<sub>E</sub>X templates come with their own `definition`, `theorem` and similar environments that authors are supposed (or even required) to use. Their concrete syntax however is usually not compatible with all the additional arguments that S<sub>T</sub>E<sub>X</sub> allows for semantic information.

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

---

---

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

---

---

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

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

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

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

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

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

```

1 \stexpatchassertion[theorem]
2   {\ifx\sassertiontitle\@empty
3     \begin{theorem}
4   \else
5     \begin{theorem}[\sassertiontitle]
6   \fi}
7 {\end{theorem}}

```

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

```

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

```

---

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

---

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

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

---

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

---

For each of the four macros, there exists an additional macro that takes the full URI of the relevant symbol currently being highlighted as a second argument. That allows us to e.g. use pdf tooltips and links. For example, this document uses<sup>5</sup>

```

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

```

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



## Chapter 6

# Additional Packages

### 6.1 Tikzinput: Treating TIKZ code as images

---

**image**

---

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

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

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

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

---

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

---

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

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

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

---

`\mhtikzinput`  
`\cmhtikzinput`

---

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

---

`\libusetikzlibrary`

---

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

## 6.2 Modular Document Structuring

### 6.2.1 Introduction

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

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

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

### 6.2.2 Package Options

The `document-structure` package accepts the following options:

<code>class=&lt;name&gt;</code>	load <code>&lt;name&gt;.cls</code> instead of <code>article.cls</code>
<code>topsect=&lt;sect&gt;</code>	The top-level sectioning level; the default for <code>&lt;sect&gt;</code> is <code>section</code>

### 6.2.3 Document Fragments

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

```

1 \begin{smodule}{foo}
2   \symdef{bar}{Ba_r}
3   ...
4   \begin{sfragment}[id=sec.bardriv,loadmodules]
5     {Introducing $\protect\bar$ Derivations}

```

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

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

**blindfragment** Therefore the document-structure package provides a variant `blindfragment` that does not produce markup, but increments the sectioning level and logically groups document parts that belong together, but where traditional document markup relies on convention rather than explicit markup. The `blindfragment` environment is useful e.g. for creating frontmatter at the correct level. The example below shows a typical setup for the outer document structure of a book with parts and chapters.

```

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

```

Here we use two levels of `blindfragment`:

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

---

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

---

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

---

`\currentsectionlevel`  
`\CurrentSectionLevel`

---

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

## 6.2.4 Ending Documents Prematurely

---

`\prematurestop`  
`\afterprematurestop`

---

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

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

Text fragments and modules can be made more re-usable by the use of global variables. For instance, the admin section of a course can be made course-independent (and therefore re-usable) by using variables (actually token registers) `courseAcronym` and `courseTitle` instead of the text itself. The variables can then be set in the `TeX` preamble of the course notes file.

## 6.2.5 Global Document Variables

To make document fragments more reusable, we sometimes want to make the content depend on the context. We use **document variables** for that.

---

`\setSGvar`  
`\useSGvar`

---

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

---

`\ifSGvar`

---

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

## 6.3 Slides and Course Notes

### 6.3.1 Introduction

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

The `notesslides` class takes the notion of a slide frame from Till Tantau’s excellent `beamer` class and adapts its notion of frames for use in the `TeX` 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.

### 6.3.2 Package Options

The `notesslides` class takes a variety of class options:

<u>slides</u> <u>notes</u>	The options <code>slides</code> and <code>notes</code> switch between slides mode and notes mode (see <a href="#">subsection 6.3.3</a> ).
<u>sectocframes</u>	If the option <code>sectocframes</code> is given, then for the <code>sfragments</code> , special frames with the <code>sfragment</code> title (and number) are generated.
<u>frameimages</u> <u>fiboxed</u>	If the option <code>frameimages</code> is set, then slide mode also shows the <code>\frameimage</code> -generated frames (see ). If also the <code>fiboxed</code> option is given, the slides are surrounded by a box.

### 6.3.3 Notes and Slides

- `frame` Slides are represented with the `frame` environment just like in the `beamer` class, see [\[Tanb\]](#) for details.
- `note` The `notesslides` class adds the `note` environment for encapsulating the course note fragments.



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  $\LaTeX$  becomes confused and throws error messages that are difficult to decipher.

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

```

1 \ifnotes\maketitle\else
2 \frame[noframenumbering]\maketitle\fi
3
4 \begin{note}
5   We start this course with ...
6 \end{note}
7
8 \begin{frame}
9   \frametitle{The first slide}
10  ...

```

```

11 \end{frame}
12 \begin{note}
13   ... and more explanatory text
14 \end{note}
15
16 \begin{frame}
17   \frametitle{The second slide}
18   ...
19 \end{frame}
20 ...

```

---

`\ifnotes`

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



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



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

---

`\inputref*`

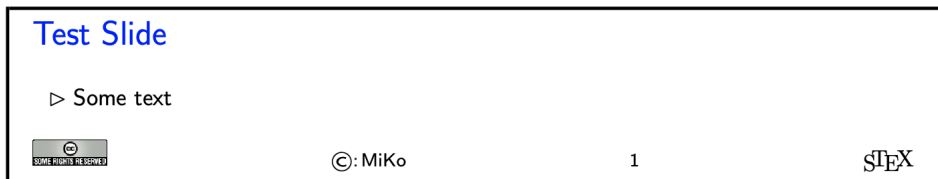
If we want to transclude a the contents of a file as a note, we can use a new variant `\inputref*` of the `\inputref` macro: `\inputref*{foo}` is equivalent to `\begin{note}\inputref{foo}\end{note}`.

`nexample`, `nsproof`, `nassertion`

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

### 6.3.4 Customizing Header and Footer Lines

The `notesslides` package and class comes with a simple default theme named `sTeX` that provided by the `beamterthemesTeX`. It is assumed as the default theme for `sTeX`-based notes and slides. The result in `notes` mode (which is like the `slides` version except that the slide height is variable) is



The footer line can be customized. In particular the logos.

---

<code>\setslidelogo</code>	The default logo provided by the <code>notesslides</code> package is the $\text{\LaTeX}$ logo it can be customized using <code>\setslidelogo{&lt;logo name&gt;}</code> .
----------------------------	--

---



---

<code>\setsource</code>	The default footer line of the <code>notesslides</code> package mentions copyright and licensing. In <code>notesslides \source</code> stores the author's name as the copyright holder. By default it is the author's name as defined in the <code>\author</code> macro in the preamble. <code>\setsource{&lt;name&gt;}</code> can change the writer's name.
-------------------------	--

---



---

<code>\setlicensing</code>	For licensing, we use the Creative Commons Attribution-ShareAlike license by default to strengthen the public domain. If package <code>hyperref</code> is loaded, then we can attach a hyperlink to the license logo. <code>\setlicensing[&lt;url&gt;]{&lt;logo name&gt;}</code> is used for customization, where <code>&lt;url&gt;</code> is optional.
----------------------------	---

---

### 6.3.5 Frame Images

Sometimes, we want to integrate slides as images after all – e.g. because we already have a PowerPoint presentation, to which we want to add  $\text{\LaTeX}$  notes.

---

<code>\frameimage</code> <code>\mhframeimage</code>	In this case we can use <code>\frameimage[&lt;opt&gt;]{&lt;path&gt;}</code> , where <code>&lt;opt&gt;</code> are the options of <code>\includegraphics</code> from the <code>graphicx</code> package [CR99] and <code>&lt;path&gt;</code> is the file path (extension can be left off like in <code>\includegraphics</code> ). We have added the <code>label</code> key that allows to give a frame label that can be referenced like a regular <code>beamer</code> frame.
--	--

---

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

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

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


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

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

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

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

---

<code>\textwarning</code>	The <code>\textwarning</code> macro generates a warning sign: 
---------------------------	---

---

### 6.3.6 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:

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

It generates a paragraph that references the excursion whose source is in the file `../fragments/founif.en.tex` and automatically books the file for the `\printexcursions` command that is used here to put it into the appendix. We will look at the mechanics now.

---

`\excursion`

The `\excursion{<ref>}{<path>}{<text>}` is syntactic sugar for

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

---

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

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

Sometimes, we want to reference – in an excursion – part of another. We can use `\excursionref{<label>}` for that.

---

`\excursiongroup`

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

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



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



## 6.4 Representing Problems and Solutions

### 6.4.1 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>4</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.

### 6.4.2 Problems and Solutions

<hr/> <code>solutions</code>	The <code>problem</code> package takes the options <code>solutions</code> (should solutions be output?), <code>notes</code> (should the problem notes be presented?), <code>hints</code> (do we give the hints?), <code>gnotes</code> (do we show grading notes?), <code>pts</code> (do we display the points awarded for solving the problem?), <code>min</code> (do we display the estimated minutes for problem soling). If theses are specified, then the corresponding auxiliary parts of the problems are output, otherwise, they remain invisible.
<code>notes</code>	
<code>hints</code>	
<code>gnotes</code>	
<code>pts</code>	
<code>min</code>	
<code>boxed</code>	
<code>test</code>	
<hr/>	The <code>boxed</code> option specifies that problems should be formatted in framed boxes so 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>problem</code>	The main environment provided by the <code>problempackage</code> is (surprise surprise) the <code>problem</code> environment. It is used to mark up problems and exercises. The environment takes an optional <code>KeyVal</code> argument with the keys <code>id</code> as an identifier that can be reference later, <code>pts</code> for the points to be gained from this exercise in homework or quiz situations, <code>min</code> for the estimated minutes needed to solve the problem, and finally <code>title</code> for an informative title of the problem.

#### Example 40

Input:

---

<sup>4</sup>for the moment multiple choice problems are not supported, but may well be in a future version

```

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

```

Output:

**Problem 6.4.1 (Fitting Elephants)**  
 How many Elephants can you fit into a Volkswagen beetle?

---

**Hint:** Think positively, this is simple!

---

**Note:** Justify your answer

---

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

---

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

---

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

**hint,exnote,gnote** The `hint` and `exnote` environments can be used in a `problem` environment to give hints and to make notes that elaborate certain aspects of the problem. The `gnote` (grading notes) environment can be used to document situations that may arise in grading.

---

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

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

---

**\ifsolutions**

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

### 6.4.3 Markup for Added-Value Services

The `problem` package is all about specifying the meaning of the various moving parts of practice/exam problems. The motivation for the additional markup is that we can base added-value services from these, for instance auto-grading and immediate feedback.

The simplest example of this are multiple-choice problems, where the `problem` package allows to annotate answer options with the intended values and possibly feedback that can be delivered to the users in an interactive setting. In this section we will give some infrastructure for these, we expect that this will grow over time.

#### Multiple Choice Blocks

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

---

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

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

What we see when this is formatted to PDF depends on the context. In solutions mode (we start the solutions in the code fragment below) we get

#### Example 41

Input:

```

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

```

Output:

#### Problem 6.4.2 (Functions)

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

☐ def

**Correct!**

☐ function

**Wrong!** *that is for C and C++*

☐ fun

**Wrong!** *that is for Standard ML*

☐ public static void

**Wrong!** *that is for Java*

In “exam mode” where disable solutions (here via \stopsolutions)

#### Example 42

Input:

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

Output:

#### Problem 6.4.3 (Functions)

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

☐ def

☐ function

☐ fun

☐ public static void

we get the questions without solutions (that is what the students see during the exam/quiz).

#### Filling-In Concrete Solutions

The next simplest situation, where we can implement auto-grading is the case where we have fill-in-the-blanks

---

`\fillinsol`

The `\fillinsol` macro takes<sup>6</sup> an a single argument, which contains a concrete solution (i.e. a number, a string, ...), which generates a fill-in-box in test mode:

#### Example 43

Input:

```
1 \stopsolutions
2 \begin{problem}[id=elephants.fillin,title=Fitting Elephants]
3   How many Elefants can you fit into a Volkswagen beetle? \fillinsol{4}
4 \end{problem}
```

Output:

##### Problem 6.4.4 (Fitting Elephants)

How many Elefants can you fit into a Volkswagen beetle?

and the actual solution in solutions mode:

#### Example 44

Input:

```
1 \begin{problem}[id=elephants.fillin,title=Fitting Elephants]
2   How many Elefants can you fit into a Volkswagen beetle? \fillinsol{4}
3 \end{problem}
```

Output:

##### Problem 6.4.5 (Fitting Elephants)

How many Elefants can you fit into a Volkswagen beetle?

4!

Obviously, the argument of `\fillinsol` can be used for auto-grading. For concrete data like numbers, this is immediate, for more complex data like strings “soft comparisons” might be in order.<sup>7</sup>

## 6.4.4 Including Problems

---

`\includeproblem`

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

The sum of the points and estimated minutes (that we specified in the `pts` and `min` keys to the `problem` environment or the `\includeproblem` macro) to the log file and the

---

<sup>7</sup>EDNOTE: For the moment we only assume a single concrete value as correct. In the future we will almost certainly want to extend the functionality to multiple answer classes that allow different feedback like in MCQ. This still needs a bit of design. Also we want to make the formatting of the answer in solutions/test mode configurable.

screen after each run. This is useful in preparing exams, where we want to make sure that the students can indeed solve the problems in an allotted time period.

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

## 6.5 Homeworks, Quizzes and Exams

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

### 6.5.2 Package Options

<code>solutions</code>	The <code>hwexam</code> package and class take the options <code>solutions</code> , <code>notes</code> , <code>hints</code> , <code>gnotes</code> , <code>pts</code> , <code>min</code> , and <code>boxed</code> that are just passed on to the <code>problems</code> package (cf. its documentation for a description of the intended behavior).
<code>notes</code>	
<code>hints</code>	
<code>gnotes</code>	
<code>pts</code>	
<code>min</code>	
<code>multiple</code>	Furthermore, the <code>hwexam</code> package takes the option <code>multiple</code> that allows to combine multiple assignment sheets into a compound document (the assignment sheets are treated as section, there is a table of contents, etc.).
<code>test</code>	Finally, there is the option <code>test</code> that modifies the behavior to facilitate formatting tests. Only in <code>test</code> mode, the macros <code>\testspace</code> , <code>\testnewpage</code> , and <code>\testemptypage</code> have an effect: they generate space for the students to solve the given problems. Thus they can be left in the <code>L<sup>A</sup>T<sub>E</sub>X</code> source.

### 6.5.3 Assignments

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

### 6.5.4 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.

### 6.5.5 Typesetting Exams

`\testspace`  
`\testnewpage`  
`\testemptypage`

`\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`  
`duration`  
`min`  
`reqpts`

Finally, the `\testheading` takes an optional keyword argument where the keys `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.

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

Will result in

Name:

Matriculation Number:

## 320101 General Computer Science (Fall 2010)

2022-07-21

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

Write the solutions to the sheet.

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

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

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

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

	To be used for grading, do not write here													
prob.	6.4.1	6.4.2	6.4.3	6.4.4	6.4.5	1.1	2.1	2.2	2.3	3.1	3.2	3.3	Sum	grade
total	10					4	4	6	6	4	4	2	40	
reached														

good luck

EdN:8

8

---

<sup>8</sup>EDNOTE: MK: The first three “problems” come from the stex examples above, how do we get rid of this?



## Part II

# Documentation

# Chapter 7

## sTeX-Basics

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

### 7.1 Macros and Environments

---

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

---

---

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

---

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

#### 7.1.1 HTML Annotations

---

<code>\if@latexml</code>	L <sup>A</sup> T <sub>E</sub> X2e conditional for L <sup>A</sup> T <sub>E</sub> 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  $\langle content \rangle$  with

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

### 7.1.2 Babel Languages

---

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

---

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

### 7.1.3 Auxiliary Methods

---

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

---

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

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

---

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

---

# Chapter 8

## STEX-MathHub

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

### 8.1 Macros and Environments

---

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

---

#### 8.1.1 Files, Paths, URIs

---

<code>\stex_path_from_string:Nn</code>	<code>\stex_path_from_string:Nn</code> $\langle path-variable \rangle$ $\{\langle string \rangle\}$ 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> <code>\stex_filestack_pop:</code>	Push and pop (repectively) a file path to the file stack, to keep track of the current file. Are called in hooks <code>file/before</code> and <code>file/after</code> , respectively.
--	---

---

### 8.1.2 MathHub Archives

---

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

---

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

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

---

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

---

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

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

---

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

---

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

---

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

---



---

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

---

### 8.1.3 Using Content in Archives

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

## Chapter 9

# STEX-References

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

### 9.1 Macros and Environments

---

**\STEXreftitle**

---

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

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

---

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

---

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

---

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

---

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

---

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

---

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

---

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

---

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

#### 9.1.1 Setting Reference Targets

---

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

---

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

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

---

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

---

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

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

### 9.1.2 Using References

---

`\sref`    `\sref[<opt-args>]{<id>}`

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

---

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

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

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

---

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

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



# Chapter 10

## sTeX-Modules

This sub package contains code related to Modules

### 10.1 Macros and Environments

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

---

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

---

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.

---

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

---

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

---

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

---

The names of all constants declared in the module

---

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

---

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

### 10.1.1 The `smodule` environment

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

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

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

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

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

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

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

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

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

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

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

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

---

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

---

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

---

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

---

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

---

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

---

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

---

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

---

---

**`\stex_activate_module:n`**

---

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

# Chapter 11

## STEX-Module Inheritance

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

### 11.1 Macros and Environments

#### 11.1.1 SMS Mode

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

---

`\g_stex_smsmode_allowedmacros_tl`

---

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

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

---

`\g_stex_smsmode_allowedmacros_escape_tl`

---

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

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

---

`\g_stex_smsmode_allowedenvs_seq`

---

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

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

---

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

---

Tests whether SMS mode is currently active.

<hr/> <hr/> <code>\stex_file_in_smsmode:nn</code>	<code>\stex_in_smsmode:nn</code> $\{\langle filename \rangle\}$ $\{\langle code \rangle\}$ Executes $\langle code \rangle$ in SMS mode, followed by the content of $\langle filename \rangle$ . $\langle code \rangle$ can be used e.g. to set the current repository, and is executed within a new tex group, and the same group as the file content.
---	---

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

### 11.1.2 Imports and Inheritance

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

<hr/> <hr/> <code>\usemodule</code>	<code>\importmodule</code> $[\langle archive-ID \rangle]$ $\{\langle module-path \rangle\}$ Like <code>\importmodule</code> , but does not export its contents; i.e. including the current module will not activate the used module
-------------------------------------	--

---

`\stex_import_module_uri:nn`

---

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

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

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

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

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

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

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

2. Otherwise:

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

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

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

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

---

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

---

stores the result in these four variables.

---

`\stex_import_require_module:nnnn {<ns>} {<archive-ID>} {<path>} {<name>}`

---

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

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

# Chapter 12

## STEX-Symbols

Code related to symbol declarations and notations

### 12.1 Macros and Environments

---

$\backslash\text{symdecl}$	$\backslash\text{symdecl}\{\langle\text{macroname}\rangle\}[\langle\text{args}\rangle]$
----------------------------	---

---

Declares a new symbol with semantic macro  $\backslash\text{macroname}$ . Optional arguments are:

- **name**: An (OMDOC) name. By default equal to  $\langle\text{macroname}\rangle$ .
- **type**: An (ideally semantic) term, representing a *type*. Not used by STEX, but passed on to MMT for semantic services.
- **def**: An (ideally semantic) term, representing a *definiens*. Not used by STEX, but passed on to MMT for semantic services.
- **local**: A boolean (by default false). If set, this declaration will not be added to the module content, i.e. importing the current module will not make this declaration available.
- **args**: Specifies the “signature” of the semantic macro. Can be either an integer  $0 \leq n \leq 9$ , or a (more precise) sequence of the following characters:
  - i** a “normal” argument, e.g.  $\backslash\text{symdecl}\{\text{plus}\}[\text{args}=\text{ii}]$  allows for  $\backslash\text{plus}\{2\}\{2\}$ .
  - a** an *associative* argument; i.e. a sequence of arbitrarily many arguments provided as a comma-separated list, e.g.  $\backslash\text{symdecl}\{\text{plus}\}[\text{args}=\text{a}]$  allows for  $\backslash\text{plus}\{2,2,2\}$ .
  - 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.  $\backslash\text{symdecl}\{\text{forall}\}[\text{args}=\text{bi}]$  allows for  $\backslash\text{forall}\{x\in\text{Nat}\}\{x\geq 0\}$ .



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

# Chapter 13

## STEX-Terms

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

### 13.1 Macros and Environments

---

<u>\STEXsymbol</u>	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>
--------------------	--

---

<u>\symref</u>	<code>\symref{&lt;symbol&gt;}{&lt;text&gt;}</code> shortcut for <code>\STEXsymbol{&lt;symbol&gt;}! [&lt;text&gt;]</code>
----------------	---

---

<u>\stex_invoke_symbol:n</u>	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 `!`, it will invoke the symbol *itself* rather than its application (and continue to `\stex_term_custom:nn`), i.e. it allows to refer to `\plus![addition]` as an operation, rather than `\plus[addition of]{some}{terms}`.

---

<u>\STEXInternalTermMathOMSiiii</u>	<code>&lt;URI&gt;&lt;fragment&gt;&lt;precedence&gt;&lt;body&gt;</code>
<u>\STEXInternalTermMathOMAiiai</u>	
<u>\STEXInternalTermMathOMBiiii</u>	

Annotates `<body>` as an OMDOC-term (OMID, OMA or OMBIND, respectively) with head symbol `<URI>`, generated by the specific notation `<fragment>` with (upwards) operator precedence `<precedence>`. Inserts parentheses according to the current downwards precedence and operator precedence.

---

<u>\STEXInternalTermMathArgiii</u>	<code>\stex_term_arg:nnn&lt;int&gt;&lt;prec&gt;&lt;body&gt;</code>
------------------------------------	--

Annotates `<body>` as the `<int>`th argument of the current OMA or OMBIND, with (downwards) argument precedence `<prec>`.

<hr/> <hr/>	<code>\STEXInternalTermMathAssocArgiiii</code>	<code>\stex_term_arg:nnn⟨int⟩⟨prec⟩⟨notation⟩⟨body⟩</code>
		Annotates $\langle body \rangle$ as the $\langle int \rangle$ th (associative) <i>sequence</i> argument (as comma-separated list of terms) of the current OMA or OMBIND, with (downwards) argument precedence $\langle prec \rangle$ and associative notation $\langle notation \rangle$ .
<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 $\TeX$ 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 $\TeX$ for automated bracketing (by default ( and )) to $\langle left \rangle$ and $\langle right \rangle$ . Note that $\langle left \rangle$ and $\langle right \rangle$ need to be allowed after <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>\comp</code> <code>\compemph</code> <code>\compemph@uri</code> <code>\defemph</code> <code>\defemph@uri</code> <code>\symrefemph</code> <code>\symrefemph@uri</code> <code>\varemp</code> <code>\varemp@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 14

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

Code related to structural features

### 14.1 Macros and Environments

#### 14.1.1 Structures

`mathstructure` TODO

## Chapter 15

# sTeX-Statements

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

### 15.1 Macros and Environments

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

## Chapter 16

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

## Chapter 17

# sT<sub>E</sub>X-Metatheory

### 17.1 Symbols

**Part III**  
**Extensions**



## Chapter 18

# Tikzinput: Treating TIKZ code as images

### 18.1 Macros and Environments

## Chapter 19

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

## Chapter 20

# NotesSlides – Slides and Course Notes

## Chapter 21

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

## Chapter 22

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

Part IV

# Implementation

## Chapter 23

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

### 23.1 The $\text{\TeX}$ Document Class

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

```
1 <*cls>
2
3 %%%%%%%%% basics.dtx %%%%%%%%%
4
5 \RequirePackage{expl3,l3keys2e}
6 \ProvidesExplClass{stex}{2022/05/24}{3.1.0}{sTeX document class}
7
8 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
9 \ProcessOptions
10
11 \bool_set_true:N \c_stex_document_class_bool
12
13 \RequirePackage{stex}
14
15 \stex_html_backend:TF {
16   \LoadClass{article}
17 }{
18   \LoadClass[border=1px,varwidth,crop=false]{standalone}
19   \setlength\textwidth{15cm}
20 }
21 \RequirePackage{standalone}
22
23
24 \clist_if_empty:NT \c_stex_languages_clist {
25   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
26   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
27   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
28   \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
29     \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
30       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
```

```

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

```

## 23.2 Preliminaries

```

44 <*package>
45
46 %%%%%%%%% basics.dtx %%%%%%%%%
47
48 \RequirePackage{expl3,l3keys2e,ltxcmds}
49 \ProvidesExplPackage{stex}{2022/05/24}{3.1.0}{sTeX package}
50
51 \bool_if_exist:NF \c_stex_document_class_bool {
52   \bool_set_false:N \c_stex_document_class_bool
53   \RequirePackage{standalone}
54 }
55
56 \message{^^J*~This~is~sTeX~version~3.1.0~*^^J}
57
58 %\RequirePackage{morewrites}
59 %\RequirePackage{amsmath}
60
61 Package options:
62 \keys_define:nn { stex } {
63   debug      .clist_set:N = \c_stex_debug_clist ,
64   lang       .clist_set:N = \c_stex_languages_clist ,
65   mathhub    .tl_set_x:N  = \mathhub ,
66   usesms     .bool_set:N  = \c_stex_persist_mode_bool ,
67   writesms   .bool_set:N  = \c_stex_persist_write_mode_bool ,
68   image      .bool_set:N  = \c_tikzinput_image_bool ,
69   unknown    .code:n      = {}
70 }
71 \ProcessKeysOptions { stex }

```

**\stex** The sTeX logo:

**\sTeX** `\RequirePackage{stex-logo} % externalized for backwards-compatibility reasons`

(End definition for `\stex` and `\sTeX`. These functions are documented on page 68.)

## 23.3 Messages and logging



```

72 <@@=stex_log>
    Warnings and error messages
73 \msg_new:nnn{stex}{error/unknownlanguage}{
74   Unknown~language:~#1
75 }
76 \msg_new:nnn{stex}{warning/nomathhub}{
77   MATHHUB~system~variable~not~found~and~no~
78   \detokenize{\mathhub}~value~set!
79 }
80 \msg_new:nnn{stex}{error/deactivated-macro}{
81   The~\detokenize{#1}~command~is~only~allowed~in~#2!
82 }

```

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

```

83 \cs_new_protected:Nn \stex_debug:nn {
84   \clist_if_in:NnTF \c_stex_debug_clist { all } {
85     \msg_set:nnn{stex}{debug / #1}{
86       \\Debug~#1:~#2\\
87     }
88     \msg_none:nn{stex}{debug / #1}
89   }{
90     \clist_if_in:NnT \c_stex_debug_clist { #1 } {
91       \msg_set:nnn{stex}{debug / #1}{
92         \\Debug~#1:~#2\\
93       }
94       \msg_none:nn{stex}{debug / #1}
95     }
96   }
97 }

```

(End definition for \stex\_debug:nn. This function is documented on page 68.)

Redirecting messages:

```

98 \clist_if_in:NnTF \c_stex_debug_clist {all} {
99   \msg_redirect_module:nnn{ stex }{ none }{ term }
100 }{
101   \clist_map_inline:Nn \c_stex_debug_clist {
102     \msg_redirect_name:nnn{ stex }{ debug / ##1 }{ term }
103   }
104 }
105
106 \stex_debug:nn{log}{debug~mode~on}

```

## 23.4 HTML Annotations

```

107 <@@=stex_annotate>

```

**\l\_stex\_html\_arg\_tl** Used by annotation macros to ensure that the HTML output to annotate is not empty.  
**\c\_stex\_html\_emptyarg\_tl**

```

108 \tl_new:N \l_stex_html_arg_tl

```

(End definition for \l\_stex\_html\_arg\_tl and \c\_stex\_html\_emptyarg\_tl. These variables are documented on page ??.)

`\stex_html_checkempty:n`

```

109 \cs_new_protected:Nn \stex_html_checkempty:n {
110   \tl_set:Nn \l_stex_html_arg_tl { #1 }
111   \tl_if_empty:NT \l_stex_html_arg_tl {
112     \tl_set_eq:NN \l_stex_html_arg_tl \c_stex_html_emptyarg_tl
113   }
114 }

```

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

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

`\stex_if_do_html:TF`

```

115 \bool_new:N \stex_html_do_output_bool
116 \bool_set_true:N \stex_html_do_output_bool
117
118 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
119   \bool_if:nTF \stex_html_do_output_bool
120     \prg_return_true: \prg_return_false:
121 }

```

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

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

```

122 \cs_new_protected:Nn \stex_suppress_html:n {
123   \exp_args:Nne \use:nn {
124     \bool_set_false:N \stex_html_do_output_bool
125     #1
126   }{
127     \stex_if_do_html:T {
128       \bool_set_true:N \stex_html_do_output_bool
129     }
130   }
131 }

```

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

`\stex_annotate:anv`

`\stex_annotate_invisible:n`

`\stex_annotate_invisible:nnn`

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

The p<sub>D</sub>F<sub>L</sub>A<sub>T</sub>E<sub>X</sub>-macros largely do nothing; the R<sub>U</sub>S<sub>E</sub>TEX-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.

```

132 \tl_if_exist:NF\stex@backend{
133   \ifcsname if@rustex\endcsname
134   \def\stex@backend{rustex}
135   \else
136     \ifcsname if@latexml\endcsname
137     \def\stex@backend{latexml}
138     \else
139     \def\stex@backend{pdflatex}
140   \fi
141   \fi
142 }
143 \input{stex-backend-\stex@backend.cfg}
144
145 \newif\ifstexhtml
146 \stex_html_backend:TF\stexhtmltrue\stexhtmlfalse
147

```

(End definition for `\stex_annotate:nnn`, `\stex_annotate_invisible:n`, and `\stex_annotate_invisible:nnn`. These functions are documented on page 69.)

## 23.5 Babel Languages

148 `<@=stex_language>`

`\c_stex_languages_prop`  
`\c_stex_language_abbrevs_prop`

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

```

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

```

(End definition for `\c_stex_languages_prop` and `\c_stex_language_abbrevs_prop`. These variables are documented on page 69.)

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

```

174 \cs_new_protected:Nn \stex_set_language:Nn {
175   \str_set:Nx \l_tmpa_str {#2}
176   \prop_get:NoNT \c_stex_languages_prop \l_tmpa_str #1 {
177     \ifx\@onlypreamble\@notprerr
178       \ltx@ifpackageloaded{babel}{
179         \exp_args:No \selectlanguage #1
180       }{}
181     \else
182       \exp_args:No \str_if_eq:nnTF #1 {turkish} {
183         \RequirePackage[#1,shorthands=:!]{babel}
184       }{
185         \RequirePackage[#1]{babel}
186       }
187     \fi
188   }
189 }
190

```

```

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

```

## 23.6 Persistence

```

228 <@@=stex_persist>
229 \bool_if:NTF \c_stex_persist_mode_bool {
230   \def \stex_persist:n #1 {}
231   \def \stex_persist:x #1 {}
232 }{
233   \bool_if:NTF \c_stex_persist_write_mode_bool {
234     \iow_new:N \c__stex_persist_iow
235     \iow_open:Nn \c__stex_persist_iow{\jobname.sms}
236     \AtEndDocument{
237       \iow_close:N \c__stex_persist_iow
238     }
239     \cs_new_protected:Nn \stex_persist:n {
240       \tl_set:Nn \l_tmpa_tl { #1 }

```

```

241 \regex_replace_all:nnN { \cP\# } { \cO\# } \l_tmpa_tl
242 \regex_replace_all:nnN { \ } { \~ } \l_tmpa_tl
243 \exp_args:NNo \iow_now:Nn \c__stex_persist_iow \l_tmpa_tl
244 }
245 \cs_generate_variant:Nn \stex_persist:n {x}
246 }{
247 \def \stex_persist:n #1 {}
248 \def \stex_persist:x #1 {}
249 }
250 }

```

## 23.7 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}{\detokenize{#1}}{#2}
255 }
256 }

```

(End definition for \stex\_deactivate\_macro:Nn. This function is documented on page 69.)

**\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 69.)

**\ignorespacesandpars**

```

260 \protected\def\ignorespacesandpars{
261 \begingroup\catcode13=10\relax
262 \@ifnextchar\par{
263 \endgroup\expandafter\ignorespacesandpars\@gobble
264 }{
265 \endgroup
266 }
267 }
268
269 \cs_new_protected:Nn \stex_copy_control_sequence:NNN {
270 \tl_set:Nx \_tmp_args_tl {\cs_argument_spec:N #2}
271 \exp_args:NNo \tl_remove_all:Nn \_tmp_args_tl \c_hash_str
272 \int_set:Nn \l_tmpa_int {\tl_count:N \_tmp_args_tl}
273
274 \tl_clear:N \_tmp_args_tl
275 \int_step_inline:nn \l_tmpa_int {
276 \tl_put_right:Nx \_tmp_args_tl {\exp_not:n{####}\exp_not:n{##1}}
277 }
278
279 \tl_set:Nn #3 {\cs_generate_from_arg_count:NNnn #1 \cs_set:Npn}
280 \tl_put_right:Nx #3 { {\int_use:N \l_tmpa_int}{
281 \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
282 \exp_after:wN\exp_after:wN\exp_after:wN {

```

```

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

```

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

`\MMTrule`

```

327 \NewDocumentCommand \MMTrule {m m}{
328   \seq_set_split:Nnn \l_tmpa_seq , {#2}
329   \int_zero:N \l_tmpa_int
330   \stex_annotate_invisible:nnn{mmtrule}{scala://#1}{
331     \seq_if_empty:NF \l_tmpa_seq {
332       $\seq_map_inline:Nn \l_tmpa_seq {

```

```

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

```

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

## Chapter 24

# STEX -MathHub Implementation

```
381 <*package>
382
383 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
384
385 <@@=stex_path>
386
387 Warnings and error messages
388 \msg_new:nnn{stex}{error/norepository}{
389   No~archive~#1~found~in~#2
390 }
391 \msg_new:nnn{stex}{error/notinarchive}{
392   Not~currently~in~an~archive,~but~\detokenize{#1}~
393   needs~one!
394 }
395 \msg_new:nnn{stex}{error/nofile}{
396   \detokenize{#1}~could~not~find~file~#2
397 }
398 \msg_new:nnn{stex}{error/twofiles}{
399   \detokenize{#1}~found~two~candidates~for~#2
400 }
```

### 24.1 Generic Path Handling

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

`\stex_path_from_string:Nn`

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



```

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

```

(End definition for `\stex_path_from_string:Nn`. This function is documented on page 70.)

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

```

417 \cs_new_protected:Nn \stex_path_to_string:NN {
418   \exp_args:Nne \str_set:Nn #2 { \seq_use:Nn #1 / }
419 }
420
421 \cs_new:Nn \stex_path_to_string:N {
422   \seq_use:Nn #1 /
423 }

```

(End definition for `\stex_path_to_string:NN` and `\stex_path_to_string:N`. These functions are documented on page 70.)

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

```

424 \str_const:Nn \c__stex_path_dot_str {.}
425 \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.

```

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

```

```

448         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
449     }
450 }
451 }{
452     \str_if_empty:NF \l_tmpa_tl {
453         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
454     }
455 }
456 }
457 }
458 \seq_gset_eq:NN #1 \l_tmpa_seq
459 }
460 }

```

(End definition for `\stex_path_canonicalize:N`. This function is documented on page 70.)

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

```

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

```

(End definition for `\stex_path_if_absolute:N $\underline{TF}$` . This function is documented on page 70.)

## 24.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

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

```

(End definition for `\stex_kpsewhich:n`. This function is documented on page 70.)

We determine the PWD

`\c_stex_pwd_seq`  
`\c_stex_pwd_str`

```
490 \sys_if_platform_windows:TF{
491   \begingroup\escapechar=-1\catcode'\=12
492   \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
493   \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
494   \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_
495   }}{
496   \stex_kpsewhich:n{-var-value~PWD}
497 }
498
499 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
500 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
501 \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 70.)

## 24.3 File Hooks and Tracking

502 `<@@=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

503 `\seq_gclear_new:N\g__stex_files_stack`

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`  
`\c_stex_mainfile_str`

```
504 \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
505 \stex_path_from_string:Nn \c_stex_mainfile_seq
506   \c_stex_mainfile_str
```

(End definition for `\c_stex_mainfile_seq` and `\c_stex_mainfile_str`. These variables are documented on page 70.)

`\g_stex_currentfile_seq`

507 `\seq_gclear_new:N\g_stex_currentfile_seq`

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

`\stex_filestack_push:n`

```
508 \cs_new_protected:Nn \stex_filestack_push:n {
509   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
510   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
511     \stex_path_from_string:Nn\g_stex_currentfile_seq{
512       \c_stex_pwd_str/#1
513     }
514   }
```

```

514 }
515 \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
516 \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
517 }

```

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

**`\stex_filestack_pop:`**

```

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

```

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

Hooks for the current file:

```

529 \AddToHook{file/before}{
530   \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
531 }
532 \AddToHook{file/after}{
533   \stex_filestack_pop:
534 }

```

## 24.4 MathHub Repositories

```

535 <@=stex_mathhub>

```

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

**`\c_stex_mathhub_str`**

```

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

```

```

554   }{
555     \stex_kpsewhich:n{-var-value-HOME}
556   }
557   \ior_open:NnT \g_tmpa_ior{\l_stex_kpsewhich_return_str / .stex / mathhub.path}{
558     \begingroup\escapechar=-1\catcode'\=12
559     \ior_str_get:NN \g_tmpa_ior \l_tmpa_str
560     \sys_if_platform_windows:T{
561       \exp_args:NNx\str_replace_all:Nnn\l_tmpa_str{\c_backslash_str}/
562     }
563     \str_gset_eq:NN \c_stex_mathhub_str\l_tmpa_str
564     \endgroup
565     \ior_close:N \g_tmpa_ior
566   }
567 }
568 \str_if_empty:NTF\c_stex_mathhub_str{
569   \msg_warning:nn{stex}{warning/nomathhub}
570 }{
571   \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
572   \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
573 }
574 }{
575   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
576   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
577     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
578       \c_stex_pwd_str/\mathhub
579     }
580   }
581   \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
582   \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
583 }

```

(End definition for `\mathhub`, `\c_stex_mathhub_seq`, and `\c_stex_mathhub_str`. These variables are documented on page 71.)

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

```

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

```

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

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

601 \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:

```

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

```

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

\c\_stex\_mathhub\_manifest\_ior

File variable used for MANIFEST-files

637 \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:

```

638 \cs_new_protected:Nn \_stex_mathhub_parse_manifest:n {
639   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
640   \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}

```

```

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

```

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

\stex\_set\_current\_repository:n

```

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

```

(End definition for \stex\_set\_current\_repository:n. This function is documented on page 71.)

`\stex_require_repository:n`

```

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

```

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

`\l_stex_current_repository_prop`

Current MathHub repository

```

696 %\prop_new:N \l_stex_current_repository_prop
697 \bool_if:NF \c_stex_persist_mode_bool {
698   \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
699   \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
700     \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
701   } {
702     \__stex_mathhub_parse_manifest:n { main }
703     \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
704     \l_tmpa_str
705     \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str_manifest_prop }
706     \c_stex_mathhub_main_manifest_prop
707     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
708     \stex_debug:nn{mathhub}{Current~repository:~
709     \prop_item:Nn \l_stex_current_repository_prop {id}
710   }
711 }
712 }

```

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

`\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.

```

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

```



```

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

```

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

## 24.5 Using Content in Archives

`\mhpath`

```

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

```

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

`\inputref`

`\mhinput`

```

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

```

```

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

```

(End definition for `\inputref` and `\mhinput`. These functions are documented on page 72.)

### `\addmhbibresource`

```

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

```

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

## `\libinput`

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

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

## `\libusepackage`

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

```

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

```

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

`\mhgraphics`  
`\cmhgraphics`

```

896
897 \AddToHook{begindocument}{
898 \ltx@ifpackageloaded{graphicx}{
899 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
900 \providecommand\mhgraphics[2] [] {%
901 \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
902 \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}}
903 \providecommand\cmhgraphics[2] [] {\begin{center}\mhgraphics[#1]{#2}\end{center}}
904 }{}

```

(End definition for `\mhgraphics` and `\cmhgraphics`. These functions are documented on page 72.)

`\lstinputmhlisting`  
`\clstinputmhlisting`

```

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

```

(End definition for `\lstinputmhlisting` and `\clstinputmhlisting`. These functions are documented on page 72.)

## Chapter 25

# STEX -References Implementation

```
915 <*package>
916
917 %%%%%%%%% stex-references.dtx %%%%%%%%%
918
919 <@@=stex_refs>
```

Warnings and error messages

```
920
```

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

```
921 %\iow_new:N \c__stex_refs_refs_iow
922 \AtBeginDocument{
923 % \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
924 }
925 \AtEndDocument{
926 % \iow_close:N \c__stex_refs_refs_iow
927 }
```

`\STEXreftitle`

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

*(End definition for `\STEXreftitle`. This function is documented on page 73.)*

### 25.1 Document URIs and URLs

`\l_stex_current_docns_str`

```
933 \str_new:N \l_stex_current_docns_str
```

*(End definition for `\l_stex_current_docns_str`. This variable is documented on page 73.)*

`\stex_get_document_uri:`

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

(End definition for `\stex_get_document_uri:`. This function is documented on page 73.)

`\l_stex_current_docurl_str`

```
974 \str_new:N \l_stex_current_docurl_str
```

(End definition for `\l_stex_current_docurl_str`. This variable is documented on page 73.)

`\stex_get_document_url:`

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

```

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

```

(End definition for `\stex_get_document_url`:. This function is documented on page 73.)

## 25.2 Setting Reference Targets

```

1017 \str_const:Nn \c__stex_refs_url_str{URL}
1018 \str_const:Nn \c__stex_refs_ref_str{REF}
1019 \str_new:N \l__stex_refs_curr_label_str
1020 % @currentlabel -> number
1021 % @currentlabelname -> title
1022 % @currentHref -> name.number <- id of some kind
1023 % \theH# -> \arabic{section}
1024 % \the# -> number
1025 % \hyper@makecurrent{#}
1026 \int_new:N \l__stex_refs_unnamed_counter_int

```

`\stex_ref_new_doc_target:n`

```

1027 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
1028   \stex_get_document_uri:
1029   \str_clear:N \l__stex_refs_curr_label_str
1030   \str_set:Nx \l_tmpa_str { #1 }
1031   \str_if_empty:NT \l_tmpa_str {
1032     \int_incr:N \l__stex_refs_unnamed_counter_int
1033     \str_set:Nx \l_tmpa_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
1034   }
1035   \str_set:Nx \l__stex_refs_curr_label_str {
1036     \l_stex_current_docns_str?\l_tmpa_str
1037   }
1038   \seq_if_exist:cF{g__stex_refs_labels_\l_tmpa_str_seq}{
1039     \seq_new:c {g__stex_refs_labels_\l_tmpa_str_seq}
1040   }
1041   \seq_if_in:coF{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str {
1042     \seq_gput_right:co{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str
1043   }
1044   \stex_if_smsmode:TF {
1045     \stex_get_document_url:
1046     \str_gset_eq:cN {sref_url_\l__stex_refs_curr_label_str_str}\l_stex_current_docurl_str
1047     \str_gset_eq:cN {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_url_str
1048   }{
1049     %\iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter\unexpanded\expandafter{
1050     \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
1051     \immediate\write\@auxout{\stexauxadddocref{\l_stex_current_docns_str}{\l_tmpa_str}}
1052     \str_gset:cx {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_ref_str
1053   }
1054 }

```

(End definition for `\stex_ref_new_doc_target:n`. This function is documented on page 73.)

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

```

1055 \cs_new_protected:Npn \stexauxadddocref #1 #2 {
1056   \str_set:Nn \l_tmpa_str {#1?#2}
1057   \str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
1058   \seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
1059     \seq_new:c {g__stex_refs_labels_#2_seq}
1060   }
1061   \seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
1062     \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
1063   }
1064 }

```

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

```

1065 \AtEndDocument{
1066   \def\stexauxadddocref#1 #2 {}{}
1067 }

```

`\stex_ref_new_sym_target:n`

```

1068 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
1069   \stex_if_smsmode:TF {
1070     \str_if_exist:cF{sref_sym_#1_type}{
1071       \stex_get_document_url:
1072       \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str

```



```

1073     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
1074   }
1075   ){
1076     \str_if_empty:NF \l__stex_refs_curr_label_str {
1077       \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
1078       \immediate\write\@auxout{
1079         \exp_not:N\expandafter\def\exp_not:N\csname \exp_not:N\detokenize{sref_sym_#1_label_
1080           \l__stex_refs_curr_label_str
1081         }
1082       }
1083     }
1084   }
1085 }

```

(End definition for `\stex_ref_new_sym_target:n`. This function is documented on page 73.)

## 25.3 Using References

```

1086 \str_new:N \l__stex_refs_indocument_str

```

**\sref** Optional arguments:

```

1087
1088 \keys_define:nn { stex / sref } {
1089   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
1090   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
1091   pre           .tl_set:N = \l__stex_refs_pre_tl ,
1092   post          .tl_set:N = \l__stex_refs_post_tl ,
1093 }
1094 \cs_new_protected:Nn \__stex_refs_args:n {
1095   \tl_clear:N \l__stex_refs_linktext_tl
1096   \tl_clear:N \l__stex_refs_fallback_tl
1097   \tl_clear:N \l__stex_refs_pre_tl
1098   \tl_clear:N \l__stex_refs_post_tl
1099   \str_clear:N \l__stex_refs_repo_str
1100   \keys_set:nn { stex / sref } { #1 }
1101 }

```

The actual macro:

```

1102 \NewDocumentCommand \sref { 0{} m}{
1103   \__stex_refs_args:n { #1 }
1104   \str_if_empty:NTF \l__stex_refs_indocument_str {
1105     \str_set:Nx \l_tmpa_str { #2 }
1106     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
1107     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
1108       \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
1109         \seq_get_left:cNF {g__stex_refs_labels_\l_tmpa_str _seq} \l_tmpa_str {
1110           \str_clear:N \l_tmpa_str
1111         }
1112       }{
1113         \str_clear:N \l_tmpa_str
1114       }
1115     }{
1116       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1117       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str

```

```

1118 \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
1119 \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str_seq}{
1120   \str_set_eq:NN \l_tmpc_str \l_tmpa_str
1121   \str_clear:N \l_tmpa_str
1122   \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str_seq} {
1123     \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
1124       \str_range:nnn { ##1 }{-\l_tmpa_int}{ -1 }
1125     }{
1126       \seq_map_break:n {
1127         \str_set:Nn \l_tmpa_str { ##1 }
1128       }
1129     }
1130   }
1131 }{
1132   \str_clear:N \l_tmpa_str
1133 }
1134 }
1135 \str_if_empty:NTF \l_tmpa_str {
1136   \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_lin
1137 }{
1138   \str_if_eq:cNTF {sref_\l_tmpa_str_type} \c__stex_refs_ref_str {
1139     \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1140       \cs_if_exist:cTF{autoref}{
1141         \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1142       }{
1143         \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1144       }
1145     }{
1146       \ltx@ifpackageloaded{hyperref}{
1147         \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
1148       }{
1149         \l__stex_refs_linktext_tl
1150       }
1151     }
1152   }{
1153     \ltx@ifpackageloaded{hyperref}{
1154       \href{\use:c{sref_url_\l_tmpa_str_str}}{\tl_if_empty:NTF \l__stex_refs_linktext_t
1155     }{
1156       \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs
1157     }
1158   }
1159 }
1160 }{
1161   % TODO
1162 }
1163 }

```

(End definition for \sref. This function is documented on page 74.)

### \srefsym

```

1164 \NewDocumentCommand \srefsym { 0{} m}{
1165   \stex_get_symbol:n { #2 }
1166   \__stex_refs_sym_aux:nn{##1}{\l_stex_get_symbol_uri_str}
1167 }

```

```

1168
1169 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1170   \str_if_exist:cTF {sref_sym_#2 _label_str }{
1171     \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1172   }{
1173     \__stex_refs_args:n { #1 }
1174     \str_if_empty:NTF \l__stex_refs_indocument_str {
1175       \tl_if_exist:cTF{sref_sym_#2 _type}{
1176         % doc uri in \l_tmpb_str
1177         \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1178         \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1179           % reference
1180           \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1181             \cs_if_exist:cTF{autoref}{
1182               \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1183             }{
1184               \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1185             }
1186           }{
1187             \ltx@ifpackageloaded{hyperref}{
1188               \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1189             }{
1190               \l__stex_refs_linktext_tl
1191             }
1192           }
1193         }{
1194           % URL
1195           \ltx@ifpackageloaded{hyperref}{
1196             \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl \
1197           }{
1198             \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_re
1199           }
1200         }
1201       }{
1202         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_l
1203       }
1204     }{
1205       % TODO
1206     }
1207   }
1208 }

```

(End definition for \srefsym. This function is documented on page 74.)

**\srefsymuri**

```

1209 \cs_new_protected:Npn \srefsymuri #1 #2 {
1210   \__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1211 }

```

(End definition for \srefsymuri. This function is documented on page 74.)

```

1212 </package>

```

## Chapter 26

# STEX -Modules Implementation

```
1213 <*package>
1214
1215 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1216
1217 <@@=stex_modules>
1218
1219     Warnings and error messages
1218 \msg_new:nnn{stex}{error/unknownmodule}{
1219     No~module~#1~found
1220 }
1221 \msg_new:nnn{stex}{error/syntax}{
1222     Syntax~error:~#1
1223 }
1224 \msg_new:nnn{stex}{error/siglanguage}{
1225     Module~#1~declares~signature~#2,~but~does~not~
1226     declare~its~language
1227 }
1228 \msg_new:nnn{stex}{warning/deprecated}{
1229     #1~is~deprecated;~please~use~#2~instead!
1230 }
1231
1232 \msg_new:nnn{stex}{error/conflictingmodules}{
1233     Conflicting~imports~for~module~#1
1234 }
```

```
\l_stex_current_module_str The current module:
1235 \str_new:N \l_stex_current_module_str
1236
1237 (End definition for \l_stex_current_module_str. This variable is documented on page 76.)

\l_stex_all_modules_seq Stores all available modules
1236 \seq_new:N \l_stex_all_modules_seq
1237
1238 (End definition for \l_stex_all_modules_seq. This variable is documented on page 76.)
```

```

\stex_if_in_module_p:
\stex_if_in_module:TF
1237 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1238   \str_if_empty:NTF \l_stex_current_module_str
1239   \prg_return_false: \prg_return_true:
1240 }

(End definition for \stex_if_in_module:TF. This function is documented on page 76.)

```

```

\stex_if_module_exists_p:n
\stex_if_module_exists:nTF
1241 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1242   \prop_if_exist:cTF { c_stex_module_#1_prop }
1243   \prg_return_true: \prg_return_false:
1244 }

(End definition for \stex_if_module_exists:nTF. This function is documented on page 76.)

```

```

\stex_add_to_current_module:n
\STEXexport
1245 \cs_new_protected:Nn \stex_execute_in_module:n { \stex_if_in_module:T {
1246   \stex_add_to_current_module:n { #1 }
1247   \stex_do_up_to_module:n { #1 }
1248 }}
1249 \cs_generate_variant:Nn \stex_execute_in_module:n {x}
1250
1251 \cs_new_protected:Nn \stex_add_to_current_module:n {
1252   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str_code} { #1 }
1253 }
1254 \cs_generate_variant:Nn \stex_add_to_current_module:n {x}
1255 \cs_new_protected:Npn \STEXexport {
1256   \ExplSyntaxOn
1257   \__stex_modules_export:n
1258 }
1259 \cs_new_protected:Nn \__stex_modules_export:n {
1260   \ignorespacesandpars#1\ExplSyntaxOff
1261   \stex_add_to_current_module:n { \ignorespacesandpars#1}
1262   \stex_smsmode_do:
1263 }
1264 \let \stex_module_export_helper:n \use:n
1265 \stex_deactivate_macro:Nn \STEXexport {module~environments}

(End definition for \stex_add_to_current_module:n and \STEXexport. These functions are documented
on page 76.)

```

```

\stex_add_constant_to_current_module:n
1266 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1267   \str_set:Nx \l_tmpa_str { #1 }
1268   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str_constants} { \l_tmpa_str }
1269 }

(End definition for \stex_add_constant_to_current_module:n. This function is documented on page
76.)

```

```

\stex_add_import_to_current_module:n
1270 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1271   \str_set:Nx \l_tmpa_str { #1 }
1272   \exp_args:Nno

```

```

1273 \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1274 \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1275 }
1276 }

```

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

`\stex_collect_imports:n`

```

1277 \cs_new_protected:Nn \stex_collect_imports:n {
1278 \seq_clear:N \l_stex_collect_imports_seq
1279 \__stex_modules_collect_imports:n {#1}
1280 }
1281 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1282 \seq_map_inline:cn {c_stex_module_#1_imports} {
1283 \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1284 \__stex_modules_collect_imports:n { ##1 }
1285 }
1286 }
1287 \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1288 \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1289 }
1290 }

```

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

`\stex_do_up_to_module:n`

```

1291 \int_new:N \l__stex_modules_group_depth_int
1292 \cs_new_protected:Nn \stex_do_up_to_module:n {
1293 \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1294 #1
1295 }{
1296 #1
1297 \expandafter \tl_gset:Nn
1298 \csname l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1299 \expandafter\expandafter\expandafter\endcsname
1300 \expandafter\expandafter\expandafter { \csname
1301 l__stex_modules_aftergroup_\l_stex_current_module_str _tl\endcsname #1 }
1302 \aftergroup\__stex_modules_aftergroup_do:
1303 }
1304 }
1305 \cs_generate_variant:Nn \stex_do_up_to_module:n {x}
1306 \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1307 \stex_debug:nn{aftergroup}{\cs_meaning:c{
1308 l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1309 }}}
1310 \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1311 \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1312 \tl_gclear:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1313 }{
1314 \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1315 \aftergroup\__stex_modules_aftergroup_do:
1316 }
1317 }
1318 \cs_new_protected:Nn \stex_reset_up_to_module:n {
1319 \expandafter\let\csname l__stex_modules_aftergroup_#1_tl\endcsname\undefined

```

1320 }

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

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

1321

(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.

```

1322 \str_new:N \l_stex_module_ns_str
1323 \str_new:N \l_stex_module_subpath_str
1324 \cs_new_protected:Nn __stex_modules_compute_namespace:nN {
1325   \seq_set_eq:NN \l_tmpa_seq #2
1326   % split off file extension
1327   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str % <- filename
1328   \exp_args:Nnno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1329   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str % <- filename without suffixes
1330   \seq_put_right:No \l_tmpa_seq \l_tmpb_str % <- file path including name without suffixes
1331
1332   \bool_set_true:N \l_tmpa_bool
1333   \bool_while_do:Nn \l_tmpa_bool {
1334     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1335     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1336       {source} { \bool_set_false:N \l_tmpa_bool }
1337     }{}{
1338       \seq_if_empty:NT \l_tmpa_seq {
1339         \bool_set_false:N \l_tmpa_bool
1340       }
1341     }
1342   }
1343
1344   \stex_path_to_string:NN \l_tmpa_seq \l_stex_module_subpath_str
1345   % \l_tmpa_seq <- sub-path relative to archive
1346   \str_if_empty:NTF \l_stex_module_subpath_str {
1347     \str_set:Nx \l_stex_module_ns_str {#1}
1348   }{
1349     \str_set:Nx \l_stex_module_ns_str {
1350       #1/\l_stex_module_subpath_str
1351     }
1352   }
1353 }
1354
1355 \cs_new_protected:Nn \stex_modules_current_namespace: {
1356   \str_clear:N \l_stex_module_subpath_str
1357   \prop_if_exist:NTF \l_stex_current_repository_prop {
1358     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1359     __stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1360   }{
1361     % split off file extension
1362     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1363     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str

```

```

1364 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1365 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1366 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1367 \str_set:Nx \l_stex_module_ns_str {
1368   file:/\stex_path_to_string:N \l_tmpa_seq
1369 }
1370 }
1371 }

```

(End definition for `\stex_modules_current_namespace:.` This function is documented on page 77.)

## 26.1 The smodule environment

smodule arguments:

```

1372 \keys_define:nn { stex / module } {
1373   title      .tl_set:N      = \smodulename ,
1374   type       .str_set_x:N   = \smodulename ,
1375   id         .str_set_x:N   = \smoduleid ,
1376   deprecate  .str_set_x:N   = \l_stex_module_deprecate_str ,
1377   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1378   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1379   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1380   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1381   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1382   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1383   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1384 }
1385
1386 \cs_new_protected:Nn \__stex_modules_args:n {
1387   \str_clear:N \smodulename
1388   \str_clear:N \smodulename
1389   \str_clear:N \smoduleid
1390   \str_clear:N \l_stex_module_ns_str
1391   \str_clear:N \l_stex_module_deprecate_str
1392   \str_clear:N \l_stex_module_lang_str
1393   \str_clear:N \l_stex_module_sig_str
1394   \str_clear:N \l_stex_module_creators_str
1395   \str_clear:N \l_stex_module_contributors_str
1396   \str_clear:N \l_stex_module_meta_str
1397   \str_clear:N \l_stex_module_srccite_str
1398   \keys_set:nn { stex / module } { #1 }
1399 }
1400
1401 % module parameters here? In the body?
1402

```

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

```

1403 \cs_new_protected:Nn \stex_module_setup:nn {
1404   \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1405   \str_set:Nx \l_stex_module_name_str { #2 }
1406   \__stex_modules_args:n { #1 }

```



First, we set up the name and namespace of the module.

Are we in a nested module?

```

1407 \stex_if_in_module:TF {
1408   % Nested module
1409   \prop_get:cnN {c_stex_module\_l_stex_current_module_str _prop}
1410   { ns } \l_stex_module_ns_str
1411   \str_set:Nx \l_stex_module_name_str {
1412     \prop_item:cn {c_stex_module\_l_stex_current_module_str _prop}
1413     { name } / \l_stex_module_name_str
1414   }
1415   \str_if_empty:NT \l_stex_module_lang_str {
1416     \str_set:Nx \l_stex_module_lang_str {
1417       \prop_item:cn {c_stex_module\_l_stex_current_module_str _prop}
1418       { lang }
1419     }
1420   }
1421 }{
1422   % not nested:
1423   \str_if_empty:NT \l_stex_module_ns_str {
1424     \stex_modules_current_namespace:
1425     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1426       / {\l_stex_module_ns_str}
1427     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1428     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1429       \str_set:Nx \l_stex_module_ns_str {
1430         \stex_path_to_string:N \l_tmpa_seq
1431       }
1432     }
1433   }
1434 }

```

Next, we determine the language of the module:

```

1435 \str_if_empty:NT \l_stex_module_lang_str {
1436   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1437   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1438   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1439   \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
1440     \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
1441       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
1442     }
1443   }
1444   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1445   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
1446     \seq_pop_right:NN \l_tmpa_seq \l_stex_module_lang_str
1447     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1448       inferred~from~file~name}
1449   }
1450 }
1451
1452 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
1453   \exp_args:NNo \stex_set_language:Nn \l_tmpa_str \l_stex_module_lang_str
1454 }}

```

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

```

1455 \str_if_empty:NTF \l_stex_module_sig_str {
1456   \exp_args:Nnx \prop_gset_from_keyval:cn {
1457     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1458   } {
1459     name      = \l_stex_module_name_str ,
1460     ns        = \l_stex_module_ns_str ,
1461     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1462     lang      = \l_stex_module_lang_str ,
1463     sig       = \l_stex_module_sig_str ,
1464     deprecate = \l_stex_module_deprecate_str ,
1465     meta      = \l_stex_module_meta_str
1466   }
1467   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1468   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1469   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _copymodules}
1470   \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1471   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1472 \str_if_empty:NT \l_stex_module_meta_str {
1473   \str_set:Nx \l_stex_module_meta_str {
1474     \c_stex_metatheory_ns_str ? Metatheory
1475   }
1476 }
1477 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1478   \bool_set_true:N \l_stex_in_meta_bool
1479   \exp_args:Nx \stex_add_to_current_module:n {
1480     \bool_set_true:N \l_stex_in_meta_bool
1481     \stex_activate_module:n {\l_stex_module_meta_str}
1482     \bool_set_false:N \l_stex_in_meta_bool
1483   }
1484   \stex_activate_module:n {\l_stex_module_meta_str}
1485   \bool_set_false:N \l_stex_in_meta_bool
1486 }
1487 }{
1488   \str_if_empty:NT \l_stex_module_lang_str {
1489     \msg_error:nnxx{stex}{error/siglanguage}{
1490       \l_stex_module_ns_str?\l_stex_module_name_str
1491     }{\l_stex_module_sig_str}
1492   }
1493   \stex_debug:nn{modules}{Signature~\l_stex_module_sig_str~for~\l_stex_module_ns_str?\l_st
1494   \stex_if_module_exists:nTF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1495     \stex_debug:nn{modules}{(already exists)}
1496   }{
1497     \stex_debug:nn{modules}{(needs loading)}
1498     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1499     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1500     \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1501     \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1502     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1503     \str_set:Nx \l_tmpa_str {
1504       \stex_path_to_string:N \l_tmpa_seq /

```

```

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

```

(End definition for `\stex_module_setup:nn`. This function is documented on page 77.)

**smodule** The module environment.

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

```

1536 \cs_new_protected:Nn \_stex_modules_begin_module: {
1537   \stex_reactivate_macro:N \STEXexport
1538   \stex_reactivate_macro:N \importmodule
1539   \stex_reactivate_macro:N \symdecl
1540   \stex_reactivate_macro:N \notation
1541   \stex_reactivate_macro:N \symdef
1542
1543   \stex_debug:nn{modules}{
1544     New~module:\\
1545     Namespace:~\l_stex_module_ns_str\\
1546     Name:~\l_stex_module_name_str\\
1547     Language:~\l_stex_module_lang_str\\
1548     Signature:~\l_stex_module_sig_str\\
1549     Metatheory:~\l_stex_module_meta_str\\
1550     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1551   }
1552

```

```

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

```

1572 \cs_new_protected:Nn \_stex_modules_end_module: {
1573   \stex_debug:nn{modules}{Closing module~\prop_item:cn {c_stex_module_\l_stex_current_module}
1574   \stex_reset_up_to_module:n \l_stex_current_module_str
1575   \stex_if_smsmode:T {
1576     \stex_persist:x {
1577       \prop_set_from_keyval:cn{c_stex_module_\l_stex_current_module_str _prop}{
1578         \exp_after:wN \prop_to_keyval:N \csname c_stex_module_\l_stex_current_module_str _pr
1579       }
1580       \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _constants}{
1581         \seq_use:cn{c_stex_module_\l_stex_current_module_str _constants},
1582       }
1583       \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _imports}{
1584         \seq_use:cn{c_stex_module_\l_stex_current_module_str _imports},
1585       }
1586       \tl_set:cn {c_stex_module_\l_stex_current_module_str _code}
1587     }
1588     \exp_after:wN \let \exp_after:wN \l_tmpa_tl \csname c_stex_module_\l_stex_current_module
1589     \exp_after:wN \stex_persist:n \exp_after:wN { \exp_after:wN { \l_tmpa_tl } }
1590   }
1591 }

```

(End definition for `\_stex_modules_end_module:.`)

The core environment

```

1592 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1593 \NewDocumentEnvironment { smodule } { 0 } { m } {
1594   \stex_module_setup:nn{#1}{#2}
1595   %\par
1596   \stex_if_smsmode:F{
1597     \tl_if_empty:NF \smoduletitle {
1598       \exp_args:No \stex_document_title:n \smoduletitle
1599     }

```

```

1600 \tl_clear:N \l_tmpa_tl
1601 \clist_map_inline:Nn \smodulotype {
1602   \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1603     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1604   }
1605 }
1606 \tl_if_empty:NTF \l_tmpa_tl {
1607   \__stex_modules_smodule_start:
1608 }{
1609   \l_tmpa_tl
1610 }
1611 }
1612 \__stex_modules_begin_module:
1613 \str_if_empty:NF \smoduleid {
1614   \stex_ref_new_doc_target:n \smoduleid
1615 }
1616 \stex_smsmode_do:
1617 } {
1618   \__stex_modules_end_module:
1619   \stex_if_smsmode:F {
1620     \end{stex_annotate_env}
1621     \clist_set:Nn \l_tmpa_clist \smodulotype
1622     \tl_clear:N \l_tmpa_tl
1623     \clist_map_inline:Nn \l_tmpa_clist {
1624       \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1625         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1626       }
1627     }
1628     \tl_if_empty:NTF \l_tmpa_tl {
1629       \__stex_modules_smodule_end:
1630     }{
1631       \l_tmpa_tl
1632     }
1633   }
1634 }

```

### **\stexpatchmodule**

```

1635 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1636 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1637
1638 \newcommand\stexpatchmodule[3] [] {
1639   \str_set:Nx \l_tmpa_str{ #1 }
1640   \str_if_empty:NTF \l_tmpa_str {
1641     \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1642     \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1643   }{
1644     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1645     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1646   }
1647 }

```

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

## 26.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n 1648 \NewDocumentCommand \STEXModule { m } {
1649   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1650   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1651   \tl_set:Nn \l_tmpa_tl {
1652     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1653   }
1654   \seq_map_inline:Nn \l_stex_all_modules_seq {
1655     \str_set:Nn \l_tmpb_str { ##1 }
1656     \str_if_eq:eeT { \l_tmpa_str } {
1657       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1658     } {
1659       \seq_map_break:n {
1660         \tl_set:Nn \l_tmpa_tl {
1661           \stex_invoke_module:n { ##1 }
1662         }
1663       }
1664     }
1665   }
1666   \l_tmpa_tl
1667 }
1668
1669 \cs_new_protected:Nn \stex_invoke_module:n {
1670   \stex_debug:nn{modules}{Invoking~module~#1}
1671   \peek_charcode_remove:NTF ! {
1672     \__stex_modules_invoke_uri:nN { #1 }
1673   } {
1674     \peek_charcode_remove:NTF ? {
1675       \__stex_modules_invoke_symbol:nn { #1 }
1676     } {
1677       \msg_error:nnx{stex}{error/syntax}{
1678         ?~or~!~expected~after~
1679         \c_backslash_str STEXModule{#1}
1680       }
1681     }
1682   }
1683 }
1684
1685 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1686   \str_set:Nn #2 { #1 }
1687 }
1688
1689 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1690   \stex_invoke_symbol:n{#1?#2}
1691 }

```

(End definition for `\STEXModule` and `\stex_invoke_module:n`. These functions are documented on page 77.)

```

\stex_activate_module:n
1692 \bool_new:N \l_stex_in_meta_bool
1693 \bool_set_false:N \l_stex_in_meta_bool

```

```

1694 \cs_new_protected:Nn \stex_activate_module:n {
1695   \stex_debug:nn{modules}{Activating~module~#1}
1696   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1697     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1698     \use:c{ c_stex_module_#1_code }
1699   }
1700 }

```

*(End definition for \stex\_activate\_module:n. This function is documented on page 78.)*

```

1701 \</package>

```

## Chapter 27

# STEX -Module Inheritance Implementation

```
1702 ⟨*package⟩
1703
1704 %%%%%%%%% inheritance.dtx %%%%%%%%%
1705
```

### 27.1 SMS Mode

```
1706 ⟨@@=stex_smsmode⟩

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq
1707 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1708 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1709 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1710
1711 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1712   \makeatletter
1713   \makeatother
1714   \ExplSyntaxOn
1715   \ExplSyntaxOff
1716   \rustexBREAK
1717 }
1718
1719 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1720   \symdef
1721   \importmodule
1722   \notation
1723   \symdecl
1724   \STEXexport
1725   \inlineass
1726   \inlinedef
1727   \inlineex
1728   \endinput
1729   \setnotation
```



```

1730 \copynotation
1731 \assign
1732 \renamedekl
1733 \donotcopy
1734 \instantiate
1735 \textsymdecl
1736 }
1737
1738 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1739   \tl_to_str:n {
1740     smodule,
1741     copymodule,
1742     interpretmodule,
1743     realization,
1744     sdefinition,
1745     sexample,
1746     sassertion,
1747     sparagraph,
1748     mathstructure
1749   }
1750 }

```

(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 79.)

`\stex_if_smsmode_p:`

```

\stex_if_smsmode:TF
1751 \bool_new:N \g__stex_smsmode_bool
1752 \bool_set_false:N \g__stex_smsmode_bool
1753 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1754   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1755 }

```

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

`\_stex_smsmode_in_smsmode:nn`

```

1756 \cs_new_protected:Nn \_stex_smsmode_in_smsmode:nn { \stex_suppress_html:n {
1757   \vbox_set:Nn \l_tmpa_box {
1758     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1759     \bool_gset_true:N \g__stex_smsmode_bool
1760     #2
1761     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1762   }
1763   \box_clear:N \l_tmpa_box
1764 } }

```

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

`\stex_file_in_smsmode:nn`

```

1765 \quark_new:N \q__stex_smsmode_break
1766
1767 \NewDocumentCommand \_stex_smsmode_importmodule: { 0{} m } {
1768   \seq_gput_right:Nn \l__stex_smsmode_importmodules_seq {{#1}{#2}}
1769   \stex_smsmode_do:
1770 }
1771

```

```

1772 \cs_new_protected:Nn \__stex_smsmode_module:nn {
1773   \__stex_modules_args:n{#1}
1774   \stex_if_in_module:F {
1775     \str_if_empty:NF \l_stex_module_sig_str {
1776       \stex_modules_current_namespace:
1777       \str_set:Nx \l_stex_module_name_str { #2 }
1778       \stex_if_module_exists:nF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1779         \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1780         \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1781         \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1782         \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1783         \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1784         \str_set:Nx \l_tmpa_str {
1785           \stex_path_to_string:N \l_tmpa_seq /
1786           \l_tmpa_str . \l_stex_module_sig_str .tex
1787         }
1788         \IfFileExists \l_tmpa_str {
1789           \exp_args:NNx \seq_gput_right:Nn \l__stex_smsmode_sigmodules_seq \l_tmpa_str
1790         }{
1791           \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1792         }
1793       }
1794     }
1795   }
1796 }
1797
1798 \prg_new_conditional:Nnn \__stex_smsmode_check_import_pair:nn {T,F,TF} {
1799   %\stex_debug:nn{import-pair}{\detokenize{#1}~{#2}}
1800   \tl_if_empty:nTF{#1}{
1801     \prop_if_exist:NTF \l_stex_current_repository_prop
1802     {
1803       %\stex_debug:nn{import-pair}{in repository \prop_item:Nn \l_stex_current_repository_
1804       \prg_return_true:
1805     } {
1806       \seq_set_split:Nnn \l_tmpa_seq ? {#2}
1807       \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
1808       \tl_if_empty:NT \l_tmpa_tl {
1809         \seq_pop_left:NN \l_tmpa_seq \l_tmpa_tl
1810       }
1811       %\stex_debug:nn{import-pair}{\seq_use:Nn \l_tmpa_seq,~of~length~\seq_count:N \l_tmpa
1812       \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1
1813       \prg_return_true: \prg_return_false:
1814     }
1815   }\prg_return_true:
1816 }
1817
1818 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1819   \stex_filestack_push:n{#1}
1820   \seq_gclear:N \l__stex_smsmode_importmodules_seq
1821   \seq_gclear:N \l__stex_smsmode_sigmodules_seq
1822   % ----- new -----
1823   \__stex_smsmode_in_smsmode:nn{#1}{
1824     \let\importmodule\__stex_smsmode_importmodule:
1825     \let\stex_module_setup:nn\__stex_smsmode_module:nn

```

```

1826 \let\__stex_modules_begin_module:\relax
1827 \let\__stex_modules_end_module:\relax
1828 \seq_clear:N \g_stex_smsmode_allowedenvs_seq
1829 \exp_args:NNx \seq_put_right:Nn \g_stex_smsmode_allowedenvs_seq {\tl_to_str:n{smodule}}
1830 \tl_clear:N \g_stex_smsmode_allowedmacros_tl
1831 \tl_clear:N \g_stex_smsmode_allowedmacros_escape_tl
1832 \tl_put_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\importmodule}
1833 \everyeof{\q__stex_smsmode_break\noexpand}
1834 \expandafter\expandafter\expandafter
1835 \stex_smsmode_do:
1836 \csname @ @ input\endcsname "#1"\relax
1837
1838 \seq_map_inline:Nn \l__stex_smsmode_sigmodules_seq {
1839   \stex_filestack_push:n{##1}
1840   \expandafter\expandafter\expandafter
1841   \stex_smsmode_do:
1842   \csname @ @ input\endcsname "##1"\relax
1843   \stex_filestack_pop:
1844 }
1845 }
1846 % ----- new -----
1847 \__stex_smsmode_in_smsmode:nn{#1} {
1848   #2
1849   % ----- new -----
1850   \begingroup
1851   \%stex_debug:nn{smsmode}{Here:~\seq_use:Nn\l__stex_smsmode_importmodules_seq, }
1852   \seq_map_inline:Nn \l__stex_smsmode_importmodules_seq {
1853     \__stex_smsmode_check_import_pair:nnT ##1 { \begingroup
1854       \stex_import_module_uri:nn ##1
1855       \stex_import_require_module:nnnn
1856       \l_stex_import_ns_str
1857       \l_stex_import_archive_str
1858       \l_stex_import_path_str
1859       \l_stex_import_name_str \endgroup
1860     }
1861   }
1862   \endgroup
1863   \%stex_debug:nn{smsmode}{Actually~loading~file~#1}
1864   % ----- new -----
1865   \everyeof{\q__stex_smsmode_break\noexpand}
1866   \expandafter\expandafter\expandafter
1867   \stex_smsmode_do:
1868   \csname @ @ input\endcsname "#1"\relax
1869 }
1870 \stex_filestack_pop:
1871 }

```

(End definition for `\stex_file_in_smsmode:nn`. This function is documented on page 80.)

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

```

1872 \cs_new_protected:Npn \stex_smsmode_do: {
1873   \stex_if_smsmode:T {
1874     \__stex_smsmode_do:w

```

```

1875 }
1876 }
1877 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1878   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
1879     \expandafter\if\expandafter\relax\noexpand#1
1880     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1881     \else\expandafter\__stex_smsmode_do:w\fi
1882   }{
1883     \__stex_smsmode_do:w % #1
1884   }
1885 }
1886 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1887   \cs_if_eq:NNTF #1 \q__stex_smsmode_break {
1888     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1889       #1\__stex_smsmode_do:w
1890     }{
1891       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1892         #1
1893       }{
1894         \cs_if_eq:NNTF \begin #1 {
1895           \__stex_smsmode_check_begin:n
1896         }{
1897           \cs_if_eq:NNTF \end #1 {
1898             \__stex_smsmode_check_end:n
1899           }{
1900             \__stex_smsmode_do:w
1901           }
1902         }
1903       }
1904     }
1905   }
1906 }
1907
1908 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1909   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1910     \begin{#1}
1911   }{
1912     \__stex_smsmode_do:w
1913   }
1914 }
1915 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1916   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1917     \end{#1}\__stex_smsmode_do:w
1918   }{
1919     \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1920   }
1921 }

```

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

## 27.2 Inheritance

```

1922 <@@=stex_importmodule>

```

`\stex_import_module_uri:nn`

```
1923 \cs_new_protected:Nn \stex_import_module_uri:nn {
1924   \str_set:Nx \l_stex_import_archive_str { #1 }
1925   \str_set:Nn \l_stex_import_path_str { #2 }
1926
1927   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1928   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1929   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1930
1931   \stex_modules_current_namespace:
1932   \bool_lazy_all:nTF {
1933     {\str_if_empty_p:N \l_stex_import_archive_str}
1934     {\str_if_empty_p:N \l_stex_import_path_str}
1935     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1936   }{
1937     \str_set_eq:NN \l_stex_import_path_str \l_stex_module_subpath_str
1938     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1939   }{
1940     \str_if_empty:NT \l_stex_import_archive_str {
1941       \prop_if_exist:NT \l_stex_current_repository_prop {
1942         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1943       }
1944     }
1945     \str_if_empty:NTF \l_stex_import_archive_str {
1946       \str_if_empty:NF \l_stex_import_path_str {
1947         \stex_path_from_string:Nn \l_tmpb_seq {
1948           \l_stex_module_ns_str / .. / \l_stex_import_path_str
1949         }
1950         \str_set:Nx \l_stex_import_ns_str {\stex_path_to_string:N \l_tmpb_seq}
1951         \str_replace_once:Nnn \l_stex_import_ns_str {file://} {file://}
1952       }
1953     }{
1954       \stex_require_repository:n \l_stex_import_archive_str
1955       \prop_get:cnN { c_stex_mathhub_\l_stex_import_archive_str_manifest_prop } { ns }
1956       \l_stex_import_ns_str
1957       \str_if_empty:NF \l_stex_import_path_str {
1958         \str_set:Nx \l_stex_import_ns_str {
1959           \l_stex_import_ns_str / \l_stex_import_path_str
1960         }
1961       }
1962     }
1963   }
1964 }
```

(End definition for `\stex_import_module_uri:nn`. This function is documented on page 81.)

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

Store the return values of `\stex_import_module_uri:nn`.

```
1965 \str_new:N \l_stex_import_name_str
1966 \str_new:N \l_stex_import_archive_str
1967 \str_new:N \l_stex_import_path_str
1968 \str_new:N \l_stex_import_ns_str
```

(End definition for `\l_stex_import_name_str` and others. These variables are documented on page 81.)

```

\stex_import_require_module:nnnnn {\langle ns \rangle} {\langle archive-ID \rangle} {\langle path \rangle} {\langle name \rangle}
1969 \cs_new_protected:Nn \stex_import_require_module:nnnnn {
1970 \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1971
1972 \stex_debug:nn{requiremodule}{Here:\~1:\~2:\~3:\~4}
1973
1974 \exp_args:NNxx \seq_set_split:Nnn \l_tmpa_seq {\tl_to_str:n{/}} {#4}
1975 \seq_get_left:NN \l_tmpa_seq \l_tmpc_str
1976
1977 %\stex_debug:nn{requiremodule}{Top~module:\l_tmpc_str}
1978
1979 % archive
1980 \str_set:Nx \l_tmpa_str { #2 }
1981 \str_if_empty:NTF \l_tmpa_str {
1982 \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1983 \seq_put_right:Nn \l_tmpa_seq {...}
1984 } {
1985 \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1986 \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1987 \seq_put_right:Nn \l_tmpa_seq { source }
1988 }
1989
1990 % path
1991 \str_set:Nx \l_tmpb_str { #3 }
1992 \str_if_empty:NTF \l_tmpb_str {
1993 \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / \l_tmpc_str }
1994
1995 \ltx@ifpackageloaded{babel} {
1996 \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1997 { \language } \l_tmpb_str {
1998 \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1999 }
2000 } {
2001 \str_clear:N \l_tmpb_str
2002 }
2003
2004 \stex_debug:nn{modules}{Checking~a1~\l_tmpa_str.\l_tmpb_str.tex}
2005 \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2006 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2007 }{
2008 \stex_debug:nn{modules}{Checking~a2~\l_tmpa_str.tex}
2009 \IfFileExists{ \l_tmpa_str.tex }{
2010 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2011 }{
2012 % try english as default
2013 \stex_debug:nn{modules}{Checking~a3~\l_tmpa_str.en.tex}
2014 \IfFileExists{ \l_tmpa_str.en.tex }{
2015 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2016 }{
2017 \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2018 }
2019 }
2020 }
2021

```

```

2022 } {
2023   \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
2024   \seq_concat:NNN \l_tmpb_seq \l_tmpa_seq \l_tmpb_seq
2025
2026   \ltx@ifpackageloaded{babel} {
2027     \exp_args:NnX \prop_get:NnNF \c_stex_language_abbrevs_prop
2028       { \language } \l_tmpb_str {
2029       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
2030     }
2031   } {
2032     \str_clear:N \l_tmpb_str
2033   }
2034
2035   \stex_path_canonicalize:N \l_tmpb_seq
2036   \stex_path_to_string:NN \l_tmpb_seq \l_tmpa_str
2037
2038   \stex_debug:nn{modules}{Checking~b1~\l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex}
2039   \IfFileExists{ \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }{
2040     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }
2041   }{
2042     \stex_debug:nn{modules}{Checking~b2~\l_tmpa_str/\l_tmpc_str.tex}
2043     \IfFileExists{ \l_tmpa_str/\l_tmpc_str.tex }{
2044       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.tex }
2045     }{
2046       % try english as default
2047       \stex_debug:nn{modules}{Checking~b3~\l_tmpa_str/\l_tmpc_str.en.tex}
2048       \IfFileExists{ \l_tmpa_str/\l_tmpc_str.en.tex }{
2049         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.en.tex }
2050       }{
2051         \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.\l_tmpb_str.tex}
2052         \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2053           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2054         }{
2055           \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.tex}
2056           \IfFileExists{ \l_tmpa_str.tex }{
2057             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2058           }{
2059             % try english as default
2060             \stex_debug:nn{modules}{Checking~b5~\l_tmpa_str.en.tex}
2061             \IfFileExists{ \l_tmpa_str.en.tex }{
2062               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2063             }{
2064               \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2065             }
2066           }
2067         }
2068       }
2069     }
2070   }
2071 }
2072
2073 \str_if_eq:eeF{\g__stex_importmodule_file_str}{\seq_use:Nn \g_stex_currentfile_seq /}{
2074   \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
2075     \seq_clear:N \l_stex_all_modules_seq

```

```

2076     \str_clear:N \l_stex_current_module_str
2077     \str_set:Nx \l_tmpb_str { #2 }
2078     \str_if_empty:NF \l_tmpb_str {
2079       \stex_set_current_repository:n { #2 }
2080     }
2081     \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
2082   }
2083
2084   \stex_if_module_exists:nF { #1 ? #4 } {
2085     \msg_error:nnx{stex}{error/unknownmodule}{
2086       #1?#4~(in~file~\g__stex_importmodule_file_str)
2087     }
2088   }
2089 }
2090
2091 }
2092 \stex_activate_module:n { #1 ? #4 }
2093 }

```

(End definition for `\stex_import_require_module:nnnn`. This function is documented on page 81.)

## `\importmodule`

```

2094 \NewDocumentCommand \importmodule { 0{} m } {
2095   \stex_import_module_uri:nn { #1 } { #2 }
2096   \stex_debug:nn{modules}{Importing~module:~
2097     \l_stex_import_ns_str ? \l_stex_import_name_str
2098   }
2099   \stex_import_require_module:nnnn
2100   { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2101   { \l_stex_import_path_str } { \l_stex_import_name_str }
2102   \stex_if_smsmode:F {
2103     \stex_annotate_invisible:nnn
2104     {import} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
2105   }
2106   \exp_args:Nx \stex_add_to_current_module:n {
2107     \stex_import_require_module:nnnn
2108     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2109     { \l_stex_import_path_str } { \l_stex_import_name_str }
2110   }
2111   \exp_args:Nx \stex_add_import_to_current_module:n {
2112     \l_stex_import_ns_str ? \l_stex_import_name_str
2113   }
2114   \stex_smsmode_do:
2115   \ignorespacesandpars
2116 }
2117 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

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

## `\usemodule`

```

2118 \NewDocumentCommand \usemodule { 0{} m } {
2119   \stex_if_smsmode:F {
2120     \stex_import_module_uri:nn { #1 } { #2 }
2121     \stex_import_require_module:nnnn
2122     { \l_stex_import_ns_str } { \l_stex_import_archive_str }

```



```

2123 { \l_stex_import_path_str } { \l_stex_import_name_str }
2124 \stex_annotate_invisible:nnn
2125 {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
2126 }
2127 \stex_smsmode_do:
2128 \ignorespacesandpars
2129 }

```

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

```

2130 \cs_new_protected:Nn \stex_csl_to_imports:Nn {
2131   \tl_if_empty:nF{#2}{
2132     \clist_set:Nn \l_tmpa_clist {#2}
2133     \clist_map_inline:Nn \l_tmpa_clist {
2134       \tl_if_head_eq_charcode:nNTF {##1} [{
2135         #1 ##1
2136       }{
2137         #1{##1}
2138       }
2139     }
2140   }
2141 }
2142 \cs_generate_variant:Nn \stex_csl_to_imports:Nn {No}
2143
2144
2145 \endpackage

```

## Chapter 28

# STEX -Symbols Implementation

```
2146 <*package>
2147
2148 %%%%%%%%%% symbols.dtx %%%%%%%%%%
2149
2150 Warnings and error messages
2151 \msg_new:nnn{stex}{error/wrongargs}{
2152   args~value~in~symbol~declaration~for~#1~
2153   needs~to~be~i,~a,~b~or~B,~but~#2~given
2154 }
2155 \msg_new:nnn{stex}{error/unknownsymbol}{
2156   No~symbol~#1~found!
2157 }
2158 \msg_new:nnn{stex}{error/seqlength}{
2159   Expected~#1~arguments;~got~#2!
2160 }
2161 \msg_new:nnn{stex}{error/unknownnotation}{
2162   Unknown~notation~#1~for~#2!
2163 }
```

### 28.1 Symbol Declarations

```
2163 <@@=stex_symdecl>
\stex_all_symbols:n Map over all available symbols
2164 \cs_new_protected:Nn \stex_all_symbols:n {
2165   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
2166   \seq_map_inline:Nn \l_stex_all_modules_seq {
2167     \seq_map_inline:cn{c_stex_module_##1_constants}{
2168       \__stex_symdecl_all_symbols_cs{##1?####1}
2169     }
2170   }
2171 }
```

(End definition for `\stex_all_symbols:n`. This function is documented on page [83](#).)

## **\STEXsymbol**

```
2172 \NewDocumentCommand \STEXsymbol { m } {  
2173   \stex_get_symbol:n { #1 }  
2174   \exp_args:No  
2175   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }  
2176 }
```

(End definition for \STEXsymbol. This function is documented on page 84.)

symdecl arguments:

```
2177 \keys_define:nn { stex / symdecl } {  
2178   name      .str_set_x:N = \l_stex_symdecl_name_str ,  
2179   local     .bool_set:N = \l_stex_symdecl_local_bool ,  
2180   args      .str_set_x:N = \l_stex_symdecl_args_str ,  
2181   type      .tl_set:N = \l_stex_symdecl_type_tl ,  
2182   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,  
2183   align     .str_set:N = \l_stex_symdecl_align_str , % TODO(?)  
2184   gfc       .str_set:N = \l_stex_symdecl_gfc_str , % TODO(?)  
2185   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)  
2186   def       .tl_set:N = \l_stex_symdecl_definiens_tl ,  
2187   reorder   .str_set_x:N = \l_stex_symdecl_reorder_str ,  
2188   assoc     .choices:nn =  
2189     {bin,binl,binr,pre,conj,pwconj}  
2190     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}  
2191 }  
2192  
2193 \bool_new:N \l_stex_symdecl_make_macro_bool  
2194  
2195 \cs_new_protected:Nn \__stex_symdecl_args:n {  
2196   \str_clear:N \l_stex_symdecl_name_str  
2197   \str_clear:N \l_stex_symdecl_args_str  
2198   \str_clear:N \l_stex_symdecl_deprecate_str  
2199   \str_clear:N \l_stex_symdecl_reorder_str  
2200   \str_clear:N \l_stex_symdecl_assoctype_str  
2201   \bool_set_false:N \l_stex_symdecl_local_bool  
2202   \tl_clear:N \l_stex_symdecl_type_tl  
2203   \tl_clear:N \l_stex_symdecl_definiens_tl  
2204  
2205   \keys_set:nn { stex / symdecl } { #1 }  
2206 }
```

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

```
2207  
2208 \NewDocumentCommand \symdecl { s m O{} } {  
2209   \__stex_symdecl_args:n { #3 }  
2210   \IfBooleanTF #1 {  
2211     \bool_set_false:N \l_stex_symdecl_make_macro_bool  
2212   } {  
2213     \bool_set_true:N \l_stex_symdecl_make_macro_bool  
2214   }  
2215   \stex_symdecl_do:n { #2 }  
2216   \stex_smsmode_do:  
2217 }
```

```

2218
2219 \cs_new_protected:Nn \stex_symdecl_do:nn {
2220   \__stex_symdecl_args:n{#1}
2221   \bool_set_false:N \l_stex_symdecl_make_macro_bool
2222   \stex_symdecl_do:n{#2}
2223 }
2224
2225 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

(End definition for \symdecl. This function is documented on page 82.)

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

```

2226 \cs_new_protected:Nn \stex_symdecl_do:n {
2227   \stex_if_in_module:F {
2228     % TODO throw error? some default namespace?
2229   }
2230
2231   \str_if_empty:NT \l_stex_symdecl_name_str {
2232     \str_set:Nx \l_stex_symdecl_name_str { #1 }
2233   }
2234
2235   \prop_if_exist:cT { l_stex_symdecl_
2236     \l_stex_current_module_str ?
2237     \l_stex_symdecl_name_str
2238     _prop
2239   }{
2240     % TODO throw error (beware of circular dependencies)
2241   }
2242
2243   \prop_clear:N \l_tmpa_prop
2244   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
2245   \seq_clear:N \l_tmpa_seq
2246   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
2247   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
2248
2249   \str_if_empty:NT \l_stex_symdecl_deprecate_str {
2250     \str_if_empty:NF \l_stex_module_deprecate_str {
2251       \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
2252     }
2253   }
2254   \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
2255
2256   \exp_args:No \stex_add_constant_to_current_module:n {
2257     \l_stex_symdecl_name_str
2258   }
2259
2260   % arity/args
2261   \int_zero:N \l_tmpb_int
2262
2263   \bool_set_true:N \l_tmpa_bool
2264   \str_map_inline:Nn \l_stex_symdecl_args_str {
2265     \token_case_meaning:NnF ##1 {
2266       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2267       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }

```

```

2268     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2269     {\tl_to_str:n a} {
2270         \bool_set_false:N \l_tmpa_bool
2271         \int_incr:N \l_tmpb_int
2272     }
2273     {\tl_to_str:n B} {
2274         \bool_set_false:N \l_tmpa_bool
2275         \int_incr:N \l_tmpb_int
2276     }
2277 }{
2278     \msg_error:nnxx{stex}{error/wrongargs}{
2279         \l_stex_current_module_str ?
2280         \l_stex_symdecl_name_str
2281     }{##1}
2282 }
2283 }
2284 \bool_if:NTF \l_tmpa_bool {
2285     % possibly numeric
2286     \str_if_empty:NTF \l_stex_symdecl_args_str {
2287         \prop_put:Nnn \l_tmpa_prop { args } {}
2288         \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2289     }{
2290         \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
2291         \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2292         \str_clear:N \l_tmpa_str
2293         \int_step_inline:nn \l_tmpa_int {
2294             \str_put_right:Nn \l_tmpa_str i
2295         }
2296         \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
2297     }
2298 } {
2299     \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2300     \prop_put:Nnx \l_tmpa_prop { arity }
2301     { \str_count:N \l_stex_symdecl_args_str }
2302 }
2303 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2304
2305 \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
2306     \prop_put:Nnx \l_tmpa_prop { defined }{ false }
2307 }{
2308     \prop_put:Nnx \l_tmpa_prop { defined }{ true }
2309 }
2310
2311 % semantic macro
2312
2313 \bool_if:NT \l_stex_symdecl_make_macro_bool {
2314     \exp_args:Nx \stex_do_up_to_module:n {
2315         \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2316             \l_stex_current_module_str ? \l_stex_symdecl_name_str
2317         }}
2318     }
2319 }
2320
2321 \stex_debug:nn{symbols}{New~symbol:~

```

```

2322 \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2323 Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2324 Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
2325 Definiens:~\exp_not:o {\l_stex_symdecl_definiens_tl}
2326 }
2327
2328 % circular dependencies require this:
2329 \stex_if_do_html:T {
2330   \stex_annotate_invisible:nnn {symdecl} {
2331     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2332   } {
2333     \tl_if_empty:NF \l_stex_symdecl_type_tl {
2334       \stex_annotate_invisible:nnn{type}{\l_stex_symdecl_type_tl$}
2335     }
2336     \stex_annotate_invisible:nnn{args}{\prop_item:Nn \l_tmpa_prop { args }}{ }
2337     \stex_annotate_invisible:nnn{macroname}{#1}{ }
2338     \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2339       \stex_annotate_invisible:nnn{definiens}{ }
2340       {\l_stex_symdecl_definiens_tl$}
2341     }
2342     \str_if_empty:NF \l_stex_symdecl_assoc_type_str {
2343       \stex_annotate_invisible:nnn{assoc_type}{\l_stex_symdecl_assoc_type_str}{ }
2344     }
2345     \str_if_empty:NF \l_stex_symdecl_reorder_str {
2346       \stex_annotate_invisible:nnn{reorderargs}{\l_stex_symdecl_reorder_str}{ }
2347     }
2348   }
2349 }
2350 \prop_if_exist:cF {
2351   \l_stex_symdecl_
2352   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2353   _prop
2354 } {
2355   \bool_if:NTF \l_stex_symdecl_local_bool \stex_do_up_to_module:x \stex_execute_in_module:
2356   \__stex_symdecl_restore_symbol:nnnnnnn
2357   {\l_stex_symdecl_name_str}
2358   { \prop_item:Nn \l_tmpa_prop {args} }
2359   { \prop_item:Nn \l_tmpa_prop {arity} }
2360   { \prop_item:Nn \l_tmpa_prop {assoc} }
2361   { \prop_item:Nn \l_tmpa_prop {defined} }
2362   {\bool_if:NT \l_stex_symdecl_make_macro_bool {#1} }
2363   {\l_stex_current_module_str}
2364 }
2365 }
2366 }
2367 \cs_new_protected:Nn \__stex_symdecl_restore_symbol:nnnnnnn {
2368   \prop_clear:N \l_tmpa_prop
2369   \prop_put:Nnn \l_tmpa_prop { module } { #7 }
2370   \prop_put:Nnn \l_tmpa_prop { name } { #1 }
2371   \prop_put:Nnn \l_tmpa_prop { args } { #2 }
2372   \prop_put:Nnn \l_tmpa_prop { arity } { #3 }
2373   \prop_put:Nnn \l_tmpa_prop { assoc } { #4 }
2374   \prop_put:Nnn \l_tmpa_prop { defined } { #5 }
2375   \tl_if_empty:nF{#6}{

```

```

2376     \tl_set:cx{#6}{\stex_invoke_symbol:n{\detokenize{#7 ? #1}}}
2377   }
2378   \prop_set_eq:cN{l_stex_symdecl_ \detokenize{#7 ? #1} _prop}\l_tmpa_prop
2379   \seq_clear:c{l_stex_symdecl_ \detokenize{#7 ? #1} _notations}
2380 }

```

(End definition for `\stex_symdecl_do:n`. This function is documented on page 83.)

## `\textsymdecl`

```

2381
2382 \keys_define:nn { stex / textsymdecl } {
2383   name      .str_set_x:N = \l__stex_symdecl_name_str ,
2384   type      .tl_set:N    = \l__stex_symdecl_type_tl
2385 }
2386
2387 \cs_new_protected:Nn \_stex_textsymdecl_args:n {
2388   \str_clear:N \l__stex_symdecl_name_str
2389   \tl_clear:N \l__stex_symdecl_type_tl
2390   \keys_set:nn { stex / textsymdecl } { #1 }
2391 }
2392
2393 \NewDocumentCommand \textsymdecl {m O{} m} {
2394   \_stex_textsymdecl_args:n { #2 }
2395   \str_if_empty:NTF \l__stex_symdecl_name_str {
2396     \__stex_symdecl_args:n{name=#1,#2}
2397   }{
2398     \__stex_symdecl_args:n{#2}
2399   }
2400   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2401   \stex_symdecl_do:n{#1-sym}
2402   \stex_execute_in_module:n{
2403     \cs_set_nopar:cpn{#1name}{
2404       \ifvmode\hbox_unpack:N\c_empty_box\fi
2405       \ifmmode\hbox{#3}\else#3\fi\xspace
2406     }
2407     \cs_set_nopar:cpn{#1}{
2408       \ifmmode\csname#1-sym\expandafter\endcsname\else
2409       \ifvmode\hbox_unpack:N\c_empty_box\fi
2410       \symref{#1-sym}{#3}\expandafter\xspace
2411       \fi
2412     }
2413   }
2414   \stex_execute_in_module:x{
2415     \__stex_notation_restore_notation:nnnnn
2416     {\l_stex_current_module_str?\tl_if_empty:NTF\l__stex_symdecl_name_str{#1}\l__stex_symdecl
2417     }{0}
2418     {\exp_not:n{\STEXInternalTermMathOMSiiii{\STEXInternalCurrentSymbolStr}{}}{\neginfprec}{
2419       \comp{\hbox{#3}}\STEXInternalSymbolAfterInvokationTL
2420     }}
2421     {}
2422   }
2423   \stex_smsmode_do:
2424 }

```

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

`\stex_get_symbol:n`

```
2425 \str_new:N \l_stex_get_symbol_uri_str
2426
2427 \cs_new_protected:Nn \stex_get_symbol:n {
2428   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2429     \tl_set:Nn \l_tmpa_tl { #1 }
2430     \__stex_symdecl_get_symbol_from_cs:
2431   }{
2432     % argument is a string
2433     % is it a command name?
2434     \cs_if_exist:cTF { #1 }{
2435       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2436       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2437       \str_if_empty:NTF \l_tmpa_str {
2438         \exp_args:Nx \cs_if_eq:NNTF {
2439           \tl_head:N \l_tmpa_tl
2440         } \stex_invoke_symbol:n {
2441           \__stex_symdecl_get_symbol_from_cs:
2442         }{
2443           \__stex_symdecl_get_symbol_from_string:n { #1 }
2444         }
2445       } {
2446         \__stex_symdecl_get_symbol_from_string:n { #1 }
2447       }
2448     }{
2449       % argument is not a command name
2450       \__stex_symdecl_get_symbol_from_string:n { #1 }
2451       % \l_stex_all_symbols_seq
2452     }
2453   }
2454   \str_if_eq:eeF {
2455     \prop_item:cn {
2456       l_stex_symdecl\l_stex_get_symbol_uri_str _prop
2457     }{ deprecate }
2458   }{}{
2459     \msg_warning:nxxx{stex}{warning/deprecated}{
2460       Symbol~\l_stex_get_symbol_uri_str
2461     }{
2462       \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{ deprecate }
2463     }
2464   }
2465 }
2466
2467 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2468   \tl_set:Nn \l_tmpa_tl {
2469     \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2470   }
2471   \str_set:Nn \l_tmpa_str { #1 }
2472
2473   %\int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2474
2475   \str_if_in:NnTF \l_tmpa_str ? {
2476     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
2477     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
```



```

2478 \str_set:Nx \l_tmpb_str {\seq_use:Nn \l_tmpa_seq ?}
2479 }{
2480 \str_clear:N \l_tmpb_str
2481 }
2482 \str_if_empty:NTF \l_tmpb_str {
2483 \seq_map_inline:Nn \l_stex_all_modules_seq {
2484 \seq_map_inline:cn{c_stex_module_###1_constants}{
2485 \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2486 \seq_map_break:n{\seq_map_break:n{
2487 \tl_set:Nn \l_tmpa_tl {
2488 \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2489 }
2490 }}
2491 }
2492 }
2493 }
2494 }{
2495 \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpb_str }
2496 \seq_map_inline:Nn \l_stex_all_modules_seq {
2497 \str_if_eq:eeT{ \l_tmpb_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2498 \seq_map_inline:cn{c_stex_module_###1_constants}{
2499 \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2500 \seq_map_break:n{\seq_map_break:n{
2501 \tl_set:Nn \l_tmpa_tl {
2502 \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2503 }
2504 }}
2505 }
2506 }
2507 }
2508 }
2509 }
2510
2511 \l_tmpa_tl
2512 }
2513
2514 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
2515 \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2516 { \tl_tail:N \l_tmpa_tl }
2517 \tl_if_single:NTF \l_tmpa_tl {
2518 \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2519 \exp_after:wN \str_set:Nn \exp_after:wN
2520 \l_stex_get_symbol_uri_str \l_tmpa_tl
2521 }{
2522 % TODO
2523 % tail is not a single group
2524 }
2525 }{
2526 % TODO
2527 % tail is not a single group
2528 }
2529 }

```

(End definition for `\stex_get_symbol:n`. This function is documented on page 83.)

## 28.2 Notations

```

2530 <@@=stex_notation>

      notation arguments:
2531 \keys_define:nn { stex / notation } {
2532 % lang .tl_set_x:N = \l__stex_notation_lang_str ,
2533 variant .tl_set_x:N = \l__stex_notation_variant_str ,
2534 prec .str_set_x:N = \l__stex_notation_prec_str ,
2535 op .tl_set:N = \l__stex_notation_op_tl ,
2536 primary .bool_set:N = \l__stex_notation_primary_bool ,
2537 primary .default:n = {true} ,
2538 unknown .code:n = \str_set:Nx
2539     \l__stex_notation_variant_str \l_keys_key_str
2540 }
2541
2542 \cs_new_protected:Nn \stex_notation_args:n {
2543 % \str_clear:N \l__stex_notation_lang_str
2544 \str_clear:N \l__stex_notation_variant_str
2545 \str_clear:N \l__stex_notation_prec_str
2546 \tl_clear:N \l__stex_notation_op_tl
2547 \bool_set_false:N \l__stex_notation_primary_bool
2548
2549 \keys_set:nn { stex / notation } { #1 }
2550 }

\notation

2551 \NewDocumentCommand \notation { s m O{}} {
2552   \stex_notation_args:n { #3 }
2553   \tl_clear:N \l_stex_symdecl_definiens_tl
2554   \stex_get_symbol:n { #2 }
2555   \tl_set:Nn \l_stex_notation_after_do_tl {
2556     \__stex_notation_final:
2557     \IfBooleanTF#1{
2558       \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2559     }{}
2560     \stex_smsmode_do:\ignorespacesandpars
2561   }
2562   \stex_notation_do:nnnnn
2563   { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str_prop} { args } }
2564   { \prop_item:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str_prop } { arity } }
2565   { \l__stex_notation_variant_str }
2566   { \l__stex_notation_prec_str }
2567 }
2568 \stex_deactivate_macro:Nn \notation {module-environments}

(End definition for \notation. This function is documented on page 83.)

\stex_notation_do:nnnnn

2569 \seq_new:N \l__stex_notation_precedences_seq
2570 \tl_new:N \l__stex_notation_opprec_tl
2571 \int_new:N \l__stex_notation_currarg_int
2572 \tl_new:N \STEXInternalSymbolAfterInvokationTL
2573
2574 \cs_new_protected:Nn \stex_notation_do:nnnnn {

```

```

2575 \let\STEXInternalCurrentSymbolStr\relax
2576 \seq_clear:N \l__stex_notation_precedences_seq
2577 \tl_clear:N \l__stex_notation_opprec_tl
2578 \str_set:Nx \l__stex_notation_args_str { #1 }
2579 \str_set:Nx \l__stex_notation_arity_str { #2 }
2580 \str_set:Nx \l__stex_notation_suffix_str { #3 }
2581 \str_set:Nx \l__stex_notation_prec_str { #4 }
2582
2583 % precedences
2584 \str_if_empty:NTF \l__stex_notation_prec_str {
2585   \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2586     \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2587   }{
2588     \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2589   }
2590 } {
2591   \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2592     \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2593     \int_step_inline:nn { \l__stex_notation_arity_str } {
2594       \exp_args:NNo
2595       \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2596     }
2597   }{
2598     \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2599     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2600       \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2601       \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2602         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2603           \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2604         \seq_map_inline:Nn \l_tmpa_seq {
2605           \seq_put_right:Nn \l__stex_notation_precedences_seq { ##1 }
2606         }
2607       }
2608     }{
2609       \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2610         \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2611       }{
2612         \tl_set:No \l__stex_notation_opprec_tl { 0 }
2613       }
2614     }
2615   }
2616 }
2617
2618 \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2619 \int_step_inline:nn { \l__stex_notation_arity_str } {
2620   \seq_pop_left:NNTF \l_tmpa_seq \l_tmpb_str {
2621     \exp_args:NNo
2622     \seq_put_right:No \l__stex_notation_precedences_seq {
2623       \l__stex_notation_opprec_tl
2624     }
2625   }
2626 }
2627 \tl_clear:N \l__stex_notation_dummyargs_tl
2628

```

```

2629 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2630   \exp_args:NNe
2631   \cs_set:Npn \l_stex_notation_macrocode_cs {
2632     \STEXInternalTermMathOMSiiii { \STEXInternalCurrentSymbolStr }
2633     { \l__stex_notation_suffix_str }
2634     { \l__stex_notation_opprec_tl }
2635     { \exp_not:n { #5 } }
2636   }
2637   \l_stex_notation_after_do_tl
2638 }{
2639   \str_if_in:NnTF \l__stex_notation_args_str b {
2640     \exp_args:Nne \use:nn
2641     {
2642       \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2643       \cs_set:Npn \l__stex_notation_arity_str } { {
2644         \STEXInternalTermMathOMBiiii { \STEXInternalCurrentSymbolStr }
2645         { \l__stex_notation_suffix_str }
2646         { \l__stex_notation_opprec_tl }
2647         { \exp_not:n { #5 } }
2648       } }
2649   }{
2650     \str_if_in:NnTF \l__stex_notation_args_str B {
2651       \exp_args:Nne \use:nn
2652       {
2653         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2654         \cs_set:Npn \l__stex_notation_arity_str } { {
2655           \STEXInternalTermMathOMBiiii { \STEXInternalCurrentSymbolStr }
2656           { \l__stex_notation_suffix_str }
2657           { \l__stex_notation_opprec_tl }
2658           { \exp_not:n { #5 } }
2659         } }
2660     }{
2661       \exp_args:Nne \use:nn
2662       {
2663         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2664         \cs_set:Npn \l__stex_notation_arity_str } { {
2665           \STEXInternalTermMathOMAiinii { \STEXInternalCurrentSymbolStr }
2666           { \l__stex_notation_suffix_str }
2667           { \l__stex_notation_opprec_tl }
2668           { \exp_not:n { #5 } }
2669         } }
2670     }
2671   }
2672
2673   \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2674   \int_zero:N \l__stex_notation_currarg_int
2675   \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2676   \__stex_notation_arguments:
2677 }
2678 }

```

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

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

```

2679 \cs_new_protected:Nn \__stex_notation_arguments: {
2680   \int_incr:N \l__stex_notation_currarg_int
2681   \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2682     \l_stex_notation_after_do_tl
2683   }{
2684     \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2685     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2686     \str_if_eq:VnTF \l_tmpa_str a {
2687       \__stex_notation_argument_assoc:nn{a}
2688     }{
2689       \str_if_eq:VnTF \l_tmpa_str B {
2690         \__stex_notation_argument_assoc:nn{B}
2691       }{
2692         \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpb_str
2693         \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2694           { \STEXInternalTermMathArgiii
2695             { \l_tmpa_str\int_use:N \l__stex_notation_currarg_int }
2696             { \l_tmpb_str }
2697             { ####\int_use:N \l__stex_notation_currarg_int }
2698           }
2699         }
2700         \__stex_notation_arguments:
2701       }
2702     }
2703   }
2704 }

```

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

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

```

2705 \cs_new_protected:Nn \__stex_notation_argument_assoc:nn {
2706
2707   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2708     {\l__stex_notation_arity_str}{
2709     #2
2710   }
2711   \int_zero:N \l_tmpa_int
2712   \tl_clear:N \l_tmpa_tl
2713   \str_map_inline:Nn \l__stex_notation_args_str {
2714     \int_incr:N \l_tmpa_int
2715     \tl_put_right:Nx \l_tmpa_tl {
2716       \str_if_eq:nnTF {##1}{a}{ {} }{ }{
2717         \str_if_eq:nnTF {##1}{B}{ {} }{ }{
2718           {\_stex_term_arg:nn{##1}\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa
2719         }
2720       }
2721     }
2722   }
2723   \exp_after:wN\exp_after:wN\exp_after:wN \def
2724   \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2725   \exp_after:wN\exp_after:wN\exp_after:wN ##
2726   \exp_after:wN\exp_after:wN\exp_after:wN 1
2727   \exp_after:wN\exp_after:wN\exp_after:wN ##
2728   \exp_after:wN\exp_after:wN\exp_after:wN 2

```

```

2729 \exp_after:wN\exp_after:wN\exp_after:wN {
2730   \exp_after:wN \exp_after:wN \exp_after:wN
2731   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2732     \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2733   }
2734 }
2735
2736 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2737 \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
2738   \STEXInternalTermMathAssocArgiiii
2739   { #1\int_use:N \l__stex_notation_currarg_int }
2740   { \l_tmpa_str }
2741   { ####\int_use:N \l__stex_notation_currarg_int }
2742   { \l_tmpa_cs {####1} {####2} }
2743 } }
2744 \__stex_notation_arguments:
2745 }

```

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

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

```

2746 \cs_new_protected:Nn \__stex_notation_restore_notation:nnnnn {
2747   \cs_generate_from_arg_count:cNnn{stex_notation_\detokenize{#1} \c_hash_str \detokenize{#2}}
2748   \cs_set_nopar:Npn {#3}{#4}
2749   \tl_if_empty:nF {#5}{
2750     \tl_set:cn{stex_op_notation_\detokenize{#1} \c_hash_str \detokenize{#2}_cs}{\comp{ #5 }
2751   }
2752   \seq_if_exist:cT { l_stex_symdecl_\detokenize{#1} _notations }{
2753     \seq_put_right:cx { l_stex_symdecl_\detokenize{#1} _notations } { \detokenize{#2} }
2754   }
2755 }
2756
2757 \cs_new_protected:Nn \__stex_notation_final: {
2758
2759   \stex_execute_in_module:x {
2760     \__stex_notation_restore_notation:nnnnn
2761     {\l_stex_get_symbol_uri_str}
2762     {\l__stex_notation_suffix_str}
2763     {\l__stex_notation_arity_str}
2764     {
2765       \exp_after:wN \exp_after:wN \exp_after:wN
2766       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2767       { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \STEXInt
2768     }
2769     {\exp_args:No \exp_not:n \l__stex_notation_op_tl }
2770   }
2771
2772   \stex_debug:nn{symbols}{
2773     Notation~\l__stex_notation_suffix_str
2774     ~for~\l_stex_get_symbol_uri_str^^J
2775     Operator~precedence:~\l__stex_notation_opprec_tl^^J
2776     Argument~precedences:~
2777     \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2778     Notation: \cs_meaning:c {

```

```

2779     stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2780     \l__stex_notation_suffix_str
2781     _cs
2782   }
2783 }
2784 % HTML annotations
2785 \stex_if_do_html:T {
2786   \stex_annotate_invisible:nnn { notation }
2787   { \l_stex_get_symbol_uri_str } {
2788     \stex_annotate_invisible:nnn { notationfragment }
2789     { \l__stex_notation_suffix_str }{}
2790   \stex_annotate_invisible:nnn { precedence }
2791     { \l__stex_notation_prec_str }{}
2792
2793   \int_zero:N \l_tmpa_int
2794   \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2795   \tl_clear:N \l_tmpa_tl
2796   \int_step_inline:nn { \l__stex_notation_arity_str }{
2797     \int_incr:N \l_tmpa_int
2798     \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2799     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_rema
2800     \str_if_eq:VnTF \l_tmpb_str a {
2801       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2802         \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2803         \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2804       } }
2805     }{
2806       \str_if_eq:VnTF \l_tmpb_str B {
2807         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2808           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2809           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2810         } }
2811       }{
2812         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2813           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
2814         } }
2815       }
2816     }
2817   }
2818   \stex_annotate_invisible:nnn { notationcomp }{}{
2819     \str_set:Nx \STEXInternalCurrentSymbolStr {\l_stex_get_symbol_uri_str }
2820     $ \exp_args:Nno \use:nn { \use:c {
2821       stex_notation_ \STEXInternalCurrentSymbolStr
2822       \c_hash_str \l__stex_notation_suffix_str _cs
2823     } } { \l_tmpa_tl } $
2824   }
2825   \tl_if_empty:NF \l__stex_notation_op_tl {
2826     \stex_annotate_invisible:nnn { notationopcomp }{}{
2827       $\l__stex_notation_op_tl$
2828     }
2829   }
2830 }
2831 }
2832 }

```

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

**\setnotation**

```

2833 \keys_define:nn { stex / setnotation } {
2834   % lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2835   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2836   unknown .code:n      = \str_set:Nx
2837     \l__stex_notation_variant_str \l_keys_key_str
2838 }
2839
2840 \cs_new_protected:Nn \stex_setnotation_args:n {
2841   % \str_clear:N \l__stex_notation_lang_str
2842   \str_clear:N \l__stex_notation_variant_str
2843   \keys_set:nn { stex / setnotation } { #1 }
2844 }
2845
2846 \cs_new_protected:Nn \_stex_notation_setnotation:nn {
2847   \seq_if_exist:cT{l_stex_symdecl_#1_notations}{
2848     \seq_remove_all:cn { l_stex_symdecl_#1_notations }{ #2 }
2849     \seq_put_left:cn { l_stex_symdecl_#1_notations }{ #2 }
2850   }
2851 }
2852
2853 \cs_new_protected:Nn \stex_setnotation:n {
2854   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1_notations }
2855     { \l__stex_notation_variant_str }{
2856     \stex_execute_in_module:x{ \_stex_notation_setnotation:nn {#1}{\l__stex_notation_vari
2857     \stex_debug:nn {notations}{
2858       Setting~default~notation~
2859       {\l__stex_notation_variant_str }~for~
2860       #1 \\
2861       \expandafter\meaning\csname
2862       l_stex_symdecl_#1_notations\endcsname
2863     }
2864   }{
2865     \msg_error:nxxx{stex}{unknownnotation}{\l__stex_notation_variant_str}{#1}
2866   }
2867 }
2868
2869 \NewDocumentCommand \setnotation {m m} {
2870   \stex_get_symbol:n { #1 }
2871   \_stex_setnotation_args:n { #2 }
2872   \stex_setnotation:n{\l_stex_get_symbol_uri_str}
2873   \stex_smsmode_do:\ignorespacesandpars
2874 }
2875
2876 \cs_new_protected:Nn \stex_copy_notations:nn {
2877   \stex_debug:nn {notations}{
2878     Copying~notations~from~#2~to~#1\\
2879     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
2880   }
2881   \tl_clear:N \l_tmpa_tl
2882   \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2883     \tl_put_right:Nn \l_tmpa_tl { {##### #1} }

```



```

2884 }
2885 \seq_map_inline:cn {l_stex_symdecl_#2_notations}{\begingroup
2886   \stex_debug:nn{Here}{Here:~##1}
2887   \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2888   \edef \l_tmpa_tl {
2889     \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2890     \exp_after:wN\exp_after:wN\exp_after:wN {
2891       \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2892     }
2893   }
2894
2895   \exp_after:wN \def \exp_after:wN \l_tmpa_tl
2896   \exp_after:wN #####\exp_after:wN 1 \exp_after:wN #####\exp_after:wN 2
2897   \exp_after:wN { \l_tmpa_tl }
2898
2899   \edef \l_tmpa_tl {
2900     \exp_after:wN \exp_not:n \exp_after:wN {
2901       \l_tmpa_tl {##### 1}{##### 2}
2902     }
2903   }
2904
2905   \stex_debug:nn{Here}{Here:~\expandafter\detokenize\expandafter{\l_tmpa_tl}}
2906
2907   \stex_execute_in_module:x {
2908     \__stex_notation_restore_notation:nnnnn
2909     {#1}{##1}
2910     { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }
2911     { \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl} }
2912     {
2913       \cs_if_exist:cT{stex_op_notation_ #2\c_hash_str ##1 _cs}{
2914         \exp_args:NNo\exp_args:No\exp_not:n{\csname stex_op_notation_ #2\c_hash_str ##1
2915       }
2916     }
2917   }\endgroup
2918 }
2919 }
2920
2921 \NewDocumentCommand \copynotation {m m} {
2922   \stex_get_symbol:n { #1 }
2923   \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2924   \stex_get_symbol:n { #2 }
2925   \exp_args:Noo
2926   \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2927   \stex_smsmode_do:\ignorespacesandpars
2928 }
2929

```

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

## \symdef

```

2930 \keys_define:nn { stex / symdef } {
2931   name .str_set_x:N = \l_stex_symdecl_name_str ,
2932   local .bool_set:N = \l_stex_symdecl_local_bool ,
2933   args .str_set_x:N = \l_stex_symdecl_args_str ,

```

```

2934 type .tl_set:N = \l_stex_symdecl_type_tl ,
2935 def .tl_set:N = \l_stex_symdecl_definiens_tl ,
2936 reorder .str_set_x:N = \l_stex_symdecl_reorder_str ,
2937 op .tl_set:N = \l__stex_notation_op_tl ,
2938 % lang .str_set_x:N = \l__stex_notation_lang_str ,
2939 variant .str_set_x:N = \l__stex_notation_variant_str ,
2940 prec .str_set_x:N = \l__stex_notation_prec_str ,
2941 assoc .choices:nn =
2942 {bin,binl,binr,pre,conj,pwconj}
2943 {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
2944 unknown .code:n = \str_set:Nx
2945 \l__stex_notation_variant_str \l_keys_key_str
2946 }
2947
2948 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2949 \str_clear:N \l_stex_symdecl_name_str
2950 \str_clear:N \l_stex_symdecl_args_str
2951 \str_clear:N \l_stex_symdecl_assoctype_str
2952 \str_clear:N \l_stex_symdecl_reorder_str
2953 \bool_set_false:N \l_stex_symdecl_local_bool
2954 \tl_clear:N \l_stex_symdecl_type_tl
2955 \tl_clear:N \l_stex_symdecl_definiens_tl
2956 % \str_clear:N \l__stex_notation_lang_str
2957 \str_clear:N \l__stex_notation_variant_str
2958 \str_clear:N \l__stex_notation_prec_str
2959 \tl_clear:N \l__stex_notation_op_tl
2960
2961 \keys_set:nn { stex / symdef } { #1 }
2962 }
2963
2964 \NewDocumentCommand \symdef { m O{} } {
2965 \__stex_notation_symdef_args:n { #2 }
2966 \bool_set_true:N \l_stex_symdecl_make_macro_bool
2967 \stex_symdecl_do:n { #1 }
2968 \tl_set:Nn \l_stex_notation_after_do_tl {
2969 \__stex_notation_final:
2970 \stex_smsmode_do:\ignorespacesandpars
2971 }
2972 \str_set:Nx \l_stex_get_symbol_uri_str {
2973 \l_stex_current_module_str ? \l_stex_symdecl_name_str
2974 }
2975 \exp_args:Nx \stex_notation_do:nnnnn
2976 { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } }
2977 { \prop_item:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } }
2978 { \l__stex_notation_variant_str }
2979 { \l__stex_notation_prec_str }
2980 }
2981 \stex_deactivate_macro:Nn \symdef {module~environments}

```

(End definition for `\symdef`. This function is documented on page 83.)

## 28.3 Variables

```

2982 <@@=stex_variables>

```

```

2983
2984 \keys_define:nn { stex / vardef } {
2985   name      .str_set_x:N = \l__stex_variables_name_str ,
2986   args      .str_set_x:N = \l__stex_variables_args_str ,
2987   type      .tl_set:N    = \l__stex_variables_type_tl ,
2988   def       .tl_set:N    = \l__stex_variables_def_tl ,
2989   op        .tl_set:N    = \l__stex_variables_op_tl ,
2990   prec      .str_set_x:N = \l__stex_variables_prec_str ,
2991   reorder   .str_set_x:N = \l__stex_variables_reorder_str ,
2992   assoc     .choices:nn  =
2993     {bin,binl,binr,pre,conj,pwconj}
2994     {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
2995   bind      .choices:nn  =
2996     {forall,exists}
2997     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2998 }
2999
3000 \cs_new_protected:Nn \__stex_variables_args:n {
3001   \str_clear:N \l__stex_variables_name_str
3002   \str_clear:N \l__stex_variables_args_str
3003   \str_clear:N \l__stex_variables_prec_str
3004   \str_clear:N \l__stex_variables_assoctype_str
3005   \str_clear:N \l__stex_variables_reorder_str
3006   \str_clear:N \l__stex_variables_bind_str
3007   \tl_clear:N \l__stex_variables_type_tl
3008   \tl_clear:N \l__stex_variables_def_tl
3009   \tl_clear:N \l__stex_variables_op_tl
3010
3011   \keys_set:nn { stex / vardef } { #1 }
3012 }
3013
3014 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{}} {
3015   \__stex_variables_args:n {#2}
3016   \str_if_empty:NT \l__stex_variables_name_str {
3017     \str_set:Nx \l__stex_variables_name_str { #1 }
3018   }
3019   \prop_clear:N \l_tmpa_prop
3020   \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
3021
3022   \int_zero:N \l_tmpb_int
3023   \bool_set_true:N \l_tmpa_bool
3024   \str_map_inline:Nn \l__stex_variables_args_str {
3025     \token_case_meaning:NnF ##1 {
3026       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
3027       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
3028       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
3029       {\tl_to_str:n a} {
3030         \bool_set_false:N \l_tmpa_bool
3031         \int_incr:N \l_tmpb_int
3032       }
3033       {\tl_to_str:n B} {
3034         \bool_set_false:N \l_tmpa_bool
3035         \int_incr:N \l_tmpb_int
3036       }

```

```

3037   }{
3038     \msg_error:nnxx{stex}{error/wrongargs}{
3039       variable~\l__stex_variables_name_str
3040     }{##1}
3041   }
3042 }
3043 \bool_if:NTF \l_tmpa_bool {
3044   % possibly numeric
3045   \str_if_empty:NTF \l__stex_variables_args_str {
3046     \prop_put:Nnn \l_tmpa_prop { args } {}
3047     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
3048   }{
3049     \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
3050     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
3051     \str_clear:N \l_tmpa_str
3052     \int_step_inline:nn \l_tmpa_int {
3053       \str_put_right:Nn \l_tmpa_str i
3054     }
3055     \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
3056     \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3057   }
3058 } {
3059   \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3060   \prop_put:Nnx \l_tmpa_prop { arity }
3061   { \str_count:N \l__stex_variables_args_str }
3062 }
3063 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
3064 \tl_set:cx { #1 }{ \stex_invoke_variable:n { \l__stex_variables_name_str } }
3065
3066 \prop_set_eq:cN { l_stex_variable_\l__stex_variables_name_str _prop } \l_tmpa_prop
3067
3068 \tl_if_empty:NF \l__stex_variables_op_tl {
3069   \cs_set:cpx {
3070     stex_var_op_notation_\l__stex_variables_name_str _cs
3071   } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
3072 }
3073
3074 \tl_set:Nn \l_stex_notation_after_do_tl {
3075   \exp_args:Nne \use:nn {
3076     \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str _cs }
3077     \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
3078   } {{
3079     \exp_after:wN \exp_after:wN \exp_after:wN
3080     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
3081     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \STEXInter
3082   }}
3083 \stex_if_do_html:T {
3084   \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
3085     \stex_annotate_invisible:nnn { precedence }
3086     { \l__stex_variables_prec_str }{}
3087     \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{}}{ $\l
3088     \stex_annotate_invisible:nnn{args}{ \l__stex_variables_args_str }{}
3089     \stex_annotate_invisible:nnn{macroname}{#1}{}
3090     \tl_if_empty:NF \l__stex_variables_def_tl {

```

```

3091     \stex_annotate_invisible:nnn{definiens}{}
3092     {${\l__stex_variables_def_tl$}
3093   }
3094   \str_if_empty:NF \l__stex_variables_assoctype_str {
3095     \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{}
3096   }
3097   \str_if_empty:NF \l__stex_variables_reorder_str {
3098     \stex_annotate_invisible:nnn{reorderargs}{\l__stex_variables_reorder_str}{}
3099   }
3100   \int_zero:N \l_tmpa_int
3101   \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
3102   \tl_clear:N \l_tmpa_tl
3103   \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
3104     \int_incr:N \l_tmpa_int
3105     \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
3106     \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables
3107     \str_if_eq:VnTF \l_tmpb_str a {
3108       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3109         \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3110         \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3111       } }
3112     }{
3113       \str_if_eq:VnTF \l_tmpb_str B {
3114         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3115           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3116           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3117         } }
3118       }{
3119         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3120           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
3121         } }
3122       }
3123     }
3124   }
3125   \stex_annotate_invisible:nnn { notationcomp }{}{
3126     \str_set:Nx \STEXInternalCurrentSymbolStr {var://\l__stex_variables_name_str }
3127     $ \exp_args:Nno \use:nn { \use:c {
3128       stex_var_notation_\l__stex_variables_name_str _cs
3129     } } { \l_tmpa_tl } $
3130   }
3131   \tl_if_empty:NF \l__stex_variables_op_tl {
3132     \stex_annotate_invisible:nnn { notationopcomp }{}{
3133       ${\l__stex_variables_op_tl$
3134     }
3135   }
3136 }
3137 \str_if_empty:NF \l__stex_variables_bind_str {
3138   \stex_annotate_invisible:nnn {bindtype}{\l__stex_variables_bind_str,\l__stex_variabl
3139 }
3140 }\ignorespacesandpars
3141 }
3142
3143 \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
3144 }

```

```

3145
3146 \cs_new:Nn \_stex_reset:N {
3147   \tl_if_exist:NTF #1 {
3148     \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
3149   }{
3150     \let \exp_not:N #1 \exp_not:N \undefined
3151   }
3152 }
3153
3154 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
3155   \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
3156   \exp_args:Nnx \use:nn {
3157     % TODO
3158     \stex_annotate_invisible:nnn {vardecl}{\clist_use:Nn\l__stex_variables_names,}{
3159       #2
3160     }
3161   }{
3162     \_stex_reset:N \varnot
3163     \_stex_reset:N \vartype
3164     \_stex_reset:N \vardefi
3165   }
3166 }
3167
3168 \NewDocumentCommand \vardef { s } {
3169   \IfBooleanTF#1 {
3170     \__stex_variables_do_complex:nn
3171   }{
3172     \__stex_variables_do_simple:nnn
3173   }
3174 }
3175
3176 \NewDocumentCommand \svar { 0{} m }{
3177   \tl_if_empty:nTF {#1}{
3178     \str_set:Nn \l_tmpa_str { #2 }
3179   }{
3180     \str_set:Nn \l_tmpa_str { #1 }
3181   }
3182   \_stex_term_omv:nn {
3183     var://\l_tmpa_str
3184   }{
3185     \exp_args:Nnx \use:nn {
3186       \def\comp{\_varcomp}
3187       \str_set:Nx \STEXInternalCurrentSymbolStr { var://\l_tmpa_str }
3188       \comp{ #2 }
3189     }{
3190       \_stex_reset:N \comp
3191       \_stex_reset:N \STEXInternalCurrentSymbolStr
3192     }
3193   }
3194 }
3195
3196
3197
3198 \keys_define:nn { stex / varseq } {

```

```

3199 name .str_set_x:N = \l__stex_variables_name_str ,
3200 args .int_set:N = \l__stex_variables_args_int ,
3201 type .tl_set:N = \l__stex_variables_type_tl ,
3202 mid .tl_set:N = \l__stex_variables_mid_tl ,
3203 bind .choices:nn =
3204 {forall,exists}
3205 {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
3206 }
3207
3208 \cs_new_protected:Nn \l__stex_variables_seq_args:n {
3209 \str_clear:N \l__stex_variables_name_str
3210 \int_set:Nn \l__stex_variables_args_int 1
3211 \tl_clear:N \l__stex_variables_type_tl
3212 \str_clear:N \l__stex_variables_bind_str
3213
3214 \keys_set:nn { stex / varseq } { #1 }
3215 }
3216
3217 \NewDocumentCommand \varseq {m O{} m m m}{
3218 \l__stex_variables_seq_args:n { #2 }
3219 \str_if_empty:NT \l__stex_variables_name_str {
3220 \str_set:Nx \l__stex_variables_name_str { #1 }
3221 }
3222 \prop_clear:N \l_tmpa_prop
3223 \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
3224
3225 \seq_set_from_clist:Nn \l_tmpa_seq {#3}
3226 \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
3227 \msg_error:nnxx{stex}{error/seqlength}
3228 {\int_use:N \l__stex_variables_args_int}
3229 {\seq_count:N \l_tmpa_seq}
3230 }
3231 \seq_set_from_clist:Nn \l_tmpb_seq {#4}
3232 \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
3233 \msg_error:nnxx{stex}{error/seqlength}
3234 {\int_use:N \l__stex_variables_args_int}
3235 {\seq_count:N \l_tmpb_seq}
3236 }
3237 \prop_put:Nnn \l_tmpa_prop {starts} {#3}
3238 \prop_put:Nnn \l_tmpa_prop {ends} {#4}
3239
3240 \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
3241 \cs_set:Npn {\int_use:N \l__stex_variables_args_int} { #5 }
3242
3243 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3244 \int_step_inline:nn \l__stex_variables_args_int {
3245 \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
3246 }
3247 \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
3248 \tl_put_right:Nn \l_tmpa_tl {\,\ellipses,}
3249 \tl_if_empty:NF \l__stex_variables_mid_tl {
3250 \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
3251 \tl_put_right:Nn \l_tmpa_tl {\,\ellipses,}
3252 }

```

```

3253 \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3254 \int_step_inline:nn \l__stex_variables_args_int {
3255   \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
3256 }
3257 \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
3258 \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
3259
3260
3261 \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl
3262
3263 \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}
3264
3265 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3266
3267 \int_step_inline:nn \l__stex_variables_args_int {
3268   \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
3269     \STEXInternalTermMathArgiii{i##1}{0}{\exp_not:n{####}##1}
3270   }}
3271 }
3272
3273 \tl_set:Nx \l_tmpa_tl {
3274   \STEXInternalTermMathOMaiiii { varseq://\l__stex_variables_name_str}{0}{
3275     \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
3276   }
3277 }
3278
3279 \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \STEXInternalSymbolAfterInvokationTL} }
3280
3281 \exp_args:Nno \use:nn {
3282   \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
3283   \cs_set:Npn {\int_use:N \l__stex_variables_args_int}{\l_tmpa_tl}
3284
3285   \stex_debug:nn{sequences}{New~Sequence:~
3286     \expandafter\meaning\csname stex_varseq_\l__stex_variables_name_str _cs\endcsname\\~\\
3287     \prop_to_keyval:N \l_tmpa_prop
3288   }
3289   \stex_if_do_html:T{\stex_annotate_invisible:nnn{varseq}{\l__stex_variables_name_str}{
3290     \tl_if_empty:NF \l__stex_variables_type_tl {
3291       \stex_annotate:nnn {type}{\l__stex_variables_type_tl$}
3292     }
3293     \stex_annotate:nnn {args}{\int_use:N \l__stex_variables_args_int}{}
3294     \str_if_empty:NF \l__stex_variables_bind_str {
3295       \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{}
3296     }
3297     \stex_annotate:nnn{startindex}{\l__stex_variables_startindex}{\l__stex_variables_startindex$}
3298     \stex_annotate:nnn{endindex}{\l__stex_variables_endindex}{\l__stex_variables_endindex$}
3299
3300     \tl_clear:N \l_tmpa_tl
3301     \int_step_inline:nn \l__stex_variables_args_int {
3302       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3303         \stex_annotate:nnn{argmarker}{##1}{}
3304       } }
3305     }
3306     \stex_annotate_invisible:nnn { notationcomp }{}{

```



```

3307     \str_set:Nx \STEXInternalCurrentSymbolStr {varseq://\l__stex_variables_name_str }
3308     $ \exp_args:Nno \use:nn { \use:c {
3309         stex_varseq_\l__stex_variables_name_str _cs
3310     } } { \l_tmpa_tl } $
3311 }
3312 \stex_annotate_invisible:nnn { notationopcomp }{}{
3313     $ \prop_item:Nn \l_tmpa_prop { notation } $
3314 }
3315
3316 }}
3317
3318 \prop_set_eq:cN {stex_varseq_\l__stex_variables_name_str _prop}\l_tmpa_prop
3319 \ignorespacesandpars
3320 }
3321
3322 \end{package}

```

## Chapter 29

# STEX -Terms Implementation

```
3323 <*package>
3324
3325 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
3326
3327 <@@=stex_terms>
3328
3329 Warnings and error messages
3330 \msg_new:nnn{stex}{error/nonotation}{
3331   Symbol~#1~invoked,~but~has~no~notation#2!
3332 }
3333 \msg_new:nnn{stex}{error/notationarg}{
3334   Error~in~parsing~notation~#1
3335 }
3336 \msg_new:nnn{stex}{error/noop}{
3337   Symbol~#1~has~no~operator~notation~for~notation~#2
3338 }
3339 \msg_new:nnn{stex}{error/notallowed}{
3340   Symbol~invocation~#1~not~allowed~in~notation~component~of~#2
3341 }
3342 \msg_new:nnn{stex}{error/doubleargument}{
3343   Argument~#1~of~symbol~#2~already~assigned
3344 }
3345 \msg_new:nnn{stex}{error/overarity}{
3346   Argument~#1~invalid~for~symbol~#2~with~arity~#3
3347 }
```

### 29.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
3347
3348
3349 \bool_new:N \l_stex_allow_semantic_bool
3350 \bool_set_true:N \l_stex_allow_semantic_bool
3351
```

```

3352 \cs_new_protected:Nn \stex_invoke_symbol:n {
3353   \ifvmode\indent\fi
3354   \bool_if:NTF \l_stex_allow_semantic_bool {
3355     \str_if_eq:eeF {
3356       \prop_item:cn {
3357         l_stex_symdecl_#1_prop
3358       }{ deprecate }
3359     }{}{
3360       \msg_warning:nxxx{stex}{warning/deprecated}{
3361         Symbol~#1
3362       }{
3363         \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
3364       }
3365     }
3366     \if_mode_math:
3367       \exp_after:wN \__stex_terms_invoke_math:n
3368     \else:
3369       \exp_after:wN \__stex_terms_invoke_text:n
3370     \fi: { #1 }
3371   }{
3372     \msg_error:nxxx{stex}{error/notallowed}{#1}{\STEXInternalCurrentSymbolStr}
3373   }
3374 }
3375
3376 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
3377   \peek_charcode_remove:NTF ! {
3378     \__stex_terms_invoke_op_custom:nn {#1}
3379   }{
3380     \__stex_terms_invoke_custom:nn {#1}
3381   }
3382 }
3383
3384 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
3385   \peek_charcode_remove:NTF ! {
3386     % operator
3387     \peek_charcode_remove:NTF * {
3388       % custom op
3389       \__stex_terms_invoke_op_custom:nn {#1}
3390     }{
3391       % op notation
3392       \peek_charcode:NTF [ {
3393         \__stex_terms_invoke_op_notation:nw {#1}
3394       }{
3395         \__stex_terms_invoke_op_notation:nw {#1}[]
3396       }
3397     }
3398   }{
3399     \peek_charcode_remove:NTF * {
3400       \__stex_terms_invoke_custom:nn {#1}
3401       % custom
3402     }{
3403       % normal
3404       \peek_charcode:NTF [ {
3405         \__stex_terms_invoke_notation:nw {#1}

```

```

3406     }{
3407         \__stex_terms_invoke_notation:nw {#1}[]
3408     }
3409 }
3410 }
3411 }
3412
3413
3414 \cs_new_protected:Nn \__stex_terms_invoke_op_custom:nn {
3415     \exp_args:Nnx \use:nn {
3416         \def\comp{\_comp}
3417         \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3418         \bool_set_false:N \l_stex_allow_semantic_bool
3419         \stex_term_oms:nnn {#1}{#1 \c_hash_str CUSTOM-}{
3420             \comp{ #2 }
3421         }
3422     }{
3423         \stex_reset:N \comp
3424         \stex_reset:N \STEXInternalCurrentSymbolStr
3425         \bool_set_true:N \l_stex_allow_semantic_bool
3426     }
3427 }
3428
3429 \keys_define:nn { stex / terms } {
3430     % lang .tl_set_x:N = \l_stex_notation_lang_str ,
3431     variant .tl_set_x:N = \l_stex_notation_variant_str ,
3432     unknown .code:n      = \str_set:Nx
3433         \l_stex_notation_variant_str \l_keys_key_str
3434 }
3435
3436 \cs_new_protected:Nn \__stex_terms_args:n {
3437     % \str_clear:N \l_stex_notation_lang_str
3438     \str_clear:N \l_stex_notation_variant_str
3439
3440     \keys_set:nn { stex / terms } { #1 }
3441 }
3442
3443 \cs_new_protected:Nn \stex_find_notation:nn {
3444     \__stex_terms_args:n { #2 }
3445     \seq_if_empty:cTF {
3446         l_stex_symdecl_ #1 _notations
3447     } {
3448         \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
3449     } {
3450         \str_if_empty:NTF \l_stex_notation_variant_str {
3451             \seq_get_left:cN {l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
3452         }{
3453             \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
3454                 \l_stex_notation_variant_str
3455             }{
3456                 % \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3457             }{
3458                 \msg_error:nnxx{stex}{error/nonotation}{#1}{
3459                     ~\l_stex_notation_variant_str

```

```

3460     }
3461   }
3462 }
3463 }
3464 }
3465
3466 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3467   \exp_args:Nnx \use:nn {
3468     \def\comp{\_comp}
3469     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3470     \stex_find_notation:nn { #1 }{ #2 }
3471     \bool_set_false:N \l_stex_allow_semantic_bool
3472     \cs_if_exist:cTF {
3473       stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3474     }{
3475       \_stex_term_oms:nnn { #1 }{
3476         #1 \c_hash_str \l_stex_notation_variant_str
3477       }{
3478         \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3479       }
3480     }{
3481       \int_compare:nNnTF {\prop_item:cn {\l_stex_symdecl_#1_prop}{arity}} = 0{
3482         \cs_if_exist:cTF {
3483           stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3484         }{
3485           \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
3486             \_stex_reset:N \comp
3487             \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
3488             \_stex_reset:N \STEXInternalCurrentSymbolStr
3489             \bool_set_true:N \l_stex_allow_semantic_bool
3490           }
3491           \def\comp{\_comp}
3492           \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3493           \bool_set_false:N \l_stex_allow_semantic_bool
3494           \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3495         }{
3496           \msg_error:nnxx{stex}{error/nonotation}{#1}{
3497             ~\l_stex_notation_variant_str
3498           }
3499         }
3500       }{
3501         \msg_error:nnxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3502       }
3503     }
3504   }{
3505     \_stex_reset:N \comp
3506     \_stex_reset:N \STEXInternalCurrentSymbolStr
3507     \bool_set_true:N \l_stex_allow_semantic_bool
3508   }
3509 }
3510
3511 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {
3512   \stex_find_notation:nn { #1 }{ #2 }
3513   \cs_if_exist:cTF {

```

```

3514     stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3515   }{
3516     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
3517       \_stex_reset:N \comp
3518       \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
3519       \_stex_reset:N \STEXInternalCurrentSymbolStr
3520       \bool_set_true:N \l_stex_allow_semantic_bool
3521     }
3522     \def\comp{\_comp}
3523     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3524     \bool_set_false:N \l_stex_allow_semantic_bool
3525     \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3526   }{
3527     \msg_error:nnxx{stex}{error/nonotation}{#1}{
3528       ~\l_stex_notation_variant_str
3529     }
3530   }
3531 }
3532
3533 \prop_new:N \l__stex_terms_custom_args_prop
3534
3535 \cs_new_protected:Nn \__stex_terms_custom_comp:n { \bool_set_false:N \l_stex_allow_semantic_bool }
3536
3537 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3538   \exp_args:Nnx \use:nn {
3539     \def\comp{\_stex_terms_custom_comp:n}
3540     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3541     \prop_clear:N \l__stex_terms_custom_args_prop
3542     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3543     \prop_get:cnN {
3544       l_stex_symdecl_#1 _prop
3545     } { args } \l_tmpa_str
3546     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3547     \tl_set:Nn \arg { \_stex_terms_arg: }
3548     \str_if_empty:NTF \l_tmpa_str {
3549       \_stex_term_oms:nnn {#1}{#1\c_hash_str CUSTOM-}{\ignorespaces#2}
3550     }{
3551       \str_if_in:NnTF \l_tmpa_str b {
3552         \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
3553       }{
3554         \str_if_in:NnTF \l_tmpa_str B {
3555           \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
3556         }{
3557           \_stex_term_oma:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
3558         }
3559       }
3560     }
3561     % TODO check that all arguments exist
3562   }{
3563     \_stex_reset:N \STEXInternalCurrentSymbolStr
3564     \_stex_reset:N \arg
3565     \_stex_reset:N \comp
3566     \_stex_reset:N \l__stex_terms_custom_args_prop
3567     %\bool_set_true:N \l_stex_allow_semantic_bool

```

```

3568 }
3569 }
3570
3571 \NewDocumentCommand \__stex_terms_arg: { s O{} m}{
3572   \tl_if_empty:nTF {#2}{
3573     \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
3574     \bool_set_true:N \l_tmpa_bool
3575     \bool_do_while:Nn \l_tmpa_bool {
3576       \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int
3577         \int_incr:N \l_tmpa_int
3578       }{
3579         \bool_set_false:N \l_tmpa_bool
3580       }
3581     }
3582   }{
3583     \int_set:Nn \l_tmpa_int { #2 }
3584   }
3585   \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
3586   \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3587     \msg_error:nnxxx{stex}{error/overarity}
3588     {\int_use:N \l_tmpa_int}
3589     {\STEXInternalCurrentSymbolStr}
3590     {\str_count:N \l_tmpa_str}
3591   }
3592   \str_set:Nx \l_tmpa_str {\str_item:Nn \l_tmpa_str \l_tmpa_int}
3593   \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
3594     \bool_lazy_any:nF {
3595       {\str_if_eq_p:Vn \l_tmpa_str {a}}
3596       {\str_if_eq_p:Vn \l_tmpa_str {B}}
3597     }{
3598       \msg_error:nnxx{stex}{error/doubleargument}
3599       {\int_use:N \l_tmpa_int}
3600       {\STEXInternalCurrentSymbolStr}
3601     }
3602   }
3603   \exp_args:NNx \prop_put:Nnn \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {\ign
3604   \bool_if:NTF \l_stex_allow_semantic_bool \use_i:nn {
3605     \bool_set_true:N \l_stex_allow_semantic_bool
3606     \use:nn
3607   }
3608   {
3609     \IfBooleanTF#1{
3610       \stex_annotate_invisible:n { %TODO
3611         \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
3612       }
3613     }{ %TODO
3614       \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
3615     }}
3616     {\bool_set_false:N \l_stex_allow_semantic_bool}
3617   }
3618
3619
3620 \cs_new_protected:Nn \_stex_term_arg:nn {
3621   \bool_set_true:N \l_stex_allow_semantic_bool

```

```

3622 \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3623 \bool_set_false:N \l_stex_allow_semantic_bool
3624 }
3625
3626 \cs_new_protected:Npn \STEXInternalTermMathArgiii #1#2#3 {
3627 \exp_args:Nnx \use:nn
3628 { \int_set:Nn \l__stex_terms_downprec { #2 }
3629 \stex_term_arg:nn { #1 }{ #3 }
3630 }
3631 { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3632 }

```

(End definition for \stex\_invoke\_symbol:n. This function is documented on page 84.)

\STEXInternalTermMathAssocArgiiii

```

3633 \cs_new_protected:Npn \STEXInternalTermMathAssocArgiiii #1#2#3#4 {
3634 \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3635 \tl_set:Nn \l_tmpb_tl {\STEXInternalTermMathArgiii{#1}{#2}}
3636 \tl_if_empty:nTF { #3 }{
3637 \STEXInternalTermMathArgiii{#1}{#2}{#3}
3638 }{
3639 \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 } }{
3640 \expandafter\if\expandafter\relax\noexpand#3
3641 \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_maybe_sequence:Nn#3{#1}}
3642 \else
3643 \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_simple:nn{#1}{#3}}
3644 \fi
3645 \l_tmpa_tl
3646 }{
3647 \__stex_terms_math_assoc_arg_simple:nn{#1}{#3}
3648 }
3649 }
3650 }
3651
3652 \cs_new_protected:Nn \__stex_terms_math_assoc_arg_maybe_sequence:Nn {
3653 \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
3654 \str_if_empty:NNTF \l_tmpa_str {
3655 \exp_args:Nx \cs_if_eq:NNTF {
3656 \tl_head:N #1
3657 } \stex_invoke_sequence:n {
3658 \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
3659 \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
3660 \tl_set:Nx \l_tmpa_tl {\prop_item:cn {stex_varseq_\l_tmpa_str _prop}{notation}}
3661 \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
3662 \tl_set:Nx \l_tmpa_tl {\exp_not:N \exp_not:n{
3663 \exp_not:n{\exp_args:Nnx \use:nn} {
3664 \exp_not:n {
3665 \def\comp{\_varcomp}
3666 \str_set:Nn \STEXInternalCurrentSymbolStr
3667 } {varseq://\l_tmpa_str}
3668 \exp_not:n{ ##1 }
3669 }{
3670 \exp_not:n {
3671 \stex_reset:N \comp

```



```

3672         \_stex_reset:N \STEXInternalCurrentSymbolStr
3673     }
3674 }
3675 }}}
3676 \exp_args:Nno \use:nn {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
3677 \seq_reverse:N \l_tmpa_seq
3678 \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
3679 \seq_map_inline:Nn \l_tmpa_seq {
3680     \exp_args:NNNo \exp_args:Nno \tl_set:No \l_tmpa_tl {
3681         \exp_args:Nno
3682         \l_tmpa_cs { ##1 } \l_tmpa_tl
3683     }
3684 }
3685 \tl_set:Nx \l_tmpa_tl {
3686     \_stex_term_omv:nn {varseq://\l_tmpa_str}{
3687         \exp_args:No \exp_not:n \l_tmpa_tl
3688     }
3689 }
3690 \exp_args:No\l_tmpb_tl\l_tmpa_tl
3691 }{
3692     \_stex_terms_math_assoc_arg_simple:nn{#2} { #1 }
3693 }
3694 } {
3695     \_stex_terms_math_assoc_arg_simple:nn{#2} { #1 }
3696 }
3697 }
3698 }
3699
3700 \cs_new_protected:Nn \_stex_terms_math_assoc_arg_simple:nn {
3701     \clist_set:Nn \l_tmpa_clist{ #2 }
3702     \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3703         \tl_set:Nn \l_tmpa_tl { \_stex_term_arg:nn{A#1}{ #2 } }
3704     }{
3705         \clist_reverse:N \l_tmpa_clist
3706         \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3707         \tl_set:Nx \l_tmpa_tl { \_stex_term_arg:nn{A#1}{
3708             \exp_args:No \exp_not:n \l_tmpa_tl
3709         }}
3710         \clist_map_inline:Nn \l_tmpa_clist {
3711             \exp_args:NNNo \exp_args:Nno \tl_set:No \l_tmpa_tl {
3712                 \exp_args:Nno
3713                 \l_tmpa_cs { \_stex_term_arg:nn{A#1}{##1} } \l_tmpa_tl
3714             }
3715         }
3716     }
3717     \exp_args:No\l_tmpb_tl\l_tmpa_tl
3718 }

```

(End definition for `\STEXInternalTermMathAssocArgiiii`. This function is documented on page 85.)

## 29.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
3719 \tl_const:Nx \infprec {\int_use:N \c_max_int}
3720 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
3721 \int_new:N \l__stex_terms_downprec
3722 \int_set_eq:NN \l__stex_terms_downprec \infprec

(End definition for \infprec, \neginfprec, and \l__stex_terms_downprec. These variables are docu-
mented on page 85.)

Bracketing:

\l__stex_terms_left_bracket_str
\l__stex_terms_right_bracket_str
3723 \tl_set:Nn \l__stex_terms_left_bracket_str (
3724 \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
3725 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
3726   \bool_if:NTF \l__stex_terms_brackets_done_bool {
3727     \bool_set_false:N \l__stex_terms_brackets_done_bool
3728     #2
3729   } {
3730     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
3731       \bool_if:NTF \l__stex_inarray_bool { #2 }{
3732         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
3733         \dobrackets { #2 }
3734       }
3735     }{ #2 }
3736   }
3737 }

(End definition for \__stex_terms_maybe_brackets:nn.)

\dobrackets
3738 \bool_new:N \l__stex_terms_brackets_done_bool
3739 %\RequirePackage{scalerel}
3740 \cs_new_protected:Npn \dobrackets #1 {
3741   %\ThisStyle{\if D\m@switch
3742   %   \exp_args:Nnx \use:nn
3743   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
3744   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
3745   % \else
3746   \exp_args:Nnx \use:nn
3747   {
3748     \bool_set_true:N \l__stex_terms_brackets_done_bool
3749     \int_set:Nn \l__stex_terms_downprec \infprec
3750     \l__stex_terms_left_bracket_str
3751     #1
3752   }
3753   {
3754     \bool_set_false:N \l__stex_terms_brackets_done_bool
3755     \l__stex_terms_right_bracket_str
3756     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3757   }
3758   %\fi}
3759 }

```

(End definition for `\dobrackets`. This function is documented on page 85.)

### `\withbrackets`

```

3760 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
3761   \exp_args:Nnx \use:nn
3762   {
3763     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
3764     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
3765     #3
3766   }
3767   {
3768     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
3769     {\l__stex_terms_left_bracket_str}
3770     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
3771     {\l__stex_terms_right_bracket_str}
3772   }
3773 }

```

(End definition for `\withbrackets`. This function is documented on page 85.)

### `\STEXinvisible`

```

3774 \cs_new_protected:Npn \STEXinvisible #1 {
3775   \stex_annotate_invisible:n { #1 }
3776 }

```

(End definition for `\STEXinvisible`. This function is documented on page 85.)

OMDoc terms:

### `\STEXInternalTermMathOMSiiii`

```

3777 \cs_new_protected:Nn \_stex_term_oms:nnn {
3778   \stex_annotate:nnn{ OMID }{ #2 }{
3779     #3
3780   }
3781 }
3782
3783 \cs_new_protected:Npn \STEXInternalTermMathOMSiiii #1#2#3#4 {
3784   \_stex_terms_maybe_brackets:nn { #3 }{
3785     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3786   }
3787 }

```

(End definition for `\STEXInternalTermMathOMSiiii`. This function is documented on page 84.)

### `\_stex_term_math_omv:nn`

```

3788 \cs_new_protected:Nn \_stex_term_omv:nn {
3789   \stex_annotate:nnn{ OMV }{ #1 }{
3790     #2
3791   }
3792 }

```

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

`\STEXInternalTermMathOMAi`

```

3793 \cs_new_protected:Nn \stex_term_oma:nnn {
3794   \stex_annotate:nnn{ OMA }{ #2 }{
3795     #3
3796   }
3797 }
3798
3799 \cs_new_protected:Npn \STEXInternalTermMathOMAi #1#2#3#4 {
3800   \__stex_terms_maybe_brackets:nn { #3 }{
3801     \stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3802   }
3803 }

```

(End definition for `\STEXInternalTermMathOMAi`. This function is documented on page 84.)

`\STEXInternalTermMathOMBi`

```

3804 \cs_new_protected:Nn \stex_term_ombind:nnn {
3805   \stex_annotate:nnn{ OMBIND }{ #2 }{
3806     #3
3807   }
3808 }
3809
3810 \cs_new_protected:Npn \STEXInternalTermMathOMBi #1#2#3#4 {
3811   \__stex_terms_maybe_brackets:nn { #3 }{
3812     \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3813   }
3814 }

```

(End definition for `\STEXInternalTermMathOMBi`. This function is documented on page 84.)

`\symref`

`\symname`

```

3815 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3816
3817 \keys_define:nn { stex / symname } {
3818   pre      .tl_set_x:N      = \l__stex_terms_pre_tl ,
3819   post     .tl_set_x:N      = \l__stex_terms_post_tl ,
3820   root     .tl_set_x:N      = \l__stex_terms_root_tl
3821 }
3822
3823 \cs_new_protected:Nn \stex_symname_args:n {
3824   \tl_clear:N \l__stex_terms_post_tl
3825   \tl_clear:N \l__stex_terms_pre_tl
3826   \tl_clear:N \l__stex_terms_root_str
3827   \keys_set:nn { stex / symname } { #1 }
3828 }
3829
3830 \NewDocumentCommand \symref { m m }{
3831   \let\compemph_uri_prev:\compemph_uri
3832   \let\compemph_uri\symrefemph_uri
3833   \STEXsymbol{#1}!{ #2 }
3834   \let\compemph_uri\compemph_uri_prev:
3835 }
3836
3837 \NewDocumentCommand \synonym { 0{ } m m }{

```

```

3838 \stex_symname_args:n { #1 }
3839 \let\compemph_uri_prev:\compemph@uri
3840 \let\compemph@uri\symrefemph@uri
3841 % TODO
3842 \STEXsymbol{#2}!\{ \l__stex_terms_pre_tl #3 \l__stex_terms_post_tl }
3843 \let\compemph@uri\compemph_uri_prev:
3844 }
3845
3846 \NewDocumentCommand \symname { 0{ } m }{
3847 \stex_symname_args:n { #1 }
3848 \stex_get_symbol:n { #2 }
3849 \str_set:Nx \l_tmpa_str {
3850 \prop_item:cn { \l__stex_symdecl_ \l__stex_get_symbol_uri_str _prop } { name }
3851 }
3852 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3853
3854 \let\compemph_uri_prev:\compemph@uri
3855 \let\compemph@uri\symrefemph@uri
3856 \exp_args:NNx \use:nn
3857 \stex_invoke_symbol:n { { \l__stex_get_symbol_uri_str }!\ifmode*\fi{
3858 \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
3859 } }
3860 \let\compemph@uri\compemph_uri_prev:
3861 }
3862
3863 \NewDocumentCommand \Symname { 0{ } m }{
3864 \stex_symname_args:n { #1 }
3865 \stex_get_symbol:n { #2 }
3866 \str_set:Nx \l_tmpa_str {
3867 \prop_item:cn { \l__stex_symdecl_ \l__stex_get_symbol_uri_str _prop } { name }
3868 }
3869 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3870 \let\compemph_uri_prev:\compemph@uri
3871 \let\compemph@uri\symrefemph@uri
3872 \exp_args:NNx \use:nn
3873 \stex_invoke_symbol:n { { \l__stex_get_symbol_uri_str }!\ifmode*\fi{
3874 \exp_after:wN \stex_capitalize:n \l_tmpa_str
3875 \l__stex_terms_post_tl
3876 } }
3877 \let\compemph@uri\compemph_uri_prev:
3878 }

```

(End definition for `\symref` and `\symname`. These functions are documented on page 84.)

## 29.3 Notation Components

```

3879 <@@=stex_notationcomps>

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

```

```

3880 \cs_new_protected:Npn \_comp #1 {
3881 \str_if_empty:NF \STEXInternalCurrentSymbolStr {
3882 \stex_html_backend:TF {
3883 \stex_annotate:nnn { comp }{ \STEXInternalCurrentSymbolStr }{ #1 }
3884 }{

```

```

3885     \exp_args:Nnx \compemph@uri { #1 } { \STEXInternalCurrentSymbolStr }
3886   }
3887 }
3888 }
3889
3890 \cs_new_protected:Npn \_varcomp #1 {
3891   \str_if_empty:NF \STEXInternalCurrentSymbolStr {
3892     \stex_html_backend:TF {
3893       \stex_annotate:nnn { varcomp }{ \STEXInternalCurrentSymbolStr }{ #1 }
3894     }{
3895       \exp_args:Nnx \varempemph@uri { #1 } { \STEXInternalCurrentSymbolStr }
3896     }
3897   }
3898 }
3899
3900 \def\comp{\_comp}
3901
3902 \cs_new_protected:Npn \compemph@uri #1 #2 {
3903   \compemph{ #1 }
3904 }
3905
3906
3907 \cs_new_protected:Npn \compemph #1 {
3908   #1
3909 }
3910
3911 \cs_new_protected:Npn \defemph@uri #1 #2 {
3912   \defemph{#1}
3913 }
3914
3915 \cs_new_protected:Npn \defemph #1 {
3916   \textbf{#1}
3917 }
3918
3919 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3920   \symrefemph{#1}
3921 }
3922
3923 \cs_new_protected:Npn \symrefemph #1 {
3924   \emph{#1}
3925 }
3926
3927 \cs_new_protected:Npn \varempemph@uri #1 #2 {
3928   \varempemph{#1}
3929 }
3930
3931 \cs_new_protected:Npn \varempemph #1 {
3932   #1
3933 }

```

(End definition for `\comp` and others. These functions are documented on page 85.)

**\ellipses**

```

3934 \NewDocumentCommand \ellipses {} { \ldots }

```

(End definition for `\ellipses`. This function is documented on page 85.)

```

\parray
\prmatrix 3935 \bool_new:N \l_stex_inpparray_bool
\parrayline 3936 \bool_set_false:N \l_stex_inpparray_bool
\parraylineh 3937 \NewDocumentCommand \parray { m m } {
\parraycell 3938 \begin{group}
3939 \bool_set_true:N \l_stex_inpparray_bool
3940 \begin{array}{#1}
3941 #2
3942 \end{array}
3943 \end{group}
3944 }
3945
3946 \NewDocumentCommand \prmatrix { m } {
3947 \begin{group}
3948 \bool_set_true:N \l_stex_inpparray_bool
3949 \begin{matrix}
3950 #1
3951 \end{matrix}
3952 \end{group}
3953 }
3954
3955 \def \maybepline {
3956 \bool_if:NT \l_stex_inpparray_bool {\hline}
3957 }
3958
3959 \def \parrayline #1 #2 {
3960 #1 #2 \bool_if:NT \l_stex_inpparray_bool {\}
3961 }
3962
3963 \def \pmrow #1 { \parrayline{}{ #1 } }
3964
3965 \def \parraylineh #1 #2 {
3966 #1 #2 \bool_if:NT \l_stex_inpparray_bool {\hline}
3967 }
3968
3969 \def \parraycell #1 {
3970 #1 \bool_if:NT \l_stex_inpparray_bool {&}
3971 }

```

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

## 29.4 Variables

3972 `<@@=stex_variables>`

`\stex_invoke_variable:n` Invokes a variable

```

3973 \cs_new_protected:Nn \stex_invoke_variable:n {
3974 \if_mode_math:
3975 \exp_after:wN \__stex_variables_invoke_math:n
3976 \else:
3977 \exp_after:wN \__stex_variables_invoke_text:n

```

```

3978 \fi: {#1}
3979 }
3980
3981 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
3982 \peek_charcode_remove:NTF ! {
3983 \__stex_variables_invoke_op_custom:nn {#1}
3984 }{
3985 \__stex_variables_invoke_custom:nn {#1}
3986 }
3987 }
3988
3989
3990 \cs_new_protected:Nn \__stex_variables_invoke_math:n {
3991 \peek_charcode_remove:NTF ! {
3992 \peek_charcode_remove:NTF ! {
3993 \peek_charcode:NTF [ {
3994 % TODO throw error
3995 }{
3996 \__stex_variables_invoke_op_custom:nn
3997 }
3998 }{
3999 \__stex_variables_invoke_op:n { #1 }
4000 }
4001 }{
4002 \peek_charcode_remove:NTF * {
4003 \__stex_variables_invoke_custom:nn { #1 }
4004 }{
4005 \__stex_variables_invoke_math_ii:n { #1 }
4006 }
4007 }
4008 }
4009
4010 \cs_new_protected:Nn \__stex_variables_invoke_op_custom:nn {
4011 \exp_args:Nnx \use:nn {
4012 \def\comp{\_varcomp}
4013 \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4014 \bool_set_false:N \l_stex_allow_semantic_bool
4015 \stex_term_omv:nn {var://#1}{
4016 \comp{ #2 }
4017 }
4018 }{
4019 \stex_reset:N \comp
4020 \stex_reset:N \STEXInternalCurrentSymbolStr
4021 \bool_set_true:N \l_stex_allow_semantic_bool
4022 }
4023 }
4024
4025 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
4026 \cs_if_exist:cTF {
4027 stex_var_op_notation_ #1 _cs
4028 }{
4029 \exp_args:Nnx \use:nn {
4030 \def\comp{\_varcomp}
4031 \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }

```



```

4032     \stex_term_omv:nn { var://#1 }{
4033       \use:c{stex_var_op_notation_ #1 _cs }
4034     }
4035   }{
4036     \stex_reset:N \comp
4037     \stex_reset:N \STEXInternalCurrentSymbolStr
4038   }
4039 }{
4040   \int_compare:nNnTF {\prop_item:cn {l_stex_variable_#1_prop}{arity}} = 0{
4041     \__stex_variables_invoke_math_ii:n {#1}
4042   }{
4043     \msg_error:nnxx{stex}{error/noop}{variable~#1}{ }
4044   }
4045 }
4046 }
4047
4048 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n #1 {
4049   \cs_if_exist:cTF {
4050     stex_var_notation_#1_cs
4051   }{
4052     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4053       \stex_reset:N \comp
4054       \stex_reset:N \STEXInternalSymbolAfterInvokationTL
4055       \stex_reset:N \STEXInternalCurrentSymbolStr
4056       \bool_set_true:N \l_stex_allow_semantic_bool
4057     }
4058     \def\comp{\_varcomp}
4059     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4060     \bool_set_false:N \l_stex_allow_semantic_bool
4061     \use:c{stex_var_notation_#1_cs}
4062   }{
4063     \msg_error:nnxx{stex}{error/nonotation}{variable~#1}{s}
4064   }
4065 }
4066
4067 \cs_new_protected:Nn \__stex_variables_invoke_custom:nn {
4068   \exp_args:Nnx \use:nn {
4069     \def\comp{\_varcomp}
4070     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4071     \prop_clear:N \l__stex_terms_custom_args_prop
4072     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
4073     \prop_get:cnN {
4074       l_stex_variable_#1_prop
4075     }{ args } \l_tmpa_str
4076     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
4077     \tl_set:Nn \arg { \__stex_terms_arg: }
4078     \str_if_empty:NTF \l_tmpa_str {
4079       \stex_term_omv:nn {var://#1}{\ignorespaces#2}
4080     }{
4081       \str_if_in:NnTF \l_tmpa_str b {
4082         \stex_term_ombind:nnn {var://#1}{ }\{\ignorespaces#2}
4083       }{
4084         \str_if_in:NnTF \l_tmpa_str B {
4085           \stex_term_ombind:nnn {var://#1}{ }\{\ignorespaces#2}

```

```

4086     }{
4087       \_stex_term_oma:nnn {var://#1}{\ignorespaces#2}
4088     }
4089   }
4090 }
4091 % TODO check that all arguments exist
4092 }{
4093   \_stex_reset:N \STEXInternalCurrentSymbolStr
4094   \_stex_reset:N \arg
4095   \_stex_reset:N \comp
4096   \_stex_reset:N \l__stex_terms_custom_args_prop
4097   %\bool_set_true:N \l_stex_allow_semantic_bool
4098 }
4099 }

```

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

## 29.5 Sequences

```

4100 <@@=stex_sequences>
4101
4102 \cs_new_protected:Nn \stex_invoke_sequence:n {
4103   \peek_charcode_remove:NTF ! {
4104     \_stex_term_omv:nn {varseq://#1}{
4105       \exp_args:Nnx \use:nn {
4106         \def\comp{\_varcomp}
4107         \str_set:Nn \STEXInternalCurrentSymbolStr {varseq://#1}
4108         \prop_item:cn{stex_varseq_#1_prop}{notation}
4109       }{
4110         \_stex_reset:N \comp
4111         \_stex_reset:N \STEXInternalCurrentSymbolStr
4112       }
4113     }
4114   }{
4115     \bool_set_false:N \l_stex_allow_semantic_bool
4116     \def\comp{\_varcomp}
4117     \str_set:Nn \STEXInternalCurrentSymbolStr {varseq://#1}
4118     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4119       \_stex_reset:N \comp
4120       \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
4121       \_stex_reset:N \STEXInternalCurrentSymbolStr
4122       \bool_set_true:N \l_stex_allow_semantic_bool
4123     }
4124     \use:c { stex_varseq_#1_cs }
4125   }
4126 }
4127 </package>

```

## Chapter 30

# STEX -Structural Features Implementation

```
4128 ⟨*package⟩
4129
4130 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4131
4132 Warnings and error messages
4133 \msg_new:nnn{stex}{error/copymodule/notallowed}{
4134   Symbol~#1~can~not~be~assigned~in~copymodule~#2
4135 }
4136 \msg_new:nnn{stex}{error/interpretmodule/nodfiniens}{
4137   Symbol~#1~not~assigned~in~interpretmodule~#2
4138 }
4139 \msg_new:nnn{stex}{error/unknownstructure}{
4140   No~structure~#1~found!
4141 }
4142
4143 \msg_new:nnn{stex}{error/unknownfield}{
4144   No~field~#1~in~instance~#2~found!~\#3
4145 }
4146
4147 \msg_new:nnn{stex}{error/keyval}{
4148   Invalid~key=value~pair~#1
4149 }
4150 \msg_new:nnn{stex}{error/instantiate/missing}{
4151   Assignments~missing~in~instantiate:~#1
4152 }
4153 \msg_new:nnn{stex}{error/incompatible}{
4154   Incompatible~signature:~#1~(#2)~and~#3~(#4)
4155 }
4156
```

## 30.1 Imports with modification

```

4157 <@@=stex_copymodule>
4158 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
4159   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4160     \tl_set:Nn \l_tmpa_tl { #1 }
4161     \__stex_copymodule_get_symbol_from_cs:
4162   }{
4163     % argument is a string
4164     % is it a command name?
4165     \cs_if_exist:cTF { #1 }{
4166       \cs_set_eq:Nc \l_tmpa_tl { #1 }
4167       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4168       \str_if_empty:NNTF \l_tmpa_str {
4169         \exp_args:Nx \cs_if_eq:NNTF {
4170           \tl_head:N \l_tmpa_tl
4171         } \stex_invoke_symbol:n {
4172           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
4173         }{
4174           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4175         }
4176       } {
4177         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4178       }
4179     }{
4180       % argument is not a command name
4181       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4182       % \l_stex_all_symbols_seq
4183     }
4184   }
4185 }
4186
4187 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
4188   \str_set:Nn \l_tmpa_str { #1 }
4189   \bool_set_false:N \l_tmpa_bool
4190   \bool_if:NF \l_tmpa_bool {
4191     \tl_set:Nn \l_tmpa_tl {
4192       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
4193     }
4194     \str_set:Nn \l_tmpa_str { #1 }
4195     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4196     \seq_map_inline:Nn #2 {
4197       \str_set:Nn \l_tmpb_str { ##1 }
4198       \str_if_eq:eeT { \l_tmpa_str } {
4199         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4200       } {
4201         \seq_map_break:n {
4202           \tl_set:Nn \l_tmpa_tl {
4203             \str_set:Nn \l_stex_get_symbol_uri_str {
4204               ##1
4205             }
4206           }
4207         }
4208       }

```

```

4209     }
4210     \l_tmpa_tl
4211   }
4212 }
4213
4214 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
4215   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4216     { \tl_tail:N \l_tmpa_tl }
4217   \tl_if_single:NTF \l_tmpa_tl {
4218     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
4219       \exp_after:wN \str_set:Nn \exp_after:wN
4220         \l_stex_get_symbol_uri_str \l_tmpa_tl
4221       \__stex_copymodule_get_symbol_check:n { #1 }
4222     }{
4223       % TODO
4224       % tail is not a single group
4225     }
4226   }{
4227     % TODO
4228     % tail is not a single group
4229   }
4230 }
4231
4232 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
4233   \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
4234     \msg_error:nxxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
4235       :~\seq_use:Nn #1 {,~}
4236     }
4237   }
4238 }
4239
4240 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
4241   % import module
4242   \stex_import_module_uri:nn { #1 } { #2 }
4243   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
4244   \stex_import_require_module:nnnn
4245     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4246     { \l_stex_import_path_str } { \l_stex_import_name_str }
4247
4248   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
4249   \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
4250
4251   % fields
4252   \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
4253   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4254     \seq_map_inline:cn {c_stex_module_##1_constants}{
4255       \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
4256         ##1 ? ####1
4257       }
4258     }
4259   }
4260
4261   % setup prop
4262   \seq_clear:N \l_tmpa_seq

```

```

4263 \exp_args:Nn \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
4264   name      = \l_stex_current_copymodule_name_str ,
4265   module    = \l_stex_current_module_str ,
4266   from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
4267   includes  = \l_tmpa_seq %,
4268 % fields    = \l_tmpa_seq
4269 }
4270 \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
4271   as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
4272 \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_seq {
4273 \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
4274
4275 \stex_if_do_html:T {
4276   \begin{stex_annotate_env} {#4} {
4277     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4278   }
4279   \stex_annotate_invisible:nnn{domain}{\l_stex_import_ns_str ?\l_stex_import_name_str}{}
4280 }
4281 }
4282
4283 \cs_new_protected:Nn \stex_copymodule_end:n {
4284   % apply to every field
4285   \def \l_tmpa_cs ##1 ##2 {#1}
4286
4287   \tl_clear:N \__stex_copymodule_module_tl
4288   \tl_clear:N \__stex_copymodule_exec_tl
4289
4290   %\prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
4291   \seq_clear:N \__stex_copymodule_fields_seq
4292
4293   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4294     \seq_map_inline:cn {c_stex_module_##1_constants}{
4295
4296       \tl_clear:N \__stex_copymodule_curr_symbol_tl % <- wrap in current symbol html
4297       \l_tmpa_cs{##1}{####1}
4298
4299       \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
4300         \str_set_eq:Nc \__stex_copymodule_curr_name_str {l__stex_copymodule_copymodule_##1?#
4301         \stex_if_do_html:T {
4302           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4303             \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?###
4304           }
4305         }
4306       }{
4307         \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str /
4308       }
4309
4310       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
4311       \prop_put:Nnx \l_tmpa_prop { name } \__stex_copymodule_curr_name_str
4312       \prop_put:Nnx \l_tmpa_prop { module } \l_stex_current_module_str
4313
4314       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4315         \stex_if_do_html:T {
4316           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {

```

```

4317         $\stex_annotate_invisible:nnn{definiens}{\exp_after:wN \exp_not:N\csname l__st
4318     }
4319 }
4320 \prop_put:Nnn \l_tmpa_prop { defined } { true }
4321 }
4322
4323 \stex_add_constant_to_current_module:n \__stex_copymodule_curr_name_str
4324 \tl_put_right:Nx \__stex_copymodule_module_tl {
4325     \seq_clear:c {l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_n
4326     \prop_set_from_keyval:cn {
4327         l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_name_str _prop
4328     }{
4329         \prop_to_keyval:N \l_tmpa_prop
4330     }
4331 }
4332
4333 \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
4334     \stex_if_do_html:T {
4335         \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4336             \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1
4337         }
4338     }
4339     \tl_put_right:Nx \__stex_copymodule_module_tl {
4340         \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
4341             \stex_invoke_symbol:n {
4342                 \l_stex_current_module_str ? \__stex_copymodule_curr_name_str
4343             }
4344         }
4345     }
4346 }
4347
4348 \seq_put_right:Nx \__stex_copymodule_fields_seq {\l_stex_current_module_str ? \__stex_
4349
4350 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4351     \stex_copy_notations:nn {\l_stex_current_module_str ? \__stex_copymodule_curr_name_s
4352 }
4353
4354 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4355     \stex_if_do_html:TF{
4356         \stex_annotate_invisible:nnn{assignment} {##1?####1} { \exp_after:wN \exp_not:n \e
4357     }{
4358         \exp_after:wN \exp_not:n \exp_after:wN {\__stex_copymodule_curr_symbol_tl}
4359     }
4360 }
4361 }
4362 }
4363
4364
4365 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \__stex_copymodule_fields_seq
4366 \tl_put_left:Nx \__stex_copymodule_module_tl {
4367     \prop_set_from_keyval:cn {
4368         l_stex_copymodule_ \l_stex_current_module_str? \l_stex_current_copymodule_name_str _pro
4369     }{
4370         \prop_to_keyval:N \l_stex_current_copymodule_prop

```

```

4371   }
4372 }
4373
4374 \seq_gput_right:cx{c_stex_module_\l_stex_current_module_str _copymodules}{
4375   \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4376 }
4377
4378 \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4379 \stex_debug:nn{copymodule}{result:\meaning \__stex_copymodule_module_tl}
4380 \stex_debug:nn{copymodule}{output:\meaning \__stex_copymodule_exec_tl}
4381
4382 \__stex_copymodule_exec_tl
4383 \stex_if_do_html:T {
4384   \end{stex_annotate_env}
4385 }
4386 }
4387
4388 \NewDocumentEnvironment {copymodule} { 0{} m m}{
4389   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ copymodule }
4390   \stex_deactivate_macro:Nn \symdecl {module~environments}
4391   \stex_deactivate_macro:Nn \symdef {module~environments}
4392   \stex_deactivate_macro:Nn \notation {module~environments}
4393   \stex_reactivate_macro:N \assign
4394   \stex_reactivate_macro:N \renamedekl
4395   \stex_reactivate_macro:N \donotcopy
4396   \stex_smsmode_do:
4397 }{
4398   \stex_copymodule_end:n {}
4399 }
4400
4401 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
4402   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ interpretmodule }
4403   \stex_deactivate_macro:Nn \symdecl {module~environments}
4404   \stex_deactivate_macro:Nn \symdef {module~environments}
4405   \stex_deactivate_macro:Nn \notation {module~environments}
4406   \stex_reactivate_macro:N \assign
4407   \stex_reactivate_macro:N \renamedekl
4408   \stex_reactivate_macro:N \donotcopy
4409   \stex_smsmode_do:
4410 }{
4411   \stex_copymodule_end:n {
4412     \tl_if_exist:cF {
4413       l__stex_copymodule_copymodule_##1?##2_def_tl
4414     }{
4415       \str_if_eq:eeF {
4416         \prop_item:cn{
4417           l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
4418         }{ true }{
4419           \msg_error:nxxx{stex}{error/interpretmodule/nodéfiniens}{
4420             ##1?##2
4421           }{\l_stex_current_copymodule_name_str}
4422         }
4423       }
4424     }

```



```

4425 }
4426
4427 \iffalse \begin{stex_annotate_env} \fi
4428 \NewDocumentEnvironment {realization} { 0 } { m } {
4429   \stex_copymodule_start:nnnn { #1 } { #2 } { #2 } { realize }
4430   \stex_deactivate_macro:Nn \symdecl {module~environments}
4431   \stex_deactivate_macro:Nn \symdef {module~environments}
4432   \stex_deactivate_macro:Nn \notation {module~environments}
4433   \stex_reactivate_macro:N \donotcopy
4434   \stex_reactivate_macro:N \assign
4435   \stex_smsmode_do:
4436 } {
4437   \stex_import_module_uri:nn { #1 } { #2 }
4438   \tl_clear:N \__stex_copymodule_exec_tl
4439   \tl_set:Nx \__stex_copymodule_module_tl {
4440     \stex_import_require_module:nnnn
4441     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4442     { \l_stex_import_path_str } { \l_stex_import_name_str }
4443   }
4444
4445   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4446     \seq_map_inline:cn {c_stex_module_##1_constants}{
4447       \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str / #1 }
4448       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?###1_def_tl}{
4449         \stex_if_do_html:T {
4450           \tl_put_right:Nx \__stex_copymodule_exec_tl {
4451             \stex_annotate_invisible:nnn{assignment} {##1?###1} {
4452               $\stex_annotate_invisible:nnn{definien}{}{\exp_after:wN \exp_not:N\csname l__stex_
4453             }
4454           }
4455         }
4456         \tl_put_right:Nx \__stex_copymodule_module_tl {
4457           \prop_put:cnn {l_stex_symdecl_##1?###1_prop}{ defined }{ true }
4458         }
4459       }
4460     }
4461   }
4462   \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4463
4464   \__stex_copymodule_exec_tl
4465   \stex_if_do_html:T {\end{stex_annotate_env}}
4466 }
4467
4468 \NewDocumentCommand \donotcopy { m } {
4469   \str_clear:N \l_stex_import_name_str
4470   \str_set:Nn \l_tmpa_str { #1 }
4471   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4472   \seq_map_inline:Nn \l_stex_all_modules_seq {
4473     \str_set:Nn \l_tmpb_str { ##1 }
4474     \str_if_eq:eeT { \l_tmpa_str } {
4475       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4476     } {
4477       \seq_map_break:n {
4478         \stex_if_do_html:T {

```

```

4479         \stex_if_smsmode:F {
4480             \stex_annotate_invisible:nnn{donotcopy}{##1}{
4481                 \stex_annotate:nnn{domain}{##1}{}}
4482             }
4483         }
4484     }
4485     \str_set_eq:NN \l_stex_import_name_str \l_tmpb_str
4486 }
4487 }
4488 \seq_map_inline:cn {c_stex_module_###1_copymodules}{
4489     \str_set:Nn \l_tmpb_str { #####1 }
4490     \str_if_eq:eeT { \l_tmpa_str } {
4491         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4492     } {
4493         \seq_map_break:n {\seq_map_break:n {
4494             \stex_if_do_html:T {
4495                 \stex_if_smsmode:F {
4496                     \stex_annotate_invisible:nnn{donotcopy}{#####1}{
4497                         \stex_annotate:nnn{domain}{
4498                             \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4499                         }{}
4500                     }
4501                 }
4502             }
4503             \str_set:Nx \l_stex_import_name_str {
4504                 \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4505             }
4506         }}
4507     }
4508 }
4509 }
4510 \str_if_empty:NTF \l_stex_import_name_str {
4511     % TODO throw error
4512 }{
4513     \stex_collect_imports:n {\l_stex_import_name_str }
4514     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4515         \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
4516         \seq_map_inline:cn {c_stex_module_###1_constants}{
4517             \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? #####1 }
4518             \bool_lazy_any:nT {
4519                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_name_str}}
4520                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_macroname_str}}
4521                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_def_tl}}
4522             }{
4523                 % TODO throw error
4524             }
4525         }
4526     }
4527     \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
4528     \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_name_str }
4529     \prop_put:Nno \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
4530 }
4531 \stex_smsmode_do:
4532 }

```

```

4533
4534 \NewDocumentCommand \assign { m m }{
4535   \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
4536   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
4537   \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_def_tl}{#2}
4538   \stex_smsmode_do:
4539 }
4540
4541 \keys_define:nn { stex / renamedecl } {
4542   name          .str_set_x:N = \l_stex_renamedecl_name_str
4543 }
4544 \cs_new_protected:Nn \__stex_copymodule_renamedecl_args:n {
4545   \str_clear:N \l_stex_renamedecl_name_str
4546   \keys_set:nn { stex / renamedecl } { #1 }
4547 }
4548
4549 \NewDocumentCommand \renamedecl { O{} m m }{
4550   \__stex_copymodule_renamedecl_args:n { #1 }
4551   \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
4552   \stex_debug:nn{renamedecl}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
4553   \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_macroname_str}{#3}
4554   \str_if_empty:NTF \l_stex_renamedecl_name_str {
4555     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4556       \l_stex_get_symbol_uri_str
4557     } }
4558   } {
4559     \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_name_str}{\l_stex
4560       \stex_debug:nn{renamedecl}{@~\l_stex_current_module_str ? \l_stex_renamedecl_name_str}
4561       \prop_set_eq:cc {l_stex_symdecl_
4562         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4563         _prop
4564       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
4565       \seq_set_eq:cc {l_stex_symdecl_
4566         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4567         _notations
4568       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
4569       \prop_put:cnx {l_stex_symdecl_
4570         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4571         _prop
4572       }{ name }{ \l_stex_renamedecl_name_str }
4573       \prop_put:cnx {l_stex_symdecl_
4574         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4575         _prop
4576       }{ module }{ \l_stex_current_module_str }
4577       \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
4578         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4579       }
4580       \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4581         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4582       } }
4583     }
4584     \stex_smsmode_do:
4585   }
4586

```

```

4587 \stex_deactivate_macro:Nn \assign {copymodules}
4588 \stex_deactivate_macro:Nn \renamedekl {copymodules}
4589 \stex_deactivate_macro:Nn \donotcopy {copymodules}
4590
4591

```

## 30.2 The feature environment

structural@feature

```

4592 <@@=stex_features>
4593
4594 \NewDocumentEnvironment{structural_feature_module}{ m m m }{
4595   \stex_if_in_module:F {
4596     \msg_set:nnn{stex}{error/nomodule}{
4597       Structural~Feature~has~to~occur~in~a~module:\\
4598       Feature~#2~of~type~#1\\
4599       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
4600     }
4601     \msg_error:nn{stex}{error/nomodule}
4602   }
4603
4604   \str_set_eq:NN \l_stex_feature_parent_str \l_stex_current_module_str
4605
4606   \stex_module_setup:nn{meta=NONE}{#2 - #1}
4607
4608   \stex_if_do_html:T {
4609     \begin{stex_annotate_env}{ feature:#1 }{\l_stex_feature_parent_str ? #2 - #1}
4610     \stex_annotate_invisible:nnn{header}{\{ #3 }
4611   }
4612 }{
4613   \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
4614   \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
4615   \stex_debug:nn{features}{
4616     Feature: \l_stex_last_feature_str
4617   }
4618   \stex_if_do_html:T {
4619     \end{stex_annotate_env}
4620   }
4621 }

```

## 30.3 Structure

structure

```

4622 <@@=stex_structures>
4623 \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
4624   \prop_if_exist:cF {c_stex_module_ \l_stex_current_module_str _structures}{
4625     \prop_new:c {c_stex_module_ \l_stex_current_module_str _structures}
4626   }
4627   \prop_gput:cxn{c_stex_module_ \l_stex_current_module_str _structures}
4628   {#1}{#2}
4629 }
4630

```

```

4631 \keys_define:nn { stex / features / structure } {
4632   name          .str_set_x:N = \l__stex_structures_name_str ,
4633 }
4634
4635 \cs_new_protected:Nn \__stex_structures_structure_args:n {
4636   \str_clear:N \l__stex_structures_name_str
4637   \keys_set:nn { stex / features / structure } { #1 }
4638 }
4639
4640 \NewDocumentEnvironment{mathstructure}{m O{}}{
4641   \__stex_structures_structure_args:n { #2 }
4642   \str_if_empty:NT \l__stex_structures_name_str {
4643     \str_set:Nx \l__stex_structures_name_str { #1 }
4644   }
4645   \stex_suppress_html:n {
4646     \bool_set_true:N \l_stex_symdecl_make_macro_bool
4647     \exp_args:Nx \stex_symdecl_do:nn {
4648       name = \l__stex_structures_name_str ,
4649       def = {\STEXsymbol{module-type}}{
4650         \STEXInternalTermMathOMSiiii {
4651           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4652             { ns } ?
4653           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4654             { name } / \l__stex_structures_name_str - structure
4655         }{}{0}{}
4656       }}
4657     }{ #1 }
4658   }
4659   \exp_args:Nnnx
4660   \begin{structural_feature_module}{ structure }
4661     { \l__stex_structures_name_str }{}
4662   \stex_smsmode_do:
4663 }{
4664   \end{structural_feature_module}
4665   \stex_reset_up_to_module:n \l_stex_last_feature_str
4666   \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
4667   \seq_clear:N \l_tmpa_seq
4668   \seq_map_inline:Nn \l_stex_collect_imports_seq {
4669     \seq_map_inline:cn{c_stex_module_##1_constants}{
4670       \seq_put_right:Nn \l_tmpa_seq { ##1 ? ####1 }
4671     }
4672   }
4673   \exp_args:Nnno
4674   \prop_gput:cnn {c_stex_module_ \l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
4675   \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
4676   \stex_add_structure_to_current_module:nn
4677     \l__stex_structures_name_str
4678     \l_stex_last_feature_str
4679
4680   \stex_execute_in_module:x {
4681     \tl_set:cn { #1 }{
4682       \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures
4683     }
4684   }

```

```

4685 }
4686
4687 \cs_new:Nn \stex_invoke_structure:nn {
4688   \stex_invoke_symbol:n { #1?#2 }
4689 }
4690
4691 \cs_new_protected:Nn \stex_get_structure:n {
4692   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4693     \tl_set:Nn \l_tmpa_tl { #1 }
4694     \__stex_structures_get_from_cs:
4695   }{
4696     \cs_if_exist:cTF { #1 }{
4697       \cs_set_eq:Nc \l_tmpa_cs { #1 }
4698       \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
4699       \str_if_empty:NTF \l_tmpa_str {
4700         \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs} \stex_invoke_structure:nn {
4701           \__stex_structures_get_from_cs:
4702         }{
4703           \__stex_structures_get_from_string:n { #1 }
4704         }
4705       }{
4706         \__stex_structures_get_from_string:n { #1 }
4707       }
4708     }{
4709       \__stex_structures_get_from_string:n { #1 }
4710     }
4711   }
4712 }
4713
4714 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
4715   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4716     { \tl_tail:N \l_tmpa_tl }
4717   \str_set:Nx \l_tmpa_str {
4718     \exp_after:wN \use_i:nn \l_tmpa_tl
4719   }
4720   \str_set:Nx \l_tmpb_str {
4721     \exp_after:wN \use_ii:nn \l_tmpa_tl
4722   }
4723   \str_set:Nx \l_stex_get_structure_str {
4724     \l_tmpa_str ? \l_tmpb_str
4725   }
4726   \str_set:Nx \l_stex_get_structure_module_str {
4727     \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
4728   }
4729 }
4730
4731 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
4732   \tl_set:Nn \l_tmpa_tl {
4733     \msg_error:nnn{stex}{error/unknownstructure}{#1}
4734   }
4735   \str_set:Nn \l_tmpa_str { #1 }
4736   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4737
4738   \seq_map_inline:Nn \l_stex_all_modules_seq {

```

```

4739 \prop_if_exist:cT {c_stex_module_##1_structures} {
4740 \prop_map_inline:cn {c_stex_module_##1_structures} {
4741 \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
4742 \prop_map_break:n{\seq_map_break:n{
4743 \tl_set:Nn \l_tmpa_tl {
4744 \str_set:Nn \l_stex_get_structure_str {##1?####1}
4745 \str_set:Nn \l_stex_get_structure_module_str {####2}
4746 }
4747 }}
4748 }
4749 }
4750 }
4751 }
4752 \l_tmpa_tl
4753 }

```

**\instantiate**

```

4754
4755 \keys_define:nn { stex / instantiate } {
4756 name .str_set_x:N = \l__stex_structures_name_str
4757 }
4758 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
4759 \str_clear:N \l__stex_structures_name_str
4760 \keys_set:nn { stex / instantiate } { #1 }
4761 }
4762
4763 \NewDocumentCommand \instantiate {m O{} m m O{}}{
4764 \beginingroup
4765 \stex_get_structure:n {#3}
4766 \__stex_structures_instantiate_args:n { #2 }
4767 \str_if_empty:NT \l__stex_structures_name_str {
4768 \str_set:Nn \l__stex_structures_name_str { #1 }
4769 }
4770 \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4771 \seq_clear:N \l__stex_structures_fields_seq
4772 \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4773 \seq_map_inline:Nn \l_stex_collect_imports_seq {
4774 \seq_map_inline:cn {c_stex_module_##1_constants}{
4775 \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4776 }
4777 }
4778
4779 \tl_if_empty:nF{#5}{
4780 \seq_set_split:Nnn \l_tmpa_seq , {#5}
4781 \prop_clear:N \l_tmpa_prop
4782 \seq_map_inline:Nn \l_tmpa_seq {
4783 \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4784 \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4785 \msg_error:nnn{stex}{error/keyval}{##1}
4786 }
4787 \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
4788 \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4789 \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
4790 \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}

```

```

4791     \exp_args:Nxx \str_if_eq:nnF
4792     {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4793     {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}{
4794     \msg_error:nnxxxx{stex}{error/incompatible}
4795     {l__stex_structures_dom_str}
4796     {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4797     {\l_stex_get_symbol_uri_str}
4798     {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}
4799     }
4800     \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
4801   }
4802 }
4803
4804 \seq_map_inline:Nn \l__stex_structures_fields_seq {
4805   \str_set:Nx \l_tmpa_str {field:\l__stex_structures_name_str . \prop_item:cn {l_stex_sy
4806   \stex_debug:nn{instantiate}{Field~\l_tmpa_str :~##1}
4807
4808   \stex_add_constant_to_current_module:n {\l_tmpa_str}
4809   \stex_execute_in_module:x {
4810     \prop_set_from_keyval:cn { l_stex_symdecl_ \l_stex_current_module_str?\l_tmpa_str _p
4811     name   = \l_tmpa_str ,
4812     args   = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4813     arity  = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4814     assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4815   }
4816   \seq_clear:c {l_stex_symdecl\l_stex_current_module_str?\l_tmpa_str _notations}
4817 }
4818
4819 \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4820   \stex_find_notation:nn{##1}{}
4821   \stex_execute_in_module:x {
4822     \seq_put_right:cn {l_stex_symdecl\l_stex_current_module_str?\l_tmpa_str _notation
4823   }
4824
4825   \stex_copy_control_sequence_ii:ccN
4826   {stex_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_notation_
4827   {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4828   \l_tmpa_tl
4829   \exp_args:No \stex_execute_in_module:n \l_tmpa_tl
4830
4831
4832   \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
4833     \tl_set_eq:Nc \l_tmpa_cs {stex_op_notation_##1\c_hash_str \l_stex_notation_variant
4834     \stex_execute_in_module:x {
4835       \tl_set:cn
4836       {stex_op_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_not
4837       { \exp_args:No \exp_not:n \l_tmpa_cs}
4838     }
4839   }
4840
4841 }
4842
4843 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\l_stex_cur
4844 }

```



```

4845
4846 \stex_execute_in_module:x {
4847   \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
4848     domain = \l_stex_get_structure_module_str ,
4849     \prop_to_keyval:N \l_tmpa_prop
4850   }
4851   \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
4852 }
4853 \stex_debug:nn{instantiate}{
4854   Instance~\l_stex_current_module_str?\l__stex_structures_name_str \
4855   \prop_to_keyval:N \l_tmpa_prop
4856 }
4857 \exp_args:Nxx \stex_symdecl_do:nn {
4858   type={\STEXsymbol{module-type}{
4859     \STEXInternalTermMathOMSiiii {
4860       \l_stex_get_structure_module_str
4861     }{}{0}{}
4862   }}
4863 }{\l__stex_structures_name_str}
4864 % {
4865   \str_set:Nx \l_stex_get_symbol_uri_str {\l_stex_current_module_str?\l__stex_structures
4866   \tl_set:Nn \l_stex_notation_after_do_tl {\__stex_notation_final:}
4867   \stex_notation_do:nnnnn{}{}{}{}{\comp{#4}}
4868 % }
4869 %\exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}
4870 \endgroup
4871 \stex_smsmode_do:\ignorespacesandpars
4872 }
4873
4874 \cs_new_protected:Nn \stex_symbol_or_var:n {
4875   \cs_if_exist:cTF{#1}{
4876     \cs_set_eq:Nc \l_tmpa_tl { #1 }
4877     \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4878     \str_if_empty:NTF \l_tmpa_str {
4879       \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
4880       \stex_invoke_variable:n {
4881         \bool_set_true:N \l_stex_symbol_or_var_bool
4882         \bool_set_false:N \l_stex_instance_or_symbol_bool
4883         \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4884         \tl_set:Nx \l_tmpa_tl {\exp_after:wN \use:n \l_tmpa_tl}
4885         \str_set:Nx \l_stex_get_symbol_uri_str {
4886           \exp_after:wN \use:n \l_tmpa_tl
4887         }
4888       }{ % TODO \stex_invoke_varinstance:n
4889         \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl } \stex_invoke_varinstance:n {
4890           \bool_set_true:N \l_stex_symbol_or_var_bool
4891           \bool_set_true:N \l_stex_instance_or_symbol_bool
4892           \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4893           \tl_set:Nx \l_tmpa_tl {\exp_after:wN \use:n \l_tmpa_tl}
4894           \str_set:Nx \l_stex_get_symbol_uri_str {
4895             \exp_after:wN \use:n \l_tmpa_tl
4896           }
4897         }{
4898           \bool_set_false:N \l_stex_symbol_or_var_bool

```

```

4899         \stex_get_symbol:n{#1}
4900     }
4901 }
4902 }{
4903     \__stex_structures_symbolorvar_from_string:n{ #1 }
4904 }
4905 }{
4906     \__stex_structures_symbolorvar_from_string:n{ #1 }
4907 }
4908 }
4909
4910 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
4911     \prop_if_exist:cTF {l_stex_variable_#1 _prop}{
4912         \bool_set_true:N \l_stex_symbol_or_var_bool
4913         \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
4914     }{
4915         \bool_set_false:N \l_stex_symbol_or_var_bool
4916         \stex_get_symbol:n{#1}
4917     }
4918 }
4919
4920 \keys_define:nn { stex / varinstantiate } {
4921     name          .str_set_x:N = \l__stex_structures_name_str,
4922     bind          .choices:nn =
4923         {forall,exists}
4924         {\str_set:Nx \l__stex_structures_bind_str {\l_keys_choice_tl}}
4925 }
4926 }
4927 \cs_new_protected:Nn \__stex_structures_varinstantiate_args:n {
4928     \str_clear:N \l__stex_structures_name_str
4929     \str_clear:N \l__stex_structures_bind_str
4930     \keys_set:nn { stex / varinstantiate } { #1 }
4931 }
4932
4933 \NewDocumentCommand \varinstantiate {m O{}} m m O{}{{
4934     \begingroup
4935         \stex_get_structure:n {#3}
4936         \__stex_structures_varinstantiate_args:n { #2 }
4937         \str_if_empty:NT \l__stex_structures_name_str {
4938             \str_set:Nn \l__stex_structures_name_str { #1 }
4939         }
4940         \stex_if_do_html:TF{
4941             \stex_annotate:nnn{varinstance}{\l__stex_structures_name_str}
4942         }{\use:n}
4943         {
4944             \stex_if_do_html:T{
4945                 \stex_annotate_invisible:nnn{domain}{\l_stex_get_structure_module_str}{}
4946             }
4947             \seq_clear:N \l__stex_structures_fields_seq
4948             \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4949             \seq_map_inline:Nn \l_stex_collect_imports_seq {
4950                 \seq_map_inline:cn {c_stex_module_##1_constants}{
4951                     \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4952                 }

```

```

4953 }
4954 \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4955 \prop_clear:N \l_tmpa_prop
4956 \tl_if_empty:nF {#5} {
4957   \seq_set_split:Nnn \l_tmpa_seq , {#5}
4958   \seq_map_inline:Nn \l_tmpa_seq {
4959     \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4960     \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4961       \msg_error:nnn{stex}{error/keyval}{##1}
4962     }
4963     \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_structures_dom_str
4964     \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4965     \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_in_seq
4966     \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
4967     \stex_if_do_html:T{
4968       \stex_annotate:nnn{assign}{\l__stex_structures_dom_str,
4969         \bool_if:NTF\l_stex_symbol_or_var_bool{var://}{}\l_stex_get_symbol_uri_str}{}}
4970     }
4971     \bool_if:NTF \l_stex_symbol_or_var_bool {
4972       \exp_args:Nxx \str_if_eq:nnF
4973         {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4974         {\prop_item:cn{l_stex_variable\l_stex_get_symbol_uri_str _prop}{args}}{\
4975       \msg_error:nnxxx{stex}{error/incompatible}
4976       {\l__stex_structures_dom_str}
4977       {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4978       {\l_stex_get_symbol_uri_str}
4979       {\prop_item:cn{l_stex_variable\l_stex_get_symbol_uri_str _prop}{args}}}
4980     }
4981     \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n {
4982   }}{
4983     \exp_args:Nxx \str_if_eq:nnF
4984       {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4985       {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}{\
4986     \msg_error:nnxxx{stex}{error/incompatible}
4987     {\l__stex_structures_dom_str}
4988     {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4989     {\l_stex_get_symbol_uri_str}
4990     {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}}
4991   }
4992   \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n {
4993   }}
4994 }
4995 }
4996 \tl_gclear:N \g__stex_structures_aftergroup_tl
4997 \seq_map_inline:Nn \l__stex_structures_fields_seq {
4998   \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdecl\l__stex_structures_fields_seq #1} \l_stex_get_symbol_uri_str}
4999   \stex_debug:nn{varinstantiate}{Field~\l_tmpa_str :~##1}
5000   \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
5001     \stex_find_notation:nn{##1}{}
5002     \cs_gset_eq:cc{g__stex_structures_tmpa\l_tmpa_str _cs}
5003       {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5004     \stex_debug:nn{varinstantiate}{Notation:~\cs_meaning:c{g__stex_structures_tmpa\l_tmpa_str _cs}}
5005     \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
5006       \cs_gset_eq:cc {g__stex_structures_tmpa_op\l_tmpa_str _cs}

```

```

5007         {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5008         \stex_debug:nn{varinstantiate}{Operator-Notation:~\cs_meaning:c{g__stex_struct
5009     }
5010 }
5011
5012 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
5013     \prop_set_from_keyval:cn { l_stex_variable_ \l_tmpa_str _prop}{
5014         name = \l_tmpa_str ,
5015         args = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
5016         arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
5017         assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
5018     }
5019     \cs_set_eq:cc {stex_var_notation_\l_tmpa_str _cs}
5020     {g__stex_structures_tmpa_\l_tmpa_str _cs}
5021     \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str _cs}
5022     {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
5023 }
5024 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_inv
5025 }
5026 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
5027     \prop_set_from_keyval:cn {l_stex_varinstance_\l__stex_structures_name_str _prop }{
5028         domain = \l_stex_get_structure_module_str ,
5029         \prop_to_keyval:N \l_tmpa_prop
5030     }
5031     \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
5032     \tl_set:cn {l_stex_varinstance_\l__stex_structures_name_str _op_tl}{
5033         \exp_args:Nnx \exp_not:N \use:nn {
5034             \str_set:Nn \exp_not:N \STEXInternalCurrentSymbolStr {var://\l__stex_structures_
5035             \_stex_term_omv:nn {var://\l__stex_structures_name_str}{
5036                 \exp_not:n{
5037                     \_varcomp{#4}
5038                 }
5039             }
5040         }{
5041             \exp_not:n{\_stex_reset:N \STEXInternalCurrentSymbolStr}
5042         }
5043     }
5044 }
5045 }
5046 \stex_debug:nn{varinstantiate}{\expandafter\detokenize\expandafter{g__stex_structures_a
5047 \aftergroup\g__stex_structures_aftergroup_tl
5048 \endgroup
5049 \stex_smsmode_do:\ignorespacesandpars
5050 }
5051
5052 \cs_new_protected:Nn \stex_invoke_instance:n {
5053     \peek_charcode_remove:NTF ! {
5054         \stex_invoke_symbol:n{#1}
5055     }{
5056         \_stex_invoke_instance:nn {#1}
5057     }
5058 }
5059
5060

```

```

5061 \cs_new_protected:Nn \stex_invoke_varinstance:n {
5062   \peek_charcode_remove:NTF ! {
5063     \exp_args:Nnx \use:nn {
5064       \def\comp{\_varcomp}
5065       \use:c{l_stex_varinstance_#1_op_tl}
5066     }{
5067       \_stex_reset:N \comp
5068     }
5069   }{
5070     \_stex_invoke_varinstance:nn {#1}
5071   }
5072 }
5073
5074 \cs_new_protected:Nn \_stex_invoke_instance:nn {
5075   \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
5076     \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
5077   }{
5078     \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
5079     \msg_error:nnxxx{stex}{error/unknownfield}{#2}{#1}{
5080       \prop_to_keyval:N \l_tmpa_prop
5081     }
5082   }
5083 }
5084
5085 \cs_new_protected:Nn \_stex_invoke_varinstance:nn {
5086   \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
5087     \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
5088     \l_tmpa_tl
5089   }{
5090     \msg_error:nnnnn{stex}{error/unknownfield}{#2}{#1}{}
5091   }
5092 }

```

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

`\stex_invoke_structure:nnn`

```

5093 % #1: URI of the instance
5094 % #2: URI of the instantiated module
5095 \cs_new_protected:Nn \stex_invoke_structure:nnn {
5096   \tl_if_empty:nTF{ #3 }{
5097     \prop_set_eq:Nc \l__stex_structures_structure_prop {
5098       c_stex_feature_ #2 _prop
5099     }
5100     \tl_clear:N \l_tmpa_tl
5101     \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
5102     \seq_map_inline:Nn \l_tmpa_seq {
5103       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
5104       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
5105       \cs_if_exist:cT {
5106         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
5107       }{
5108         \tl_if_empty:NF \l_tmpa_tl {
5109           \tl_put_right:Nn \l_tmpa_tl {,}
5110         }

```

```

5111         \tl_put_right:Nx \l_tmpa_tl {
5112             \stex_invoke_symbol:n {#1/\l_tmpa_str}!
5113         }
5114     }
5115 }
5116 \exp_args:No \mathstruct \l_tmpa_tl
5117 }{
5118     \stex_invoke_symbol:n{#1/#3}
5119 }
5120 }

```

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

```

5121 </package>

```

## Chapter 31

# STEX -Statements Implementation

```
5122 <*package>
5123
5124 %%%%%%%%%%% features.dtx %%%%%%%%%%%
5125
5126 <@@=stex_statements>

Warnings and error messages

5127

\titleemph

5128 \def\titleemph#1{\textbf{#1}}

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

### 31.1 Definitions

#### definiendum

```
5129 \keys_define:nn {stex / definiendum }{
5130   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
5131   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
5132   root     .str_set_x:N    = \l__stex_statements_definiendum_root_str,
5133   gfa      .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
5134 }
5135 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
5136   \str_clear:N \l__stex_statements_definiendum_root_str
5137   \tl_clear:N \l__stex_statements_definiendum_post_tl
5138   \str_clear:N \l__stex_statements_definiendum_gfa_str
5139   \keys_set:nn { stex / definiendum }{ #1 }
5140 }
5141 \NewDocumentCommand \definiendum { O{} m m } {
5142   \__stex_statements_definiendum_args:n { #1 }
5143   \stex_get_symbol:n { #2 }
5144   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5145   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
5146     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```

5147     \tl_set:Nn \l_tmpa_tl { #3 }
5148   } {
5149     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
5150     \tl_set:Nn \l_tmpa_tl {
5151       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
5152     }
5153   }
5154 } {
5155   \tl_set:Nn \l_tmpa_tl { #3 }
5156 }
5157
5158 % TODO root
5159 \stex_html_backend:TF {
5160   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
5161 } {
5162   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
5163 }
5164 }
5165 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

(End definition for definiendum. This function is documented on page 42.)

#### definame

```

5166
5167 \NewDocumentCommand \definame { 0{ } m } {
5168   \__stex_statements_definiendum_args:n { #1 }
5169   % TODO: root
5170   \stex_get_symbol:n { #2 }
5171   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5172   \str_set:Nx \l_tmpa_str {
5173     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5174   }
5175   \str_replace_all:Nnn \l_tmpa_str {-} {~}
5176   \stex_html_backend:TF {
5177     \stex_if_do_html:T {
5178       \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5179         \l_tmpa_str\l__stex_statements_definiendum_post_tl
5180       }
5181     }
5182   } {
5183     \exp_args:Nnx \defemph@uri {
5184       \l_tmpa_str\l__stex_statements_definiendum_post_tl
5185     } { \l_stex_get_symbol_uri_str }
5186   }
5187 }
5188 \stex_deactivate_macro:Nn \definame {definition~environments}
5189
5190 \NewDocumentCommand \Definame { 0{ } m } {
5191   \__stex_statements_definiendum_args:n { #1 }
5192   \stex_get_symbol:n { #2 }
5193   \str_set:Nx \l_tmpa_str {
5194     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5195   }
5196   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}

```



```

5197 \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5198 \stex_html_backend:TF {
5199   \stex_if_do_html:T {
5200     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5201       \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5202     }
5203   }
5204 } {
5205   \exp_args:Nnx \defemph@uri {
5206     \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5207   } { \l_stex_get_symbol_uri_str }
5208 }
5209 }
5210 \stex_deactivate_macro:Nn \Definame {definition-environments}
5211
5212 \NewDocumentCommand \premise { m }{
5213   \noindent\stex_annotate:nnn{ premise }{}{\ignorespaces #1 }
5214 }
5215 \NewDocumentCommand \conclusion { m }{
5216   \noindent\stex_annotate:nnn{ conclusion }{}{\ignorespaces #1 }
5217 }
5218 \NewDocumentCommand \definiens { 0{} m }{
5219   \str_clear:N \l_stex_get_symbol_uri_str
5220   \tl_if_empty:nF {#1} {
5221     \stex_get_symbol:n { #1 }
5222   }
5223   \str_if_empty:NT \l_stex_get_symbol_uri_str {
5224     \int_compare:nNnTF {\clist_count:N \l__stex_statements_sdefinition_for_clist} = 1 {
5225       \str_set:Nx \l_stex_get_symbol_uri_str {\clist_item:Nn \l__stex_statements_sdefinition
5226     }{
5227       % TODO throw error
5228     }
5229   }
5230   \str_if_eq:eeT {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{module}}
5231   {\l_stex_current_module_str}{
5232     \str_if_eq:eeF {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defin
5233   }{true}{
5234     \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5235     \exp_args:Nx \stex_add_to_current_module:n {
5236       \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5237     }
5238   }
5239 }
5240 \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
5241 }
5242
5243 \NewDocumentCommand \varbindforall {m}{
5244   \stex_symbol_or_var:n {#1}
5245   \bool_if:NTF\l_stex_symbol_or_var_bool{
5246     \stex_if_do_html:T {
5247       \stex_annotate_invisible:nnn {bindtype}{forall,\l_stex_get_symbol_uri_str}{}
5248     }
5249   }{
5250     % todo throw error

```

```

5251 }
5252 }
5253
5254 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
5255 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
5256 \stex_deactivate_macro:Nn \definiens {definition~environments}
5257 \stex_deactivate_macro:Nn \varbindforall {definition~or~assertion~environments}
5258

```

(End definition for definame. This function is documented on page [42](#).)

#### sdefinition

```

5259
5260 \keys_define:nn {stex / sdefinition }{
5261   type      .str_set_x:N = \sdefinitiontype,
5262   id        .str_set_x:N = \sdefinitionid,
5263   name      .str_set_x:N = \sdefinitionname,
5264   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5265   title     .tl_set:N     = \sdefinitiontitle
5266 }
5267 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
5268   \str_clear:N \sdefinitiontype
5269   \str_clear:N \sdefinitionid
5270   \str_clear:N \sdefinitionname
5271   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5272   \tl_clear:N \sdefinitiontitle
5273   \keys_set:nn { stex / sdefinition }{ #1 }
5274 }
5275
5276 \NewDocumentEnvironment{sdefinition}{0{}}{
5277   \__stex_statements_sdefinition_args:n{ #1 }
5278   \stex_reactivate_macro:N \definiendum
5279   \stex_reactivate_macro:N \definame
5280   \stex_reactivate_macro:N \Definame
5281   \stex_reactivate_macro:N \premise
5282   \stex_reactivate_macro:N \definiens
5283   \stex_reactivate_macro:N \varbindforall
5284   \stex_if_smsmode:F{
5285     \seq_clear:N \l_tmpb_seq
5286     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5287       \tl_if_empty:nF{ ##1 }{
5288         \stex_get_symbol:n { ##1 }
5289         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5290           \l_stex_get_symbol_uri_str
5291         }
5292       }
5293     }
5294     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5295     \exp_args:Nnnx
5296     \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
5297     \str_if_empty:NF \sdefinitiontype {
5298       \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{ }
5299     }
5300     \str_if_empty:NF \sdefinitionname {

```

```

5301     \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5302   }
5303   \clist_set:No \l_tmpa_clist \sdefinitiontype
5304   \tl_clear:N \l_tmpa_tl
5305   \clist_map_inline:Nn \l_tmpa_clist {
5306     \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
5307       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
5308     }
5309   }
5310   \tl_if_empty:NTF \l_tmpa_tl {
5311     \__stex_statements_sdefinition_start:
5312   }{
5313     \l_tmpa_tl
5314   }
5315 }
5316 \stex_ref_new_doc_target:n \sdefinitionid
5317 \stex_smsmode_do:
5318 ){
5319   \stex_suppress_html:n {
5320     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5321   }
5322   \stex_if_smsmode:F {
5323     \clist_set:No \l_tmpa_clist \sdefinitiontype
5324     \tl_clear:N \l_tmpa_tl
5325     \clist_map_inline:Nn \l_tmpa_clist {
5326       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
5327         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
5328       }
5329     }
5330     \tl_if_empty:NTF \l_tmpa_tl {
5331       \__stex_statements_sdefinition_end:
5332     }{
5333       \l_tmpa_tl
5334     }
5335     \end{stex_annotate_env}
5336   }
5337 }

```

### **\stexpatchdefinition**

```

5338 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
5339   \stex_par:\noindent\titleemph{Definition}\tl_if_empty:NF \sdefinitiontitle {
5340     ~(\sdefinitiontitle)
5341   }~}
5342 }
5343 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\stex_par:\medskip}
5344
5345 \newcommand\stexpatchdefinition[3]{} {
5346   \str_set:Nx \l_tmpa_str{ #1 }
5347   \str_if_empty:NTF \l_tmpa_str {
5348     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
5349     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
5350   }{
5351     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
5352     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }

```

```

5353     }
5354 }

```

(End definition for `\stexpatchdefinition`. This function is documented on page 49.)

`\inlinedef` inline:

```

5355 \keys_define:nn {stex / inlinedef }{
5356   type      .str_set_x:N = \sdefinitiontype,
5357   id        .str_set_x:N = \sdefinitionid,
5358   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5359   name      .str_set_x:N = \sdefinitionname
5360 }
5361 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
5362   \str_clear:N \sdefinitiontype
5363   \str_clear:N \sdefinitionid
5364   \str_clear:N \sdefinitionname
5365   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5366   \keys_set:nn { stex / inlinedef }{ #1 }
5367 }
5368 \NewDocumentCommand \inlinedef { 0{} m } {
5369   \begingroup
5370   \__stex_statements_inlinedef_args:n{ #1 }
5371   \stex_reactivate_macro:N \definiendum
5372   \stex_reactivate_macro:N \definame
5373   \stex_reactivate_macro:N \Definame
5374   \stex_reactivate_macro:N \premise
5375   \stex_reactivate_macro:N \definiens
5376   \stex_reactivate_macro:N \varbindforall
5377   \stex_ref_new_doc_target:n \sdefinitionid
5378   \stex_if_smsmode:TF{\stex_suppress_html:n {
5379     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5380   }}{
5381     \seq_clear:N \l_tmpb_seq
5382     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5383       \tl_if_empty:nF{ ##1 }{
5384         \stex_get_symbol:n { ##1 }
5385         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5386           \l_stex_get_symbol_uri_str
5387         }
5388       }
5389     }
5390     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5391     \exp_args:Nnx
5392     \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpb_seq {,}}{
5393       \str_if_empty:NF \sdefinitiontype {
5394         \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5395       }
5396       #2
5397       \str_if_empty:NF \sdefinitionname {
5398         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sdefinitionname}}
5399         \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5400       }
5401     }
5402   }

```



```

5450 \clist_map_inline:Nn \l_tmpa_clist {
5451   \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
5452     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
5453   }
5454 }
5455 \tl_if_empty:NTF \l_tmpa_tl {
5456   \__stex_statements_sassertion_start:
5457 }{
5458   \l_tmpa_tl
5459 }
5460 }
5461 \str_if_empty:NTF \sassertionid {
5462   \str_if_empty:NF \sassertionname {
5463     \stex_ref_new_doc_target:n {}
5464   }
5465 } {
5466   \stex_ref_new_doc_target:n \sassertionid
5467 }
5468 \stex_smsmode_do:
5469 ){
5470   \str_if_empty:NF \sassertionname {
5471     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5472     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5473   }
5474   \stex_if_smsmode:F {
5475     \clist_set:No \l_tmpa_clist \sassertiontype
5476     \tl_clear:N \l_tmpa_tl
5477     \clist_map_inline:Nn \l_tmpa_clist {
5478       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
5479         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
5480       }
5481     }
5482     \tl_if_empty:NTF \l_tmpa_tl {
5483       \__stex_statements_sassertion_end:
5484     }{
5485       \l_tmpa_tl
5486     }
5487     \end{stex_annotate_env}
5488   }
5489 }

```

## \stexpatchassertion

```

5490
5491 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
5492   \stex_par:\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
5493     (\sassertiontitle)
5494   }~}
5495 }
5496 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\stex_par:\medskip}
5497
5498 \newcommand\stexpatchassertion[3] [] {
5499   \str_set:Nx \l_tmpa_str{ #1 }
5500   \str_if_empty:NTF \l_tmpa_str {
5501     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }

```

```

5502     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
5503   }{
5504     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
5505     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
5506   }
5507 }

```

(End definition for `\stexpatchassertion`. This function is documented on page 49.)

`\inlineass` inline:

```

5508 \keys_define:nn {stex / inlineass }{
5509   type      .str_set_x:N = \sassertiontype,
5510   id        .str_set_x:N = \sassertionid,
5511   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
5512   name      .str_set_x:N = \sassertionname
5513 }
5514 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
5515   \str_clear:N \sassertiontype
5516   \str_clear:N \sassertionid
5517   \str_clear:N \sassertionname
5518   \clist_clear:N \l__stex_statements_sassertion_for_clist
5519   \keys_set:nn { stex / inlineass }{ #1 }
5520 }
5521 \NewDocumentCommand \inlineass { 0{} m } {
5522   \beginngroup
5523   \stex_reactivate_macro:N \premise
5524   \stex_reactivate_macro:N \conclusion
5525   \stex_reactivate_macro:N \varbindforall
5526   \__stex_statements_inlineass_args:n{ #1 }
5527   \str_if_empty:NTF \sassertionid {
5528     \str_if_empty:NF \sassertionname {
5529       \stex_ref_new_doc_target:n {}
5530     }
5531   } {
5532     \stex_ref_new_doc_target:n \sassertionid
5533   }
5534
5535   \stex_if_smsmode:TF{
5536     \str_if_empty:NF \sassertionname {
5537       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5538       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5539     }
5540   }{
5541     \seq_clear:N \l_tmpb_seq
5542     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5543       \tl_if_empty:nF{ ##1 }{
5544         \stex_get_symbol:n { ##1 }
5545         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5546           \l_stex_get_symbol_uri_str
5547         }
5548       }
5549     }
5550     \exp_args:Nnx
5551     \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpb_seq {},,}{

```

```

5552     \str_if_empty:NF \sassertiontype {
5553       \stex_annotate_invisible:nnn{typestrings}{\sassertiontype}{ }
5554     }
5555     #2
5556     \str_if_empty:NF \sassertionname {
5557       \stex_suppress_html:n{\stex_symdecl_do:nn{ }{\sassertionname}}
5558       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5559       \stex_annotate_invisible:nnn{statementname}{\sassertionname}{ }
5560     }
5561   }
5562 }
5563 \endgroup
5564 \stex_smsmode_do:
5565 }

```

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

## 31.3 Examples

`sexample`

```

5566
5567 \keys_define:nn {stex / sexample }{
5568   type      .str_set_x:N = \exampletype,
5569   id        .str_set_x:N = \sexampleid,
5570   title     .tl_set:N   = \sexamplename,
5571   name      .str_set_x:N = \sexamplename ,
5572   for       .clist_set:N = \l__stex_statements_sexample_for_clist,
5573 }
5574 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
5575   \str_clear:N \sexampletype
5576   \str_clear:N \sexampleid
5577   \str_clear:N \sexamplename
5578   \tl_clear:N \sexamplename
5579   \clist_clear:N \l__stex_statements_sexample_for_clist
5580   \keys_set:nn { stex / sexample }{ #1 }
5581 }
5582
5583 \NewDocumentEnvironment{sexample}{0}{ }{
5584   \__stex_statements_sexample_args:n{ #1 }
5585   \stex_reactivate_macro:N \premise
5586   \stex_reactivate_macro:N \conclusion
5587   \stex_if_smsmode:F {
5588     \seq_clear:N \l_tmpb_seq
5589     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5590       \tl_if_empty:nF{ ##1 }{
5591         \stex_get_symbol:n { ##1 }
5592         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5593           \l_stex_get_symbol_uri_str
5594         }
5595       }
5596     }
5597     \exp_args:Nnnx
5598     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpb_seq {,}}

```



```

5599 \str_if_empty:NF \sexamplotype {
5600   \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{}
5601 }
5602 \str_if_empty:NF \sexamplename {
5603   \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
5604 }
5605 \clist_set:No \l_tmpa_clist \sexamplotype
5606 \tl_clear:N \l_tmpa_tl
5607 \clist_map_inline:Nn \l_tmpa_clist {
5608   \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
5609     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
5610   }
5611 }
5612 \tl_if_empty:NTF \l_tmpa_tl {
5613   \__stex_statements_sexample_start:
5614 }{
5615   \l_tmpa_tl
5616 }
5617 }
5618 \str_if_empty:NF \sexampleid {
5619   \stex_ref_new_doc_target:n \sexampleid
5620 }
5621 \stex_smsmode_do:
5622 }{
5623   \str_if_empty:NF \sexamplename {
5624     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5625   }
5626   \stex_if_smsmode:F {
5627     \clist_set:No \l_tmpa_clist \sexamplotype
5628     \tl_clear:N \l_tmpa_tl
5629     \clist_map_inline:Nn \l_tmpa_clist {
5630       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
5631         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
5632       }
5633     }
5634     \tl_if_empty:NTF \l_tmpa_tl {
5635       \__stex_statements_sexample_end:
5636     }{
5637       \l_tmpa_tl
5638     }
5639     \end{stex_annotate_env}
5640   }
5641 }

```

**\stexpatchexample**

```

5642
5643 \cs_new_protected:Nn \__stex_statements_sexample_start: {
5644   \stex_par:\noindent\titllemph{Example~\tl_if_empty:NF \sexampltitle {
5645     (\sexampltitle)
5646   }~}
5647 }
5648 \cs_new_protected:Nn \__stex_statements_sexample_end: {\stex_par:\medskip}
5649
5650 \newcommand\stexpatchexample[3]{} {

```

```

5651 \str_set:Nx \l_tmpa_str{ #1 }
5652 \str_if_empty:NTF \l_tmpa_str {
5653   \tl_set:Nn \__stex_statements_sexample_start: { #2 }
5654   \tl_set:Nn \__stex_statements_sexample_end: { #3 }
5655 }{
5656   \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
5657   \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
5658 }
5659 }

```

(End definition for `\stexpatchexample`. This function is documented on page 49.)

`\inlineex` inline:

```

5660 \keys_define:nn {stex / inlineex }{
5661   type      .str_set_x:N = \sexamplotype,
5662   id        .str_set_x:N = \sexampleid,
5663   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
5664   name      .str_set_x:N = \sexamplename
5665 }
5666 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
5667   \str_clear:N \sexamplotype
5668   \str_clear:N \sexampleid
5669   \str_clear:N \sexamplename
5670   \clist_clear:N \l__stex_statements_sexample_for_clist
5671   \keys_set:nn { stex / inlineex }{ #1 }
5672 }
5673 \NewDocumentCommand \inlineex { 0{} m } {
5674   \begingroup
5675   \stex_reactivate_macro:N \premise
5676   \stex_reactivate_macro:N \conclusion
5677   \__stex_statements_inlineex_args:n{ #1 }
5678   \str_if_empty:NF \sexampleid {
5679     \stex_ref_new_doc_target:n \sexampleid
5680   }
5681   \stex_if_smsmode:TF{
5682     \str_if_empty:NF \sexamplename {
5683       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5684     }
5685   }{
5686     \seq_clear:N \l_tmpb_seq
5687     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5688       \tl_if_empty:nF{ ##1 }{
5689         \stex_get_symbol:n { ##1 }
5690         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5691           \l_stex_get_symbol_uri_str
5692         }
5693       }
5694     }
5695     \exp_args:Nnx
5696     \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpb_seq {,}}{
5697       \str_if_empty:NF \sexamplotype {
5698         \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{}
5699       }
5700       #2

```

```

5701     \str_if_empty:NF \sexamplename {
5702       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5703       \stex_annotate_invisible:nnn{statementname}{\sexamplename}{ }
5704     }
5705   }
5706 }
5707 \endgroup
5708 \stex_smsmode_do:
5709 }

```

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

## 31.4 Logical Paragraphs

`sparagraph`

```

5710 \keys_define:nn { stex / sparagraph } {
5711   id      .str_set:x:N = \sparagraphid ,
5712   title   .tl_set:N    = \l_stex_sparagraph_title_tl ,
5713   type    .str_set:x:N = \sparagraphtype ,
5714   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5715   from    .tl_set:N    = \sparagraphfrom ,
5716   to      .tl_set:N    = \sparagraphto ,
5717   start   .tl_set:N    = \l_stex_sparagraph_start_tl ,
5718   name    .str_set:N    = \sparagraphname ,
5719   imports .tl_set:N    = \l__stex_statements_sparagraph_imports_tl
5720 }
5721
5722 \cs_new_protected:Nn \stex_sparagraph_args:n {
5723   \tl_clear:N \l_stex_sparagraph_title_tl
5724   \tl_clear:N \sparagraphfrom
5725   \tl_clear:N \sparagraphto
5726   \tl_clear:N \l_stex_sparagraph_start_tl
5727   \tl_clear:N \l__stex_statements_sparagraph_imports_tl
5728   \str_clear:N \sparagraphid
5729   \str_clear:N \sparagraphtype
5730   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5731   \str_clear:N \sparagraphname
5732   \keys_set:nn { stex / sparagraph } { #1 }
5733 }
5734 \newif\if@in@omtext\@in@omtextfalse
5735
5736 \NewDocumentEnvironment {sparagraph} { 0{} } {
5737   \stex_sparagraph_args:n { #1 }
5738   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5739     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
5740   }{
5741     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
5742   }
5743   \@in@omtexttrue
5744   \stex_if_smsmode:F {
5745     \seq_clear:N \l_tmpb_seq
5746     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5747       \tl_if_empty:nF{ ##1 }{

```

```

5748     \stex_get_symbol:n { ##1 }
5749     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5750         \l_stex_get_symbol_uri_str
5751     }
5752 }
5753 }
5754 \exp_args:Nnnx
5755 \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}
5756 \str_if_empty:NF \sparagraphtype {
5757     \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
5758 }
5759 \str_if_empty:NF \sparagraphfrom {
5760     \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5761 }
5762 \str_if_empty:NF \sparagraphto {
5763     \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5764 }
5765 \str_if_empty:NF \sparagraphname {
5766     \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
5767 }
5768 \clist_set:No \l_tmpa_clist \sparagraphtype
5769 \tl_clear:N \l_tmpa_tl
5770 \clist_map_inline:Nn \sparagraphtype {
5771     \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
5772         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
5773     }
5774 }
5775 \stex_csl_to_imports:No \usemodule \l__stex_statements_sparagraph_imports_tl
5776 \tl_if_empty:NTF \l_tmpa_tl {
5777     \__stex_statements_sparagraph_start:
5778 }{
5779     \l_tmpa_tl
5780 }
5781 }
5782 \clist_set:No \l_tmpa_clist \sparagraphtype
5783 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5784     {
5785         \stex_reactivate_macro:N \definiendum
5786         \stex_reactivate_macro:N \definame
5787         \stex_reactivate_macro:N \Definame
5788         \stex_reactivate_macro:N \premise
5789         \stex_reactivate_macro:N \definiens
5790     }
5791 \str_if_empty:NTF \sparagraphid {
5792     \str_if_empty:NTF \sparagraphname {
5793         \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5794             \stex_ref_new_doc_target:n {}
5795         }
5796     } {
5797         \stex_ref_new_doc_target:n {}
5798     }
5799 } {
5800     \stex_ref_new_doc_target:n \sparagraphid
5801 }

```

```

5802 \exp_args:NNx
5803 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5804   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5805     \tl_if_empty:nF{ ##1 }{
5806       \stex_get_symbol:n { ##1 }
5807       \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5808     }
5809   }
5810 }
5811 \stex_smsmode_do:
5812 \ignorespacesandpars
5813 }{
5814   \str_if_empty:NF \sparagraphname {
5815     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5816     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5817   }
5818   \stex_if_smsmode:F {
5819     \clist_set:No \l_tmpa_clist \sparagraphtype
5820     \tl_clear:N \l_tmpa_tl
5821     \clist_map_inline:Nn \l_tmpa_clist {
5822       \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
5823         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
5824       }
5825     }
5826     \tl_if_empty:NTF \l_tmpa_tl {
5827       \__stex_statements_sparagraph_end:
5828     }{
5829       \l_tmpa_tl
5830     }
5831     \end{stex_annotate_env}
5832   }
5833 }

```

### **\stexpatchparagraph**

```

5834
5835 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
5836   \stex_par:\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5837     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
5838       \titleemph{\l_stex_sparagraph_title_tl}:~
5839     }
5840   }{
5841     \titleemph{\l_stex_sparagraph_start_tl}~
5842   }
5843 }
5844 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\stex_par:\medskip}
5845
5846 \newcommand\stexpatchparagraph[3] [] {
5847   \str_set:Nx \l_tmpa_str{ #1 }
5848   \str_if_empty:NTF \l_tmpa_str {
5849     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
5850     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
5851   }{
5852     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2 }
5853     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }

```

```

5854     }
5855 }
5856
5857 \keys_define:nn { stex / inlinepara } {
5858   id      .str_set:N = \sparagraphid ,
5859   type    .str_set:N = \sparagraphtype ,
5860   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5861   from    .tl_set:N   = \sparagraphfrom ,
5862   to      .tl_set:N   = \sparagraphto ,
5863   name    .str_set:N   = \sparagraphname
5864 }
5865 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
5866   \tl_clear:N \sparagraphfrom
5867   \tl_clear:N \sparagraphto
5868   \str_clear:N \sparagraphid
5869   \str_clear:N \sparagraphtype
5870   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5871   \str_clear:N \sparagraphname
5872   \keys_set:nn { stex / inlinepara }{ #1 }
5873 }
5874 \NewDocumentCommand \inlinepara { 0{} m } {
5875   \begingroup
5876   \__stex_statements_inlinepara_args:n{ #1 }
5877   \clist_set:Nn \l_tmpa_clist \sparagraphtype
5878   \str_if_empty:NTF \sparagraphid {
5879     \str_if_empty:NTF \sparagraphname {
5880       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5881         \stex_ref_new_doc_target:n {}
5882       }
5883     } {
5884       \stex_ref_new_doc_target:n {}
5885     }
5886   } {
5887     \stex_ref_new_doc_target:n \sparagraphid
5888   }
5889   \stex_if_smsmode:TF{
5890     \str_if_empty:NF \sparagraphname {
5891       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
5892     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5893   }
5894 }{
5895   \seq_clear:N \l_tmpb_seq
5896   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5897     \tl_if_empty:nF{ ##1 }{
5898       \stex_get_symbol:n { ##1 }
5899       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5900         \l_stex_get_symbol_uri_str
5901       }
5902     }
5903   }
5904   \exp_args:Nnx
5905   \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}{
5906     \str_if_empty:NF \sparagraphtype {
5907       \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{

```

```

5908     }
5909     \str_if_empty:NF \sparagraphfrom {
5910       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5911     }
5912     \str_if_empty:NF \sparagraphto {
5913       \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5914     }
5915     \str_if_empty:NF \sparagraphname {
5916       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5917       \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
5918       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5919     }
5920     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5921       \clist_map_inline:Nn \l_tmpb_seq {
5922         \stex_ref_new_sym_target:n {##1}
5923       }
5924     }
5925     #2
5926   }
5927 }
5928 \endgroup
5929 \stex_smsmode_do:
5930 }
5931

```

(End definition for `\stexpatchparagraph`. This function is documented on page 49.)

```

5932 </package>

```

## Chapter 32

# The Implementation

```
5933 <*package>
5934 <@@=stex_sproof>
5935
5936 %%%%%%%%%%    sproof.dtx    %%%%%%%%%%
5937
```

### 32.1 Proofs

We first define some keys for the proof environment.

```
5938 \keys_define:nn { stex / spf } {
5939   id          .str_set_x:N = \spfid,
5940   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,
5941   from        .tl_set:N    = \l__stex_sproof_spf_from_tl ,
5942   proofend    .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
5943   type        .str_set_x:N = \spftype,
5944   title       .tl_set:N    = \spftitle,
5945   continues   .tl_set:N    = \l__stex_sproof_spf_continues_tl,
5946   functions   .tl_set:N    = \l__stex_sproof_spf_functions_tl,
5947   term        .tl_set:N    = \l__stex_sproof_spf_term_tl,
5948   method      .tl_set:N    = \l__stex_sproof_spf_method_tl,
5949   hide        .bool_set:N  = \l__stex_sproof_spf_hide_bool
5950 }
5951 \cs_new_protected:Nn \l__stex_sproof_spf_args:n {
5952   \str_clear:N \spfid
5953   \tl_clear:N \l__stex_sproof_spf_for_tl
5954   \tl_clear:N \l__stex_sproof_spf_from_tl
5955   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
5956   \str_clear:N \spftype
5957   \tl_clear:N \spftitle
5958   \tl_clear:N \l__stex_sproof_spf_continues_tl
5959   \tl_clear:N \l__stex_sproof_spf_term_tl
5960   \tl_clear:N \l__stex_sproof_spf_functions_tl
5961   \tl_clear:N \l__stex_sproof_spf_method_tl
5962   \bool_set_false:N \l__stex_sproof_spf_hide_bool
5963   \keys_set:nn { stex / spf }{ #1 }
5964 }
5965 \bool_set_true:N \l__stex_sproof_inc_counter_bool
```



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

```
5966 \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.

```
5967 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
5968 \cs_new_protected:Npn \sproofnumber {
5969   \int_set:Nn \l_tmpa_int {1}
5970   \bool_while_do:nn {
5971     \int_compare_p:nNn {
5972       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5973     } > 0
5974   }{
5975     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
5976     \int_incr:N \l_tmpa_int
5977   }
5978 }
5979 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
5980   \int_set:Nn \l_tmpa_int {1}
5981   \bool_while_do:nn {
5982     \int_compare_p:nNn {
5983       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5984     } > 0
5985   }{
5986     \int_incr:N \l_tmpa_int
5987   }
5988   \int_compare:nNnF \l_tmpa_int = 1 {
5989     \int_decr:N \l_tmpa_int
5990   }
5991   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
5992     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1
5993   }
5994 }
5995
5996 \cs_new_protected:Npn \__stex_sproof_add_counter: {
5997   \int_set:Nn \l_tmpa_int {1}
5998   \bool_while_do:nn {
5999     \int_compare_p:nNn {
6000       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6001     } > 0
6002   }{
6003     \int_incr:N \l_tmpa_int
6004   }
6005   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 1 }
6006 }
6007
```

```

6008 \cs_new_protected:Npn \__stex_sproof_remove_counter: {
6009   \int_set:Nn \l_tmpa_int {1}
6010   \bool_while_do:nn {
6011     \int_compare_p:nNn {
6012       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6013     } > 0
6014   }{
6015     \int_incr:N \l_tmpa_int
6016   }
6017   \int_decr:N \l_tmpa_int
6018   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 0 }
6019 }

```

**\sproofend** This macro places a little box at the end of the line if there is space, or at the end of the next line if there isn't

```

6020 \def\sproof@box{
6021   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
6022 }
6023 \def\sproofend{
6024   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
6025     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
6026   }
6027 }

```

(End definition for \sproofend. This function is documented on page 49.)

**spf@\*kw**

```

6028 \def\spf@proofsketch@kw{Proof~Sketch}
6029 \def\spf@proof@kw{Proof}
6030 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

6031 \AddToHook{begindocument}{
6032   \ltx@ifpackageloaded{babel}{
6033     \makeatletter
6034     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6035     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6036       \input{sproof-ngerman.ldf}
6037     }
6038     \clist_if_in:NnT \l_tmpa_clist {finnish}{
6039       \input{sproof-finnish.ldf}
6040     }
6041     \clist_if_in:NnT \l_tmpa_clist {french}{
6042       \input{sproof-french.ldf}
6043     }
6044     \clist_if_in:NnT \l_tmpa_clist {russian}{
6045       \input{sproof-russian.ldf}
6046     }
6047     \makeatother
6048   }{}
6049 }

```

spfsketch

```

6050 \newcommand\spfsketch[2] [] {
6051   \begin{group}
6052   \let \premise \stex_proof_premise:
6053   \stex_sproof_spf_args:n{#1}
6054   \stex_if_smsmode:TF {
6055     \str_if_empty:NF \spfid {
6056       \stex_ref_new_doc_target:n \spfid
6057     }
6058   }{
6059     \seq_clear:N \l_tmpa_seq
6060     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6061       \tl_if_empty:nF{ ##1 }{
6062         \stex_get_symbol:n { ##1 }
6063         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6064           \l_stex_get_symbol_uri_str
6065         }
6066       }
6067     }
6068     \exp_args:Nnx
6069     \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
6070       \str_if_empty:NF \spftype {
6071         \stex_annotate_invisible:nnn{type}{\spftype}{ }
6072       }
6073       \clist_set:Nn \l_tmpa_clist \spftype
6074       \tl_set:Nn \l_tmpa_tl {
6075         \titleemph{
6076           \tl_if_empty:NTF \spftitle {
6077             \spf@proofsketch@kw
6078           }{
6079             \spftitle
6080           }
6081         }:-
6082       }
6083       \clist_map_inline:Nn \l_tmpa_clist {
6084         \exp_args:Nn \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6085           \tl_clear:N \l_tmpa_tl
6086         }
6087       }
6088       \str_if_empty:NF \spfid {
6089         \stex_ref_new_doc_target:n \spfid
6090       }
6091       \l_tmpa_tl #2 \sproofend
6092     }
6093   }
6094   \endgroup
6095   \stex_smsmode_do:
6096 }
6097

```

(End definition for *spfsketch*. This function is documented on page 48.)

```

\__stex_sproof_maybe_comment:
\__stex_sproof_maybe_comment_end:
\__stex_sproof_start_comment:
6098 \bool_set_false:N \l__stex_sproof_in_spfblock_bool

```

```

6099
6100 \cs_new_protected:Nn \__stex_sproof_maybe_comment: {
6101   \bool_if:NF \l__stex_sproof_in_spfblock_bool {
6102     \par \setbox \l_tmpa_box \vbox \bgroup \everypar{\__stex_sproof_start_comment:}
6103   }
6104 }
6105 \cs_new_protected:Nn \__stex_sproof_maybe_comment_end: {
6106   \bool_if:NF \l__stex_sproof_in_spfblock_bool { \egroup }
6107 }
6108 \cs_new_protected:Nn \__stex_sproof_start_comment: {
6109   \csname @ @ par\endcsname\egroup\item[]\bgroup\stexcommentfont
6110 }
6111

```

(End definition for \\_\_stex\_sproof\_maybe\_comment:, \\_\_stex\_sproof\_maybe\_comment\_end:, and \\_\_stex\_sproof\_start\_comment:.)

\stexcommentfont

```

6112 \cs_new_protected:Npn \stexcommentfont {
6113   \small\itshape
6114 }

```

(End definition for \stexcommentfont. 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.

```

6115 \cs_new_protected:Nn \__stex_sproof_start_env:nnn {
6116   \seq_clear:N \l_tmpa_seq
6117   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6118     \tl_if_empty:NF{ ##1 }{
6119       \stex_get_symbol:n { ##1 }
6120       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6121         \l_stex_get_symbol_uri_str
6122       }
6123     }
6124   }
6125   \exp_args:Nnnx
6126   \begin{stex_annotate_env}{#1}{\seq_use:Nn \l_tmpa_seq {,}}
6127   \str_if_empty:NF \spftype {
6128     \stex_annotate_invisible:nnn{type}{\spftype}{}
6129   }
6130   #3 {\~\stex_annotate:nnn{spftitle}{}{#2}}
6131   \str_if_empty:NF \spfid {
6132     \stex_ref_new_doc_target:n \spfid
6133   }
6134   \begin{stex_annotate_env}{spfbody}{\bool_if:NTF \l__stex_sproof_spf_hide_bool {false}{true}
6135   \bool_if:NT \l__stex_sproof_spf_hide_bool{
6136     \stex_html_backend:F{\setbox\l_tmpa_box\vbox\bgroup}
6137   }
6138   \begin{list}{}{
6139     \setlength\topsep{0pt}
6140     \setlength\parsep{0pt}
6141     \setlength\rightmargin{0pt}

```

```

6142 } \_stex_sproof_maybe_comment:
6143 }
6144 \cs_new_protected:Nn \_stex_sproof_end_env:n {
6145   \stex_if_smsmode:F{
6146     \_stex_sproof_maybe_comment_end:
6147     \end{list}
6148     \bool_if:NT \l__stex_sproof_spf_hide_bool{
6149       \stex_html_backend:F{\egroup}
6150     }
6151     \clist_set:No \l_tmpa_clist \spftype
6152     #1
6153     \end{stex_annotate_env}
6154     \end{stex_annotate_env}
6155   }
6156 }
6157 }
6158 \NewDocumentEnvironment{sproof}{s O{} m}{
6159   \intarray_gzero:N \l__stex_sproof_counter_intarray
6160   \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
6161   \stex_reactivate_macro:N \yield
6162   \stex_reactivate_macro:N \eqstep
6163   \stex_reactivate_macro:N \assumption
6164   \stex_reactivate_macro:N \conclude
6165   \stex_reactivate_macro:N \spfstep
6166   \_stex_sproof_spf_args:n{#2}
6167   \stex_if_smsmode:TF {
6168     \str_if_empty:NF \spfid {
6169       \stex_ref_new_doc_target:n \spfid
6170     }
6171   }{
6172     \_stex_sproof_start_env:nnn{sproof}{#3}{
6173       \clist_set:No \l_tmpa_clist \spftype
6174       \tl_clear:N \l_tmpa_tl
6175       \clist_map_inline:Nn \l_tmpa_clist {
6176         \tl_if_exist:cT {\_stex_sproof_sproof_##1_start:}{
6177           \tl_set:Nn \l_tmpa_tl {\use:c{\_stex_sproof_sproof_##1_start:}}
6178         }
6179         \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6180           \tl_set:Nn \l_tmpa_tl {\use:n{}}
6181         }
6182       }
6183       \tl_if_empty:NTF \l_tmpa_tl {
6184         \_stex_sproof_sproof_start:
6185       }{
6186         \l_tmpa_tl
6187       }
6188     }
6189   }
6190   \stex_smsmode_do:
6191 }{\_stex_sproof_end_env:n{
6192   \tl_clear:N \l_tmpa_tl
6193   \clist_map_inline:Nn \l_tmpa_clist {
6194     \tl_if_exist:cT {\_stex_sproof_sproof_##1_end:}{
6195       \tl_set:Nn \l_tmpa_tl {\use:c{\_stex_sproof_sproof_##1_end:}}

```

```

6196     }
6197   }
6198   \tl_if_empty:NTF \l_tmpa_tl {
6199     \__stex_sproof_sproof_end:
6200   }{
6201     \l_tmpa_tl
6202   }
6203 }}
6204 \NewDocumentEnvironment{subproof}{s O{} m}{
6205   \__stex_sproof_spf_args:n{#2}
6206   \stex_if_smsmode:TF {
6207     \str_if_empty:NF \spfid {
6208       \stex_ref_new_doc_target:n \spfid
6209     }
6210   }{
6211     \__stex_sproof_start_env:nnn{subproof}{\item[\sproofnumber]\ignorespacesandpars #3}{}
6212   }
6213   \__stex_sproof_add_counter:
6214   \stex_smsmode_do:
6215 }{\__stex_sproof_remove_counter:\__stex_sproof_end_env:n{
6216   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6217     \__stex_sproof_inc_counter:
6218   }
6219   \aftergroup\__stex_sproof_maybe_comment:
6220 }
6221 \AddToHook{env/subproof/before}{\__stex_sproof_maybe_comment_end:}
6222
6223 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
6224   \par\noindent\titleemph{
6225     \tl_if_empty:NTF \spftype {
6226       \spf@proof@kw
6227     }{
6228       \spftype
6229     }
6230   }:
6231 }
6232 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
6233
6234 \newcommand\stexpatchproof[3] [] {
6235   \str_set:Nx \l_tmpa_str{ #1 }
6236   \str_if_empty:NTF \l_tmpa_str {
6237     \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
6238     \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
6239   }{
6240     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
6241     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
6242   }
6243 }

\pstep
\conclude
\assumption
\have
\eqstep

```

```

6248 type .str_set_x:N = \spftype,
6249 title .tl_set:N = \spftitle,
6250 method .tl_set:N = \l__stex_sproof_spf_method_tl,
6251 term .tl_set:N = \l__stex_sproof_spf_term_tl
6252 }
6253 \cs_new_protected:Nn \__stex_sproof_spfstep_args:n {
6254 \str_clear:N \spfstepid
6255 \clist_clear:N \l__stex_sproof_spf_for_clist
6256 \str_clear:N \spftype
6257 \tl_clear:N \l__stex_sproof_spf_method_tl
6258 \tl_clear:N \l__stex_sproof_spf_term_tl
6259 %\bool_set_false:N \l__stex_sproof_inc_counter_bool
6260 \keys_set:nn { stex / spfsteps }{ #1 }
6261 }
6262
6263 \cs_new_protected:Nn \__stex_sproof_make_step_macro:Nnnnn {
6264 \NewDocumentCommand #1 {s O{} +m} {
6265 \__stex_sproof_maybe_comment_end:
6266
6267 \__stex_sproof_spfstep_args:n{##2}
6268 \stex_annotate:nnn{spfstep}{#2}{
6269 \tl_if_empty:NF \l__stex_sproof_spf_term_tl {
6270 \stex_annotate_invisible:nnn{spfyield}{}{\l__stex_sproof_spf_term_tl$}
6271 }
6272 \bool_if:NTF \l__stex_sproof_in_spfblock_bool {
6273 #4
6274 }{
6275 \item[\IfBooleanTF ##1 {}{#3}]
6276 }
6277 \ignorespacesandpars ##3
6278 }
6279 \bool_if:NF \l__stex_sproof_in_spfblock_bool { \IfBooleanTF ##1 {}{ #5 } }
6280 \__stex_sproof_maybe_comment:
6281 }
6282 \stex_deactivate_macro:Nn #1 {sproof~environments}
6283 }
6284
6285 \__stex_sproof_make_step_macro:Nnnnn \assumption {assumption} \sproofnumber {} \__stex_sproof
6286 \__stex_sproof_make_step_macro:Nnnnn \conclude {conclusion} {\$ \Rightarrow$} {} {}
6287 \__stex_sproof_make_step_macro:Nnnnn \spfstep {} \sproofnumber {} \__stex_sproof_inc_counter
6288
6289 \NewDocumentCommand \eqstep {s m}{
6290 \__stex_sproof_maybe_comment_end:
6291 \bool_if:NTF \l__stex_sproof_in_spfblock_bool {
6292 $=$
6293 }{
6294 \item[$=$]
6295 }
6296 $\stex_annotate:nnn{spfstep}{eq}{ #2 }$
6297 \__stex_sproof_maybe_comment:
6298 }
6299 \stex_deactivate_macro:Nn \eqstep {sproof~environments}
6300
6301 \NewDocumentCommand \yield {+m}{

```

```

6302 \stex_annotate:nnn{spfyield}{\}{ #1 }
6303 }
6304 \stex_deactivate_macro:Nn \yield {sproof~environments}
6305
6306 \NewDocumentEnvironment{spfblock}{\}{
6307   \item[]
6308   \bool_set_true:N \l__stex_sproof_in_spfblock_bool
6309 }{
6310   \aftergroup\__stex_sproof_maybe_comment:
6311 }
6312 \AddToHook{env/spfblock/before}{\__stex_sproof_maybe_comment_end:}
6313

```

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

### `\spfidea`

```

6314 \NewDocumentCommand\spfidea{0{} +m}{
6315   \__stex_sproof_spf_args:n{#1}
6316   \titleemph{
6317     \tl_if_empty:NTF \spftype {Proof~Idea}{
6318       \spftype
6319     }:
6320   }~#2
6321   \sproofend
6322 }

```

(End definition for `\spfidea`. This function is documented on page 48.)

```

6323 \newcommand\spfjust[1]{
6324   #1
6325 }
6326 \</package>

```

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



## Chapter 33

# STEX -Others Implementation

```
6327 <*package>
6328
6329 %%%%%%%%%% others.dtx %%%%%%%%%%
6330
6331 <@@=stex_others>
        Warnings and error messages
6332 % None

\MSC Math subject classifier

6333 \NewDocumentCommand \MSC {m} {
6334 % TODO
6335 }

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

6336 \@ifpackageloaded{tikzinput}{
6337 \RequirePackage{stex-tikzinput}
6338 }{}
6339
6340 \bool_if:NT \c_stex_persist_mode_bool {
6341 \let__stex_notation_restore_notation_old:nnnnn
6342 \__stex_notation_restore_notation:nnnnn
6343 \def__stex_notation_restore_notation_new:nnnnn#1#2#3#4#5{
6344 \__stex_notation_restore_notation_old:nnnnn{#1}{#2}{#3}{#4}{#5}
6345 \ExplSyntaxOn
6346 }
6347 \def__stex_notation_restore_notation:nnnnn{
6348 \ExplSyntaxOff
6349 \catcode'\sim10
6350 \__stex_notation_restore_notation_new:nnnnn
6351 }
6352 \input{\jobname.sms}
6353 \let__stex_notation_restore_notation:nnnnn
6354 \__stex_notation_restore_notation_old:nnnnn
6355 \prop_if_exist:NT\c_stex_mathhub_main_manifest_prop{
```

```

6356     \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
6357     \l_tmpa_str
6358     \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str_manifest_prop }
6359     \c_stex_mathhub_main_manifest_prop
6360     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
6361   }
6362 }
6363 </package>

```

## Chapter 34

# STEX -Metatheory Implementation

```
6364 <*package>
6365 <@@=stex_modules>
6366
6367 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
6368
6369 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX/meta}
6370 \begingroup
6371 \stex_module_setup:nn{
6372   ns=\c_stex_metatheory_ns_str,
6373   meta=NONE
6374 }{Metatheory}
6375 \stex_reactivate_macro:N \symdecl
6376 \stex_reactivate_macro:N \notation
6377 \stex_reactivate_macro:N \symdef
6378 \ExplSyntaxOff
6379 \csname stex_suppress_html:n\endcsname{
6380   % is-a (a:A, a \in A, a is an A, etc.)
6381   \symdecl{isa}[args=ai]
6382   \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
6383   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
6384   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
6385
6386   % bind (\forall, \Pi, \lambda etc.)
6387   \symdecl{bind}[args=Bi,assoc=pre]
6388   \notation{bind}[depfun,prec=nobrackets,op={(\cdot)\;\to\;\cdot}]{\comp( #1 \comp{}\;\to\;)}
6389   \notation{bind}[forall]{\comp\forall #1.\;#2}{##1 \comp, ##2}
6390   \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6391
6392   % implicit bind
6393   \symdecl{implicitbind}[args=Bi,assoc=pre]
6394   \notation{implicitbind}[braces,prec=nobrackets,op={(\cdot\;)_I\;\cdot}]{\comp\{ #1 \comp{}
6395   \notation{implicitbind}[depfun,prec=nobrackets]{\comp( #1 \comp{}\;\to_I\; } #2}{##1 \comp,
6396   \notation{implicitbind}[Pi]{\comp\prod^I_{#1}#2}{##1\comp,##2}
6397
6398   % dummy variable
```

```

6399 \symdecl{dummyvar}
6400 \notation{dummyvar}[underscore]{\comp\_}
6401 \notation{dummyvar}[dot]{\comp\cdot}
6402 \notation{dummyvar}[dash]{\comp{\rm --}}
6403
6404 %fromto (function space, Hom-set, implication etc.)
6405 \symdecl{fromto}[args=ai]
6406 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
6407 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
6408
6409 % mapto (lambda etc.)
6410 \symdecl{mapto}[args=Bi]
6411 %\notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
6412 %\notation{mapto}[lambda]{\comp\lambda #1 \comp.\; #2}{#1 \comp, #2}
6413 %\notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.\; #2}{#1 \comp, #2}
6414
6415 % function/operator application
6416 \symdecl{apply}[args=ia]
6417 \notation{apply}[prec=0;0x\infpres,parens,op=\cdot(\cdot)]{#1 \comp( #2 \comp)}{##1 \comp,
6418 \notation{apply}[prec=0;0x\infpres,lambda]{#1 \; #2 }{##1 \; ; ##2}
6419
6420 % collection of propositions/booleans/truth values
6421 \symdecl{prop}[name=proposition]
6422 \notation{prop}[prop]{\comp{\rm prop}}
6423 \notation{prop}[BOOL]{\comp{\rm BOOL}}
6424
6425 \symdecl{judgmentholds}[args=1]
6426 \notation{judgmentholds}[vdash,op=\vdash]{\comp\vdash\; #1}
6427
6428 % sequences
6429 \symdecl{seqtype}[args=1]
6430 \notation{seqtype}[kleene]{#1^{\comp\ast}}
6431
6432 \symdecl{seqexpr}[args=a]
6433 \notation{seqexpr}[angle,prec=nobrackets]{\comp\langle #1\comp\rangle}{##1\comp,##2}
6434
6435 \symdef{seqmap}[args=abi,setlike]{\comp\{#3 \comp| #2\comp\in \dobrackets{#1} \comp\}}{##1
6436 \symdef{seqprepend}[args=ia]{#1 \comp{::} #2}{##1 \comp, ##2}
6437 \symdef{seqappend}[args=ai]{#1 \comp{::} #2}{##1 \comp, ##2}
6438 \symdef{seqfoldleft}[args=iabbi]{ \comp{foldl}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\comp
6439 \symdef{seqfoldright}[args=iabbi,op=foldr]{ \comp{foldr}\dobrackets{#1,#2}\dobrackets{#3\comp
6440 \symdef{seqhead}[args=a]{\comp{head}\dobrackets{#1}}{##1 \comp, ##2}
6441 \symdef{seqtail}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6442 \symdef{seqlast}[args=a]{\comp{last}\dobrackets{#1}}{##1 \comp, ##2}
6443 \symdef{seqinit}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6444
6445 \symdef{sequence-index}[args=2,li,prec=nobrackets]{\comp{#1}_{#2}}
6446 \notation{sequence-index}[ui,prec=nobrackets]{\comp{#1}^{\comp{#2}}}
6447
6448 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{\,\ellipses}}{##1\comp,##2}
6449 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{\,\ellipses,}#2}{##1\comp,##2}
6450 \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{\,\ellipses,}#2\comp{\,\ellipses,}#3}
6451
6452 % nat literals

```

```

6453 \symdef{natliteral}{\comp{\mathtt{Ord}}}
6454
6455 % letin (''let'', local definitions, variable substitution)
6456 \symdecl{letin}[args=bii]
6457 \notation{letin}[let]{\comp{\rm let}}\;#1\comp{=}\;#2\;\comp{\rm in}}\;#3}
6458 \notation{letin}[subst]{#3 \comp[ #1 \comp/ #2 \comp]}
6459 \notation{letin}[frac]{#3 \comp[ \frac{#2}{#1} \comp]}
6460
6461 % structures
6462 \symdecl*{module-type}[args=1]
6463 \notation{module-type}{\comp{\mathtt{MOD}}} #1}
6464 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
6465 \notation{mathstruct}[angle,prec=nobrackets]{\comp\angle #1 \comp\angle}{##1 \comp, ##2}
6466
6467 % objects
6468 \symdecl{object}
6469 \notation{object}{\comp{\mathtt{OBJECT}}}
6470
6471 }
6472
6473 % The following are abbreviations in the sTeX corpus that are left over from earlier
6474 % developments. They will eventually be phased out.
6475
6476 \ExplSyntaxOn
6477 \stex_add_to_current_module:n{
6478   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
6479   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
6480   \def\livar{\csname sequence-index\endcsname[li]}
6481   \def\uivar{\csname sequence-index\endcsname[ui]}
6482   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
6483   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
6484 }
6485 \__stex_modules_end_module:
6486 \endgroup
6487 \</package>

```

## Chapter 35

# Tikzinput Implementation

```
6488 <@@=tikzinput>
6489 <*package>
6490
6491 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
6492
6493 \ProvidesExplPackage{tikzinput}{2022/05/24}{3.1.0}{tikzinput package}
6494 \RequirePackage{l3keys2e}
6495
6496 \keys_define:nn { tikzinput } {
6497   image .bool_set:N = \c_tikzinput_image_bool,
6498   image .default:n = false ,
6499   unknown .code:n = {}
6500 }
6501
6502 \ProcessKeysOptions { tikzinput }
6503
6504 \bool_if:NTF \c_tikzinput_image_bool {
6505   \RequirePackage{graphicx}
6506
6507   \providecommand\usetikzlibrary[]{}
6508   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
6509 }{
6510   \RequirePackage{tikz}
6511   \RequirePackage{standalone}
6512
6513   \newcommand \tikzinput [2] [] {
6514     \setkeys{Gin}{#1}
6515     \ifx \Gin@ewidth \Gin@exclamation
6516       \ifx \Gin@eheight \Gin@exclamation
6517         \input { #2 }
6518       \else
6519         \resizebox{!}{ \Gin@eheight }{
6520           \input { #2 }
6521         }
6522       \fi
6523     \else
6524       \ifx \Gin@eheight \Gin@exclamation
6525         \resizebox{ \Gin@ewidth }{!}{
```

```

6526         \input { #2 }
6527     }
6528     \else
6529         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
6530             \input { #2 }
6531         }
6532     \fi
6533 \fi
6534 }
6535 }
6536
6537 \newcommand \ctikzinput [2] [] {
6538     \begin{center}
6539         \tikzinput [#1] {#2}
6540     \end{center}
6541 }
6542
6543 \@ifpackageloaded{stex}{
6544     \RequirePackage{stex-tikzinput}
6545 }{}
6546
6547 </package>
6548 <*stex>
6549 \ProvidesExplPackage{stex-tikzinput}{2022/05/24}{3.1.0}{stex-tikzinput}
6550 \RequirePackage{stex}
6551 \RequirePackage{tikzinput}
6552
6553 \newcommand\mhtikzinput[2] []{%
6554     \def\Gin@mhrepos{ }\setkeys{Gin}{#1}%
6555     \stex_in_repository:nn\Gin@mhrepos{
6556         \tikzinput[#1]{\mhp@hpath{##1}{#2}}
6557     }
6558 }
6559 \newcommand\cmhtikzinput[2] [] {\begin{center}\mhtikzinput[#1]{#2}\end{center}}
6560
6561 \cs_new_protected:Nn \__tikzinput_usetikzlibrary:nn {
6562     \pgfkeys@spdef\pgf@temp{#1}
6563     \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
6564     \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgf@temp
6565     \expandafter\edef\csname tikz@library@#1@atcode\endcsname{\the\catcode'\@}
6566     \expandafter\edef\csname tikz@library@#1@barcode\endcsname{\the\catcode'\|}
6567     \expandafter\edef\csname tikz@library@#1@dollarcode\endcsname{\the\catcode'\$}
6568     \catcode'\@=11
6569     \catcode'\|=12
6570     \catcode'\$=3
6571     \pgfutil@InputIfFileExists{#2}{-}{-}
6572     \catcode'\@=\csname tikz@library@#1@atcode\endcsname
6573     \catcode'\|=\csname tikz@library@#1@barcode\endcsname
6574     \catcode'\$=\csname tikz@library@#1@dollarcode\endcsname
6575 }
6576
6577
6578 \newcommand\libusetikzlibrary[1]{

```

```

6579 \prop_if_exist:NF \l_stex_current_repository_prop {
6580   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6581 }
6582 \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
6583   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6584 }
6585 \seq_clear:N \l__tikzinput_libinput_files_seq
6586 \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
6587 \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
6588
6589 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
6590   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / tikzlibrary
6591   \IfFileExists{ \l_tmpa_str }{
6592     \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6593   }{}
6594   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
6595   \seq_put_right:No \l_tmpa_seq \l_tmpa_str
6596 }
6597
6598 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / tikzlibrary #1 .code.t
6599 \IfFileExists{ \l_tmpa_str }{
6600   \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6601 }{}
6602
6603 \seq_if_empty:NTF \l__tikzinput_libinput_files_seq {
6604   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .code.t
6605 }{
6606   \int_compare:nNnTF {\seq_count:N \l__tikzinput_libinput_files_seq} = 1 {
6607     \seq_map_inline:Nn \l__tikzinput_libinput_files_seq {
6608       \__tikzinput_usetikzlibrary:nn{#1}{ ##1 }
6609     }
6610   }{
6611     \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .co
6612   }
6613 }
6614 }
6615 </stex>

```



## Chapter 36

# document-structure.sty Implementation

```
6616 <*package>
6617 <@@=document_structure>
6618 \ProvidesExplPackage{document-structure}{2022/05/24}{3.1.0}{Modular Document Structure}
6619 \RequirePackage{13keys2e}
```

### 36.1 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).

```
6620
6621 \keys_define:nn{ document-structure }{
6622   class      .str_set_x:N = \c_document_structure_class_str,
6623   topsect    .str_set_x:N = \c_document_structure_topsect_str,
6624   unknown    .code:n      = {
6625     \PassOptionsToClass{\CurrentOption}{stex}
6626     \PassOptionsToClass{\CurrentOption}{tikzinput}
6627   }
6628   % showignores .bool_set:N = \c_document_structure_showignores_bool,
6629 }
6630 \ProcessKeysOptions{ document-structure }
6631 \str_if_empty:NT \c_document_structure_class_str {
6632   \str_set:Nn \c_document_structure_class_str {article}
6633 }
6634 \str_if_empty:NT \c_document_structure_topsect_str {
6635   \str_set:Nn \c_document_structure_topsect_str {section}
6636 }
```

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

```
6637 \RequirePackage{xspace}
6638 \RequirePackage{comment}
6639 \RequirePackage{stex}
6640 \AddToHook{begindocument}{
```

```

6641 \ltx@ifpackageloaded{babel}{
6642   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6643   \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
6644     \makeatletter\input{document-structure-ngerman.ldf}\makeatother
6645   }
6646 }{}
6647 }

```

`\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.

```

6648 \int_new:N \l_document_structure_section_level_int
6649 \str_case:NnF \c_document_structure_topsect_str {
6650   {part}}{
6651     \int_set:Nn \l_document_structure_section_level_int {0}
6652   }
6653   {chapter}{
6654     \int_set:Nn \l_document_structure_section_level_int {1}
6655   }
6656 }{
6657   \str_case:NnF \c_document_structure_class_str {
6658     {book}{
6659       \int_set:Nn \l_document_structure_section_level_int {0}
6660     }
6661     {report}{
6662       \int_set:Nn \l_document_structure_section_level_int {0}
6663     }
6664   }{
6665     \int_set:Nn \l_document_structure_section_level_int {2}
6666   }
6667 }

```

## 36.2 Document Structure

The structure of the document is given by the `sfragment` environment. The hierarchy is adjusted automatically according to the L<sup>A</sup>T<sub>E</sub>X class in effect.

`\currentsectionlevel` For the `\currentsectionlevel` and `\Currentsectionlevel` macros we use an internal macro `\current@section@level` that only contains the keyword (no markup). We initialize it with “document” as a default. In the generated OMDoc, we only generate a text element of class `omdoc_currentsectionlevel`, which will be instantiated by CSS later.<sup>9</sup>

```

6668 \def\current@section@level{document}%
6669 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
6670 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%

```

(End definition for `\currentsectionlevel`. This function is documented on page 54.)

`\skipfragment`

```

6671 \cs_new_protected:Npn \skipfragment {

```

<sup>9</sup>EDNOTE: MK: we may have to experiment with the more powerful uppercasing macro from `mfirstuc.sty` once we internationalize.

```

6672 \ifcase\l_document_structure_section_level_int
6673 \or\stepcounter{part}
6674 \or\stepcounter{chapter}
6675 \or\stepcounter{section}
6676 \or\stepcounter{subsection}
6677 \or\stepcounter{subsubsection}
6678 \or\stepcounter{paragraph}
6679 \or\stepcounter{subparagraph}
6680 \fi
6681 }

```

(End definition for `\skipfragment`. This function is documented on page 53.)

**blindfragment**

```

6682 \newcommand\at@begin@blindsfragment[1]{
6683 \newenvironment{blindfragment}
6684 {
6685 \int_incr:N\l_document_structure_section_level_int
6686 \at@begin@blindsfragment\l_document_structure_section_level_int
6687 }{}

```

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

```

6688 \newcommand\sfragment@nonum[2]{
6689 \ifx\hyper@anchor\@undefined\else\phantomsection\fi
6690 \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
6691 }

```

(End definition for `\sfragment@nonum`. This function is documented on page ??.)

**\sfragment@num** convenience macro: `\sfragment@num{<level>}{<title>}` makes numbered sectioning with title `<title>` at level `<level>`. We have to check the `short` key was given in the `sfragment` environment and – if it is use it. But how to do that depends on whether the `rdfmata` package has been loaded. In the end we call `\sref@label@id` to enable crossreferencing.

```

6692 \newcommand\sfragment@num[2]{
6693 \tl_if_empty:NTF \l__document_structure_sfragment_short_tl {
6694 \@nameuse{#1}{#2}
6695 }{
6696 \cs_if_exist:NTF\rdfmata@sectioning{
6697 \@nameuse{rdfmata@#1@old}[\l__document_structure_sfragment_short_tl]{#2}
6698 }{
6699 \@nameuse{#1}[\l__document_structure_sfragment_short_tl]{#2}
6700 }
6701 }
6702 %\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\sfragment@id
6703 }

```

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

**sfragment**

```

6704 \keys_define:nn { document-structure / sfragment }{
6705 id .str_set_x:N = \l__document_structure_sfragment_id_str,
6706 date .str_set_x:N = \l__document_structure_sfragment_date_str,

```

```

6707 creators      .clist_set:N = \l__document_structure_sfragment_creators_clist,
6708 contributors  .clist_set:N = \l__document_structure_sfragment_contributors_clist,
6709 srccite       .tl_set:N     = \l__document_structure_sfragment_srccite_tl,
6710 type         .tl_set:N     = \l__document_structure_sfragment_type_tl,
6711 short        .tl_set:N     = \l__document_structure_sfragment_short_tl,
6712 intro        .tl_set:N     = \l__document_structure_sfragment_intro_tl,
6713 imports      .tl_set:N     = \l__document_structure_sfragment_imports_tl,
6714 loadmodules  .bool_set:N   = \l__document_structure_sfragment_loadmodules_bool
6715 }
6716 \cs_new_protected:Nn \l__document_structure_sfragment_args:n {
6717   \str_clear:N \l__document_structure_sfragment_id_str
6718   \str_clear:N \l__document_structure_sfragment_date_str
6719   \clist_clear:N \l__document_structure_sfragment_creators_clist
6720   \clist_clear:N \l__document_structure_sfragment_contributors_clist
6721   \tl_clear:N \l__document_structure_sfragment_srccite_tl
6722   \tl_clear:N \l__document_structure_sfragment_type_tl
6723   \tl_clear:N \l__document_structure_sfragment_short_tl
6724   \tl_clear:N \l__document_structure_sfragment_imports_tl
6725   \tl_clear:N \l__document_structure_sfragment_intro_tl
6726   \bool_set_false:N \l__document_structure_sfragment_loadmodules_bool
6727   \keys_set:nn { document-structure / sfragment } { #1 }
6728 }

```

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

```

6729 \newif\if@mainmatter\@mainmattertrue
6730 \newcommand\at@begin@sfragment[3][]{ }

```

Then we define a helper macro that takes care of the sectioning magic. It comes with its own key/value interface for customization.

```

6731 \keys_define:nn { document-structure / sectioning }{
6732   name      .str_set_x:N = \l__document_structure_sect_name_str  ,
6733   ref       .str_set_x:N = \l__document_structure_sect_ref_str   ,
6734   clear     .bool_set:N  = \l__document_structure_sect_clear_bool ,
6735   clear     .default:n   = {true}                                ,
6736   num       .bool_set:N  = \l__document_structure_sect_num_bool  ,
6737   num       .default:n   = {true}
6738 }
6739 \cs_new_protected:Nn \l__document_structure_sect_args:n {
6740   \str_clear:N \l__document_structure_sect_name_str
6741   \str_clear:N \l__document_structure_sect_ref_str
6742   \bool_set_false:N \l__document_structure_sect_clear_bool
6743   \bool_set_false:N \l__document_structure_sect_num_bool
6744   \keys_set:nn { document-structure / sectioning } { #1 }
6745 }
6746 \newcommand\omdoc@sectioning[3][]{
6747   \l__document_structure_sect_args:n {#1 }
6748   \let\omdoc@sect@name\l__document_structure_sect_name_str
6749   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
6750   \if@mainmatter% numbering not overridden by frontmatter, etc.
6751     \bool_if:NTF \l__document_structure_sect_num_bool {
6752       \sfragment@num{#2}{#3}
6753     }{

```

```

6754     \sfragment@nonum{#2}{#3}
6755   }
6756   \def\current@section@level{\omdoc@sect@name}
6757   \else
6758     \sfragment@nonum{#2}{#3}
6759   \fi
6760 }% 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.

```

6761 \newcommand\sfragment@redefine@addtocontents[1]{%
6762 %\edef\__document_structureimport{#1}%
6763 %\@for\@I:=\__document_structureimport\do{%
6764 %\edef\@path{\csname module@\@I @path\endcsname}%
6765 %\@ifundefined{tf@toc}\relax%
6766 %    {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
6767 %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
6768 %\def\addcontentsline##1##2##3{%
6769 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6770 %\else% hyperref.sty not loaded
6771 %\def\addcontentsline##1##2##3{%
6772 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6773 %\fi
6774 }% hypreref.sty loaded?

```

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

```

6775 \newenvironment{sfragment}[2][ ]% keys, title
6776 {
6777   \__document_structure_sfragment_args:n { #1 }%\sref@target%

```

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

```

6778   \stex_csl_to_imports:No \usemodule \l__document_structure_sfragment_imports_tl
6779
6780   \bool_if:NT \l__document_structure_sfragment_loadmodules_bool {
6781     \sfragment@redefine@addtocontents{
6782       %\@ifundefined{module@id}\used@modules%
6783       %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}
6784     }
6785   }

```

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

```

6786
6787   \stex_document_title:n { #2 }
6788
6789   \int_incr:N\l__document_structure_section_level_int
6790   \ifcase\l__document_structure_section_level_int
6791     \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
6792     \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
6793     \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
6794     \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}

```

```

6795 \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
6796 \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#1}
6797 \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#1}
6798 \fi
6799 \at@begin@sfragment[#1]\l__document_structure_section_level_int{#2}
6800 \str_if_empty:NF \l__document_structure_sfragment_id_str {
6801   \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
6802 }
6803 }% for customization
6804 {}

```

and finally, we localize the sections

```

6805 \newcommand\omdoc@part@kw{Part}
6806 \newcommand\omdoc@chapter@kw{Chapter}
6807 \newcommand\omdoc@section@kw{Section}
6808 \newcommand\omdoc@subsection@kw{Subsection}
6809 \newcommand\omdoc@subsubsection@kw{Subsubsection}
6810 \newcommand\omdoc@paragraph@kw{paragraph}
6811 \newcommand\omdoc@subparagraph@kw{subparagraph}

```

## 36.3 Front and Backmatter

Index markup is provided by the `omtext` package [Kohlhase:smmtf:git], so in the `document-structure` package we only need to supply the corresponding `\printindex` command, if it is not already defined

`\printindex`

```

6812 \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).

```

6813 \cs_if_exist:NTF\frontmatter{
6814   \let\__document_structure_orig_frontmatter\frontmatter
6815   \let\frontmatter\relax
6816 }{
6817   \tl_set:Nn\__document_structure_orig_frontmatter{
6818     \clearpage
6819     \@mainmatterfalse
6820     \pagenumbering{roman}
6821   }
6822 }
6823 \cs_if_exist:NTF\backmatter{
6824   \let\__document_structure_orig_backmatter\backmatter
6825   \let\backmatter\relax
6826 }{
6827   \tl_set:Nn\__document_structure_orig_backmatter{
6828     \clearpage
6829     \@mainmatterfalse
6830     \pagenumbering{roman}
6831   }

```

```
6832 }
```

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.

```
6833 \newenvironment{frontmatter}{
6834   \__document_structure_orig_frontmatter
6835 }{
6836   \cs_if_exist:NTF\mainmatter{
6837     \mainmatter
6838   }{
6839     \clearpage
6840     \@mainmattertrue
6841     \pagenumbering{arabic}
6842   }
6843 }
```

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

```
6844 \newenvironment{backmatter}{
6845   \__document_structure_orig_backmatter
6846 }{
6847   \cs_if_exist:NTF\mainmatter{
6848     \mainmatter
6849   }{
6850     \clearpage
6851     \@mainmattertrue
6852     \pagenumbering{arabic}
6853   }
6854 }
```

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

```
6855 \@mainmattertrue\pagenumbering{arabic}
```

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

```
6856 \def \c__document_structure_document_str{document}
6857 \newcommand\afterprematurestop{}
6858 \def\prematurestop@endsfragment{
6859   \unless\ifx\@currenvir\c__document_structure_document_str
6860     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
6861       \expandafter\prematurestop@endsfragment
6862     }fi
6863 }
6864 \providecommand\prematurestop{
6865   \message{Stopping~sTeX~processing~prematurely}
6866   \prematurestop@endsfragment
6867   \afterprematurestop
6868   \end{document}
6869 }
```

(End definition for `\prematurestop`. This function is documented on page 54.)

## 36.4 Global Variables

**\setSGvar** set a global variable

```
6870 \RequirePackage{etoolbox}
6871 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}
```

*(End definition for \setSGvar. This function is documented on page 54.)*

**\useSGvar** use a global variable

```
6872 \newrobustcmd\useSGvar[1]{%
6873   \@ifundefined{sTeX@Gvar@#1}
6874   {\PackageError{document-structure}
6875     {The sTeX Global variable #1 is undefined}
6876     {set it with \protect\setSGvar}}
6877   \@nameuse{sTeX@Gvar@#1}}
```

*(End definition for \useSGvar. This function is documented on page 54.)*

**\ifSGvar** execute something conditionally based on the state of the global variable.

```
6878 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
6879   \@ifundefined{sTeX@Gvar@#1}
6880   {\PackageError{document-structure}
6881     {The sTeX Global variable #1 is undefined}
6882     {set it with \protect\setSGvar}}
6883   {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}
```

*(End definition for \ifSGvar. This function is documented on page 54.)*



## Chapter 37

# NotesSlides – Implementation

### 37.1 Class and Package Options

We define some Package Options and switches for the `notesslides` class and activate them by passing them on to `beamer.cls` and `omdoc.cls` and the `notesslides` package. We pass the `nontheorem` option to the `statements` package when we are not in notes mode, since the `beamer` package has its own (overlay-aware) theorem environments.

```
6884 \*cls)
6885 \@@=notesslides)
6886 \ProvidesExplClass{notesslides}{2022/05/24}{3.1.0}{notesslides Class}
6887 \RequirePackage{13keys2e}
6888
6889 \keys_define:nn{notesslides / cls}{
6890   class .str_set_x:N = \c__notesslides_class_str,
6891   notes .bool_set:N = \c__notesslides_notes_bool ,
6892   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
6893   docopt .str_set_x:N = \c__notesslides_docopt_str,
6894   unknown .code:n = {
6895     \PassOptionsToPackage{\CurrentOption}{document-structure}
6896     \PassOptionsToClass{\CurrentOption}{beamer}
6897     \PassOptionsToPackage{\CurrentOption}{notesslides}
6898     \PassOptionsToPackage{\CurrentOption}{stex}
6899   }
6900 }
6901 \ProcessKeysOptions{ notesslides / cls }
6902
6903 \str_if_empty:NF \c__notesslides_class_str {
6904   \PassOptionsToPackage{class=\c__notesslides_class_str}{document-structure}
6905 }
6906
6907 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{book}{
6908   \PassOptionsToPackage{defaultttopsect=part}{notesslides}
6909 }
6910 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{report}{
6911   \PassOptionsToPackage{defaultttopsect=part}{notesslides}
6912 }
6913
6914 \RequirePackage{stex}
```

```

6915 \stex_html_backend:T {
6916   \bool_set_true:N\c__notesslides_notes_bool
6917 }
6918
6919 \bool_if:NTF \c__notesslides_notes_bool {
6920   \PassOptionsToPackage{notes=true}{notesslides}
6921   \message{notesslides.cls:~Formatting~course~materials~in~notes~mode}
6922 }{
6923   \PassOptionsToPackage{notes=false}{notesslides}
6924   \message{notesslides.cls:~Formatting~course~materials~in~slides~mode}
6925 }
6926 </cls>

```

now we do the same for the notesslides package.

```

6927 <*package>
6928 \ProvidesExplPackage{notesslides}{2022/05/24}{3.1.0}{notesslides Package}
6929 \RequirePackage{l3keys2e}
6930
6931 \keys_define:nn{notesslides / pkg}{
6932   topsect          .str_set_x:N = \c__notesslides_topsect_str,
6933   defaulttopsect   .str_set_x:N = \c__notesslides_defaulttopsec_str,
6934   notes            .bool_set:N = \c__notesslides_notes_bool ,
6935   slides           .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
6936   sectocframes     .bool_set:N = \c__notesslides_sectocframes_bool ,
6937   frameimages      .bool_set:N = \c__notesslides_frameimages_bool ,
6938   fiboxed          .bool_set:N = \c__notesslides_fiboxed_bool ,
6939   nopproblems      .bool_set:N = \c__notesslides_nopproblems_bool,
6940   unknown          .code:n      = {
6941     \PassOptionsToClass{\CurrentOption}{stex}
6942     \PassOptionsToClass{\CurrentOption}{tikzinput}
6943   }
6944 }
6945 \ProcessKeysOptions{ notesslides / pkg }
6946
6947 \RequirePackage{stex}
6948 \stex_html_backend:T {
6949   \bool_set_true:N\c__notesslides_notes_bool
6950 }
6951
6952 \newif\ifnotes
6953 \bool_if:NTF \c__notesslides_notes_bool {
6954   \notesttrue
6955 }{
6956   \notesfalse
6957 }
6958

```

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

```

6959 \str_if_empty:NTF \c__notesslides_topsect_str {
6960   \str_set_eq:NN \__notesslides_topsect \c__notesslides_defaulttopsec_str
6961 }{
6962   \str_set_eq:NN \__notesslides_topsect \c__notesslides_topsect_str
6963 }
6964 \PassOptionsToPackage{topsect=\__notesslides_topsect}{document-structure}

```

```
6965 </package>
```

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

```
6966 <*cls>
6967 \bool_if:NTF \c__notesslides_notes_bool {
6968   \str_if_empty:NT \c__notesslides_class_str {
6969     \str_set:Nn \c__notesslides_class_str {article}
6970   }
6971   \exp_after:wN\LoadClass\exp_after:wN[\c__notesslides_dcopt_str]
6972     {\c__notesslides_class_str}
6973 }{
6974   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
6975   \newcounter{Item}
6976   \newcounter{paragraph}
6977   \newcounter{subparagraph}
6978   \newcounter{Hfootnote}
6979 }
6980 \RequirePackage{document-structure}
```

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

```
6981 \RequirePackage{notesslides}
6982 </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  $\text{\TeX}$  packages. The first batch of packages we want are loaded on `notesslides.sty`. These are the general ones, we will load the  $\text{\TeX}$ -specific ones after we have done some work (e.g. defined the counters `m*`). Only the `stex-logo` package is already needed now for the default theme.

```
6983 <*package>
6984 \bool_if:NT \c__notesslides_notes_bool {
6985   \RequirePackage{a4wide}
6986   \RequirePackage{marginnote}
6987   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
6988   \RequirePackage{mdframed}
6989   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
6990   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
6991 }
6992 \RequirePackage{stex-tikzinput}
6993 \RequirePackage{comment}
6994 \RequirePackage{url}
6995 \RequirePackage{graphicx}
6996 \RequirePackage{pgf}
6997 \RequirePackage{bookmark}
```

## 37.2 Notes and Slides

For the lecture notes cases, we also provide the `\usetheme` macro that would otherwise come from the `beamer` class.

```
6998 \bool_if:NT \c__notesslides_notes_bool {
6999   \renewcommand\usetheme[2][\usepackage{#1}{beamertheme#2}]
7000 }
```

```

7001 \NewDocumentCommand \libusetheme {0{} m} {
7002   \libusepackage[#1]{beamertheme#2}
7003 }
7004

```

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

```

7005 \newcounter{slide}
7006 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
7007 \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.

```

7008 \bool_if:NTF \c__notesslides_notes_bool {
7009   \renewenvironment{note}{\ignorespaces}{}
7010 }{
7011   \excludecomment{note}
7012 }

```

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.

```

7013 \bool_if:NT \c__notesslides_notes_bool {
7014   \newlength{\slideframewidth}
7015   \setlength{\slideframewidth}{1.5pt}

```

**frame** We first define the keys.

```

7016 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
7017   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
7018     \bool_set_true:N #1
7019   }{
7020     \bool_set_false:N #1
7021   }
7022 }
7023 \keys_define:nn{notesslides / frame}{
7024   label .str_set_x:N = \l__notesslides_frame_label_str,
7025   allowframebreaks .code:n = {
7026     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
7027   },
7028   allowdisplaybreaks .code:n = {
7029     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
7030   },
7031   fragile .code:n = {
7032     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
7033   },
7034   shrink .code:n = {
7035     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
7036   },
7037   squeeze .code:n = {
7038     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
7039   },
7040   t .code:n = {

```

```

7041     \_notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
7042   },
7043   unknown    .code:n      = {}
7044 }
7045 \cs_new_protected:Nn \_notesslides_frame_args:n {
7046   \str_clear:N \l__notesslides_frame_label_str
7047   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
7048   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
7049   \bool_set_true:N \l__notesslides_frame_fragile_bool
7050   \bool_set_true:N \l__notesslides_frame_shrink_bool
7051   \bool_set_true:N \l__notesslides_frame_squeeze_bool
7052   \bool_set_true:N \l__notesslides_frame_t_bool
7053   \keys_set:nn { notesslides / frame }{ #1 }
7054 }

```

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

```

7055 \renewenvironment{frame}[1][]{
7056   \_notesslides_frame_args:n{#1}
7057   \sffamily
7058   \stepcounter{slide}
7059   \def\@currentlabel{\theslide}
7060   \str_if_empty:NF \l__notesslides_frame_label_str {
7061     \label{\l__notesslides_frame_label_str}
7062   }

```

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

```

7063   \def\itemize@level{outer}
7064   \def\itemize@outer{outer}
7065   \def\itemize@inner{inner}
7066   \renewcommand\newpage{\addtocounter{framenum}{1}}
7067   %\newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
7068   \renewenvironment{itemize}{
7069     \ifx\itemize@level\itemize@outer
7070       \def\itemize@label{$\rhd$}
7071     \fi
7072     \ifx\itemize@level\itemize@inner
7073       \def\itemize@label{$\scriptstyle\rhd$}
7074     \fi
7075     \begin{list}
7076       {\itemize@label}
7077       {\setlength{\labelsep}{.3em}
7078        \setlength{\labelwidth}{.5em}
7079        \setlength{\leftmargin}{1.5em}
7080       }
7081     \edef\itemize@level{\itemize@inner}
7082   }{
7083     \end{list}
7084   }

```

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

```

7085   \stex_html_backend:TF {
7086     \begin{stex_annotate_env}{frame}{}\vbox\bgroup
7087     \mdf@patchamsthm
7088   }{
7089     \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwid

```

```

7090     }
7091   }{
7092     \stex_html_backend:TF {
7093       \miko@slidelabel\egroup\end{stex_annotate_env}
7094     }\medskip\miko@slidelabel\end{mdframed}}
7095   }

```

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

**\frametitle**

```

7096   \renewcommand{\frametitle}[1]{
7097     \stex_document_title:n { #1 }
7098     {\Large\bf\sf\color{blue}{#1}}\medskip
7099   }
7100 }

```

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

EdN:10

**\pause** 10

```

7101 \bool_if:NT \c__notesslides_notes_bool {
7102   \newcommand\pause{}
7103 }

```

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

**nparagraph**

```

7104 \bool_if:NTF \c__notesslides_notes_bool {
7105   \newenvironment{nparagraph}[1] [] {\begin{sparagraph}[#1]}\end{sparagraph}}
7106 }{
7107   \excludecomment{nparagraph}
7108 }

```

**nfragment**

```

7109 \bool_if:NTF \c__notesslides_notes_bool {
7110   \newenvironment{nfragment}[2] [] {\begin{sfragment}[#1]{#2}}\end{sfragment}}
7111 }{
7112   \excludecomment{nfragment}
7113 }

```

**ndefinition**

```

7114 \bool_if:NTF \c__notesslides_notes_bool {
7115   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}\end{sdefinition}}
7116 }{
7117   \excludecomment{ndefinition}
7118 }

```

**nassertion**

```

7119 \bool_if:NTF \c__notesslides_notes_bool {
7120   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}\end{sassertion}}
7121 }{
7122   \excludecomment{nassertion}
7123 }

```

---

<sup>10</sup>EDNOTE: MK: fake it in notes mode for now

nsproof

```
7124 \bool_if:NTF \c__notesslides_notes_bool {
7125   \newenvironment{nproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
7126 }{
7127   \excludecomment{nproof}
7128 }
```

nexample

```
7129 \bool_if:NTF \c__notesslides_notes_bool {
7130   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}{\end{sexample}}
7131 }{
7132   \excludecomment{nexample}
7133 }
```

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

```
7134 \def\inputref@preskip{\smallskip}
7135 \def\inputref@postskip{\medskip}
```

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

\inputref\*

```
7136 \let\orig@inputref\inputref
7137 \def\inputref{\@ifstar\ninputref\orig@inputref}
7138 \newcommand\ninputref[2] []{
7139   \bool_if:NT \c__notesslides_notes_bool {
7140     \orig@inputref[#1]{#2}
7141   }
7142 }
```

(End definition for \inputref\*. This function is documented on page 56.)

## 37.3 Header and Footer Lines

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

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

```
7143 \newlength{\slidelogoheight}
7144
7145 \RequirePackage{graphicx}
7146
7147 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7148 \providecommand\mhgraphics[2] []{
7149   \def\Gin@mhrepos{}\setkeys{Gin}{#1}
7150   \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}
7151 }
7152
7153 \bool_if:NTF \c__notesslides_notes_bool {
7154   \setlength{\slidelogoheight}{.4cm}
7155 }{
7156   \setlength{\slidelogoheight}{.25cm}
7157 }
```

```

7158 \ifcsname slidelogo\endcsname\else
7159   \newsavebox{\slidelogo}
7160   \sbox{\slidelogo}{\sTeX}
7161 \fi
7162 \newrobustcmd{\setslidelogo}[2][]{
7163   \tl_if_empty:nTF{#1}{
7164     \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#2}}
7165   }{
7166     \sbox{\slidelogo}{\mhgraphics[height=\slidelogoheight,mhrepos=#1]{#2}}
7167   }
7168 }

```

(End definition for `\setslidelogo`. This function is documented on page 57.)

**\author** In notes mode, we redefine the `\author` macro so that it does not disregard the optional argument (as `beamerarticle` does). We want to use it to set the source later.

```

7169 \bool_if:NT \c__notesslides_notes_bool {
7170   \def\author{\@dblarg\@ns@author}
7171   \long\def\@ns@author[#1]#2{%
7172     \def\c__notesslides_shortauthor{#1}%
7173     \def\@author{#2}
7174   }
7175 }

```

(End definition for `\author`. 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.

```

7176 \newrobustcmd{\setsource}[1]{\def\source{#1}}

```

(End definition for `\setsource`. This function is documented on page 57.)

**\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.

```

7177 \def\copyrightnotice{%
7178   \footnotesize\copyright : \hspace{.3ex}%
7179   \ifcsname source\endcsname\source\else%
7180   \ifcsname c__notesslides_shortauthor\endcsname\c__notesslides_shortauthor\else%
7181   \PackageWarning{notesslides}{Author/Source-undefined-in-copyright-notice}%
7182   ?source/author?\fi%
7183 \fi}
7184 \newsavebox{\cclogo}
7185 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
7186 \newif\ifcchref\cchreffalse
7187 \AtBeginDocument{
7188   \@ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
7189 }
7190 \def\licensing{
7191   \ifcchref
7192     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
7193   \else
7194     {\usebox{\cclogo}}

```



```

7195 \fi
7196 }
7197 \newrobustcmd{\setlicensing}[2][]{
7198   \def\@url{#1}
7199   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
7200   \ifx\@url\@empty
7201     \def\licensing{\usebox{\cclogo}}
7202   \else
7203     \def\licensing{
7204       \ifcchref
7205         \href{#1}{\usebox{\cclogo}}
7206       \else
7207         {\usebox{\cclogo}}
7208       \fi
7209     }
7210   \fi
7211 }

```

(End definition for \setlicensing. This function is documented on page 57.)

EdN:11

```

\slidelabel Now, we set up the slide label for the article mode.11
7212 \newrobustcmd\miko@slidelabel{
7213   \vbox to \slidelogoheight{
7214     \vss\hbox to \slidewidth
7215       {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
7216   }
7217 }

```

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

## 37.4 Frame Images

**\frameimage** We have to make sure that the width is overwritten, for that we check the \Gin@ewidth macro from the graphicx package. We also add the label key.

```

7218 \def\Gin@mhrepos{}
7219 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7220 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
7221 \newrobustcmd\frameimage[2][]{
7222   \stepcounter{slide}
7223   \bool_if:NT \c__notesslides_frameimages_bool {
7224     \def\Gin@ewidth{}\setkeys{Gin}{#1}
7225     \bool_if:NF \c__notesslides_notes_bool { \vfill }
7226     \begin{center}
7227       \bool_if:NTF \c__notesslides_fiboxed_bool {
7228         \fbox{
7229           \ifx\Gin@ewidth\@empty
7230             \ifx\Gin@mhrepos\@empty
7231               \mhgraphics[width=\slidewidth,#1]{#2}
7232             \else
7233               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7234             \fi
7235           \else% Gin@ewidth empty

```

<sup>11</sup>EdNOTE: see that we can use the themes for the slides some day. This is all fake.

```

7236         \ifx\Gin@mhrepos\@empty
7237         \mhgraphics[#1]{#2}
7238     \else
7239         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7240     \fi
7241 \fi% Gin@ewidth empty
7242 }
7243 }{
7244     \ifx\Gin@ewidth\@empty
7245     \ifx\Gin@mhrepos\@empty
7246         \mhgraphics[width=\slidewidth,#1]{#2}
7247     \else
7248         \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7249     \fi
7250     \ifx\Gin@mhrepos\@empty
7251         \mhgraphics[#1]{#2}
7252     \else
7253         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7254     \fi
7255     \fi% Gin@ewidth empty
7256 }
7257 \end{center}
7258 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
7259 \bool_if:NF \c__notesslides_notes_bool { \vfill }
7260 }
7261 } % ifmks@sty@frameimages

```

(End definition for `\frameimage`. This function is documented on page 57.)

## 37.5 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`.

```

7262 \stex_html_backend:F {
7263     \bool_if:NT \c__notesslides_sectocframes_bool {
7264         \str_if_eq:VnTF \__notesslidesstopsect{part}{
7265             \newcounter{chapter}\counterwithin*{section}{chapter}
7266         }{
7267             \str_if_eq:VnT\__notesslidesstopsect{chapter}{
7268                 \newcounter{chapter}\counterwithin*{section}{chapter}
7269             }
7270         }
7271     }
7272 }

```

`\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
7273 \def\part@prefix{}
7274 \@ifpackageloaded{document-structure}{
7275     \str_case:VnF \__notesslidesstopsect {

```

```

7276 {part}{
7277   \int_set:Nn \l_document_structure_section_level_int {0}
7278   \def\thesection{\arabic{chapter}.\arabic{section}}
7279   \def\part@prefix{\arabic{chapter}.}
7280 }
7281 {chapter}{
7282   \int_set:Nn \l_document_structure_section_level_int {1}
7283   \def\thesection{\arabic{chapter}.\arabic{section}}
7284   \def\part@prefix{\arabic{chapter}.}
7285 }
7286 }{
7287   \int_set:Nn \l_document_structure_section_level_int {2}
7288   \def\part@prefix{}
7289 }
7290 }
7291
7292 \bool_if:NF \c__notesslides_notes_bool { % only in slides

```

(End definition for \section@level. This function is documented on page ??.)

The new counters are used in the `sfragment` environment that chooses the L<sup>A</sup>T<sub>E</sub>X sectioning macros according to `\section@level`.

#### `sfragment`

```

7293 \renewenvironment{sfragment}[2][]{
7294   \__document_structure_sfragment_args:n { #1 }
7295   \int_incr:N \l_document_structure_section_level_int
7296   \bool_if:NT \c__notesslides_sectocframes_bool {
7297     \stepcounter{slide}
7298     \begin{frame}[noframenumbering]
7299     \vfill\Large\centering
7300     \red{
7301       \ifcase\l_document_structure_section_level_int\or
7302         \stepcounter{part}
7303         \def\__notesslideslabel{\omdoc@part@kw}~\Roman{part}}
7304         \addcontentsline{toc}{part}{\protect\numberline{\thepart}#2}
7305         \pdfbookmark[0]{\thepart\ #2}{part.\thepart}
7306         \def\currentsectionlevel{\omdoc@part@kw}
7307       \or
7308         \stepcounter{chapter}
7309         \def\__notesslideslabel{\omdoc@chapter@kw}~\arabic{chapter}}
7310         \addcontentsline{toc}{chapter}{\protect\numberline{\thechapter}#2}
7311         \pdfbookmark[1]{\thechapter\ #2}{chapter.\cs_if_exist:cT{\thepart}\thepart.\thechapter}
7312         \def\currentsectionlevel{\omdoc@chapter@kw}
7313       \or
7314         \stepcounter{section}
7315         \def\__notesslideslabel{\part@prefix\arabic{section}}
7316         \addcontentsline{toc}{section}{\protect\numberline{\thesection}#2}
7317         \pdfbookmark[2]{\cs_if_exist:cT{\thechapter}{\thechapter}.\thesection\ #2}
7318         {section.\cs_if_exist:cT{\thepart}{\thepart}.\cs_if_exist:cT{\thechapter}{\thechapter}.\thepart}
7319         \def\currentsectionlevel{\omdoc@section@kw}
7320       \or
7321         \stepcounter{subsection}
7322         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
7323         \addcontentsline{toc}{subsection}{\protect\numberline{\thesubsection}#2}

```

```

7324         \pdfbookmark[3]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection
7325         {subsection.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection}
7326         \def\currentsectionlevel{\omdoc@subsection@kw}
7327     \or
7328         \stepcounter{subsubsection}
7329         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
7330         \addcontentsline{toc}{subsubsection}{\protect\numberline{\thesubsubsection}#2}
7331         \pdfbookmark[4]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection.\thesubsubsection}
7332         {subsubsection.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsubsection}
7333         \def\currentsectionlevel{\omdoc@subsubsection@kw}
7334     \or
7335         \stepcounter{paragraph}
7336         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{paragraph}}
7337         \addcontentsline{toc}{paragraph}{\protect\numberline{\theparagraph}#2}
7338         \pdfbookmark[5]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection.\theparagraph}
7339         {paragraph.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\theparagraph}
7340         \def\currentsectionlevel{\omdoc@paragraph@kw}
7341     \else
7342         \def\__notesslideslabel{}
7343         \def\currentsectionlevel{\omdoc@paragraph@kw}
7344     \fi% end ifcase
7345     \__notesslideslabel\quad #2%
7346 }%
7347 \vfill%
7348 \end{frame}%
7349 }
7350 \str_if_empty:NF \l__document_structure_sfragment_id_str {
7351     \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
7352 }
7353 }{}
7354 }

```

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

```

7355 \def\inserttheorembodyfont{\normalfont}
7356 %\bool_if:NF \c__notesslides_notes_bool {
7357 %   \defbeamertemplate{theorem begin}{miko}
7358 %   {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
7359 %    \ifx\inserttheoremaddition@empty\else\ (\inserttheoremaddition)\fi%
7360 %    \inserttheorempunctuation\inserttheorembodyfont\xspace}
7361 %   \defbeamertemplate{theorem end}{miko}{}}

```

and we set it as the default one.

```

7362 % \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.

```

7363 % \expandafter\def\csname Parent2\endcsname{}
7364 %}
7365
7366 \AddToHook{begindocument}{ % this does not work for some reason
7367     \setbeamertemplate{theorems}[ams style]
7368 }
7369 \bool_if:NT \c__notesslides_notes_bool {
7370     \renewenvironment{columns}[1][{}]{%

```

```

7371 \par\noindent%
7372 \begin{minipage}%
7373 \slidewidth\centering\leavevmode%
7374 }{%
7375 \end{minipage}\par\noindent%
7376 }%
7377 \newsavebox\columnbox%
7378 \renewenvironment<>{column}[2][]{%
7379 \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
7380 }{%
7381 \end{minipage}\end{lrbox}\usebox\columnbox%
7382 }%
7383 }

7384 \bool_if:NTF \c__notesslides_noproblems_bool {
7385 \newenvironment{problems}{}{}
7386 }{
7387 \excludacomment{problems}
7388 }

```

## 37.6 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.

```

7389 \gdef\printexcursions{}
7390 \newcommand\excursionref[2]{% label, text
7391 \bool_if:NT \c__notesslides_notes_bool {
7392 \begin{sparagraph}[title=Excursion]
7393 #2 \sref[fallback=the appendix]{#1}.
7394 \end{sparagraph}
7395 }
7396 }
7397 \newcommand\activate@excursion[2][{}{
7398 \gappto\printexcursions{\inputref{#1}{#2}}
7399 }
7400 \newcommand\excursion[4][{}{ repos, label, path, text
7401 \bool_if:NT \c__notesslides_notes_bool {
7402 \activate@excursion{#1}{#3}\excursionref{#2}{#4}
7403 }
7404 }

```

(End definition for `\excursion`. This function is documented on page 58.)

**\excursiongroup**

```

7405 \keys_define:nn{notesslides / excursiongroup }{
7406 id .str_set_x:N = \l__notesslides_excursion_id_str,
7407 intro .tl_set:N = \l__notesslides_excursion_intro_tl,
7408 mhrepos .str_set_x:N = \l__notesslides_excursion_mhrepos_str
7409 }
7410 \cs_new_protected:Nn \__notesslides_excursion_args:n {
7411 \tl_clear:N \l__notesslides_excursion_intro_tl
7412 \str_clear:N \l__notesslides_excursion_id_str

```

```

7413 \str_clear:N \l__notesslides_excursion_mhrepos_str
7414 \keys_set:nn {notesslides / excursionsgroup }{ #1 }
7415 }
7416 \newcommand\excursionsgroup[1][ ]{
7417   \__notesslides_excursion_args:n{ #1 }
7418   \ifdefempty\printexcursions{}% only if there are excursions
7419   {\begin{note}
7420     \begin{sfragment}[#1]{Excursions}%
7421     \ifdefempty\l__notesslides_excursion_intro_tl}{
7422       \inputref[\l__notesslides_excursion_mhrepos_str]{
7423         \l__notesslides_excursion_intro_tl
7424       }
7425     }
7426     \printexcursions%
7427     \end{sfragment}
7428   \end{note}}
7429 }
7430 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
7431 \end{package}

```

(End definition for \excursionsgroup. This function is documented on page 58.)

## Chapter 38

# The Implementation

### 38.1 Package Options

The first step is to declare (a few) package options that handle whether certain information is printed or not. They all come with their own conditionals that are set by the options.

```
7432 <*package>
7433 <@@=problems>
7434 \ProvidesExplPackage{problem}{2022/05/24}{3.1.0}{Semantic Markup for Problems}
7435 \RequirePackage{13keys2e}
7436 \RequirePackage{amssymb}% for \Box
7437
7438 \keys_define:nn { problem / pkg }{
7439   notes      .default:n    = { true },
7440   notes      .bool_set:N   = \c__problems_notes_bool,
7441   gnotes     .default:n    = { true },
7442   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
7443   hints      .default:n    = { true },
7444   hints      .bool_set:N   = \c__problems_hints_bool,
7445   solutions  .default:n    = { true },
7446   solutions  .bool_set:N   = \c__problems_solutions_bool,
7447   pts        .default:n    = { true },
7448   pts        .bool_set:N   = \c__problems_pts_bool,
7449   min        .default:n    = { true },
7450   min        .bool_set:N   = \c__problems_min_bool,
7451   boxed      .default:n    = { true },
7452   boxed      .bool_set:N   = \c__problems_boxed_bool,
7453   unknown    .code:n       = {
7454     \PassOptionsToPackage{\CurrentOption}{stex}
7455   }
7456 }
7457 \newif\ifsolutions
7458
7459 \ProcessKeysOptions{ problem / pkg }
7460 \bool_if:NTF \c__problems_solutions_bool {
7461   \solutionstrue
7462 }{
7463   \solutionsfalse
```

```

7464 }
7465 \RequirePackage{stex}

```

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

```

7466 \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.

```

7467 \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.

```

7468 \def\prob@problem@kw{Problem}
7469 \def\prob@solution@kw{Solution}
7470 \def\prob@hint@kw{Hint}
7471 \def\prob@note@kw{Note}
7472 \def\prob@gnote@kw{Grading}
7473 \def\prob@pt@kw{pt}
7474 \def\prob@min@kw{min}
7475 \def\prob@correct@kw{Correct}
7476 \def\prob@wrong@kw{Wrong}

```

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

For the other languages, we set up triggers

```

7477 \AddToHook{begindocument}{
7478   \ltx@ifpackageloaded{babel}{
7479     \makeatletter
7480     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7481     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
7482       \input{problem-ngerman.ldf}
7483     }
7484     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
7485       \input{problem-finnish.ldf}
7486     }
7487     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
7488       \input{problem-french.ldf}
7489     }
7490     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
7491       \input{problem-russian.ldf}
7492     }
7493     \makeatother
7494   }{}
7495 }

```

## 38.2 Problems and Solutions

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

```

7496 \keys_define:nn{ problem / problem }{
7497   id      .str_set_x:N = \l__problems_prob_id_str,
7498   pts     .tl_set:N    = \l__problems_prob_pts_tl,
7499   min     .tl_set:N    = \l__problems_prob_min_tl,

```



```

7500 title .tl_set:N = \l__problems_prob_title_tl,
7501 type .tl_set:N = \l__problems_prob_type_tl,
7502 imports .tl_set:N = \l__problems_prob_imports_tl,
7503 name .str_set_x:N = \l__problems_prob_name_str,
7504 refnum .int_set:N = \l__problems_prob_refnum_int
7505 }
7506 \cs_new_protected:Nn \__problems_prob_args:n {
7507 \str_clear:N \l__problems_prob_id_str
7508 \str_clear:N \l__problems_prob_name_str
7509 \tl_clear:N \l__problems_prob_pts_tl
7510 \tl_clear:N \l__problems_prob_min_tl
7511 \tl_clear:N \l__problems_prob_title_tl
7512 \tl_clear:N \l__problems_prob_type_tl
7513 \tl_clear:N \l__problems_prob_imports_tl
7514 \int_zero_new:N \l__problems_prob_refnum_int
7515 \keys_set:nn { problem / problem }{ #1 }
7516 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
7517 \let\l__problems_prob_refnum_int\undefined
7518 }
7519 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

7520 \newcounter{problem}[section]
7521 \newcommand\numberproblemsin[1]{\@addtoreset{problem}{#1}}
7522 \def\theplainsproblem{\arabic{problem}}
7523 \def\thesproblem{\thesection.\theplainsproblem}

```

(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.

```

7524 \newcommand\prob@label[1]{\thesection.#1}

```

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

`\prob@number` We consolidate the problem number into a reusable internal macro

```

7525 \newcommand\prob@number{
7526 \int_if_exist:NTF \l__problems_inclprob_refnum_int {
7527 \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
7528 }{
7529 \int_if_exist:NTF \l__problems_prob_refnum_int {
7530 \prob@label{\int_use:N \l__problems_prob_refnum_int }
7531 }{
7532 \prob@label\theplainsproblem
7533 }
7534 }
7535 }
7536 \def\sproblemautorefname{\prob@problem@kw}

```

(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.

```

7537 \newcommand\prob@title[3]{%
7538   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
7539     #2 \l__problems_inclprob_title_tl #3
7540   }{
7541     \tl_if_empty:NTF \l__problems_prob_title_tl {
7542       #1
7543     }{
7544       #2 \l__problems_prob_title_tl #3
7545     }
7546   }
7547 }

```

(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.

```

7548 \def\prob@heading{
7549   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
7550   %\sref@label{id{\prob@problem@kw~\prob@number}}{~}
7551 }

```

(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**

```

7552 \newenvironment{sproblem}[1][]{
7553   \__problems_prob_args:n{#1}%\sref@target%
7554   \@in@omtexttrue% we are in a statement (for inline definitions)
7555   \refstepcounter{sproblem}\record@problem
7556   \def\current@section@level{\prob@problem@kw}
7557
7558   \str_if_empty:NT \l__problems_prob_name_str {
7559     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
7560     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
7561     \seq_get_left:NN \l_tmpa_seq \l__problems_prob_name_str
7562   }
7563
7564   \stex_if_do_html:T{
7565     \tl_if_empty:NF \l__problems_prob_title_tl {
7566       \exp_args:No \stex_document_title:n \l__problems_prob_title_tl
7567     }
7568   }
7569
7570   \exp_args:Nno\stex_module_setup:nn{type=problem}\l__problems_prob_name_str
7571
7572   \stex_reactivate_macro:N \STEXexport
7573   \stex_reactivate_macro:N \importmodule
7574   \stex_reactivate_macro:N \symdecl
7575   \stex_reactivate_macro:N \notation
7576   \stex_reactivate_macro:N \symdef

```

```

7577
7578 \stex_if_do_html:T{
7579   \begin{stex_annotate_env} {problem} {
7580     \l_stex_module_ns_str ? \l_stex_module_name_str
7581   }
7582
7583   \stex_annotate_invisible:nnn{header}{} {
7584     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
7585     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
7586     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
7587       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
7588     }
7589   }
7590 }
7591
7592 \stex_csl_to_imports:No \importmodule \l__problems_prob_imports_tl
7593
7594
7595 \tl_if_exist:NTF \l__problems_inclprob_type_tl {
7596   \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
7597 }{
7598   \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
7599 }
7600 \str_if_exist:NTF \l__problems_inclprob_id_str {
7601   \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
7602 }{
7603   \str_set_eq:NN \sproblemid \l__problems_prob_id_str
7604 }
7605
7606
7607 \stex_if_smsmode:F {
7608   \clist_set:No \l_tmpa_clist \sproblemtype
7609   \tl_clear:N \l_tmpa_tl
7610   \clist_map_inline:Nn \l_tmpa_clist {
7611     \tl_if_exist:cT {__problems_sproblem_##1_start:}{
7612       \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_start:}}
7613     }
7614   }
7615   \tl_if_empty:NTF \l_tmpa_tl {
7616     \__problems_sproblem_start:
7617   }{
7618     \l_tmpa_tl
7619   }
7620 }
7621 \stex_ref_new_doc_target:n \sproblemid
7622 \stex_if_smsmode:TF \stex_smsmode_do: \ignorespacesandpars
7623 }{
7624   \__stex_modules_end_module:
7625   \stex_if_smsmode:F{
7626     \clist_set:No \l_tmpa_clist \sproblemtype
7627     \tl_clear:N \l_tmpa_tl
7628     \clist_map_inline:Nn \l_tmpa_clist {
7629       \tl_if_exist:cT {__problems_sproblem_##1_end:}{
7630         \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_end:}}

```

```

7631     }
7632   }
7633   \tl_if_empty:NTF \l_tmpa_tl {
7634     \__problems_sproblem_end:
7635   }{
7636     \l_tmpa_tl
7637   }
7638 }
7639 \stex_if_do_html:T{
7640   \end{stex_annotate_env}
7641 }
7642
7643 \smallskip
7644 }
7645
7646 \seq_put_right:Nx\g_stex_smsmode_allowedenvs_seq{\tl_to_str:n{sproblem}}
7647
7648
7649
7650 \cs_new_protected:Nn \__problems_sproblem_start: {
7651   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
7652 }
7653 \cs_new_protected:Nn \__problems_sproblem_end: { \par\smallskip}
7654
7655 \newcommand\stexpatchproblem[3][] {
7656   \str_set:Nx \l_tmpa_str{ #1 }
7657   \str_if_empty:NTF \l_tmpa_str {
7658     \tl_set:Nn \__problems_sproblem_start: { #2 }
7659     \tl_set:Nn \__problems_sproblem_end: { #3 }
7660   }{
7661     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
7662     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
7663   }
7664 }
7665
7666
7667 \bool_if:NT \c__problems_boxed_bool {
7668   \surroundwithhmdframed{problem}
7669 }

```

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

```

7670 \def\record@problem{
7671   \protected@write\@auxout{}
7672   {
7673     \string\@problem{\prob@number}
7674     {
7675       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7676         \l__problems_inclprob_pts_tl
7677       }{
7678         \l__problems_prob_pts_tl
7679       }
7680     }%
7681     {
7682       \tl_if_exist:NTF \l__problems_inclprob_min_tl {

```

```

7683         \l__problems_inclprob_min_tl
7684     }{
7685         \l__problems_prob_min_tl
7686     }
7687 }
7688 }
7689 }

```

(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).

```

7690 \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.

```

7691 \keys_define:nn { problem / solution }{
7692   id          .str_set_x:N = \l__problems_solution_id_str ,
7693   for         .str_set_x:N = \l__problems_solution_for_str ,
7694   type        .str_set_x:N = \l__problems_solution_type_str ,
7695   title       .tl_set:N     = \l__problems_solution_title_tl
7696 }
7697 \cs_new_protected:Nn \__problems_solution_args:n {
7698   \str_clear:N \l__problems_solution_id_str
7699   \str_clear:N \l__problems_solution_type_str
7700   \str_clear:N \l__problems_solution_for_str
7701   \tl_clear:N \l__problems_solution_title_tl
7702   \keys_set:nn { problem / solution }{ #1 }
7703 }

```

**\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.

```

7704 \box_new:N \l__problems_solution_box
7705 \newenvironment{solution}[1][{}]{
7706   \__problems_solution_args:n{#1}
7707   \stex_html_backend:TF{
7708     \stex_if_do_html:T{
7709       \begin{stex_annotate_env}{solution}{}}
7710       \str_if_empty:NF \l__problems_solution_type_str {
7711         \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{}}
7712     }
7713     \noindent\textbf{Solution}\tl_if_empty:NF\l__problems_solution_title_tl{~(\l__problem
7714   }
7715   }{
7716     \setbox\l__problems_solution_box\vbox\bgroup
7717     \par\smallskip\hrule\smallskip
7718     \noindent\textbf{Solution}\tl_if_empty:NF\l__problems_solution_title_tl{~(\l__problems
7719   }
7720   }{
7721     \stex_html_backend:TF{
7722       \stex_if_do_html:T{
7723         \end{stex_annotate_env}

```

```

7724     }
7725   }{
7726     \smallskip\hrule
7727     \egroup
7728     \bool_if:NT \c__problems_solutions_bool {
7729       \box\l__problems_solution_box
7730     }
7731   }
7732 }
7733
7734 \newcommand\startsolutions{
7735   \bool_set_true:N \c__problems_solutions_bool
7736   \solutionstrue
7737   % \specialcomment{solution}{\@startsolution}{
7738   %   \bool_if:NF \c__problems_boxed_bool {
7739   %     \hrule\medskip
7740   %   }
7741   %   \end{small}%
7742   % }
7743   % \bool_if:NT \c__problems_boxed_bool {
7744   %   \surroundwithmdframed{solution}
7745   % }
7746 }

```

(End definition for \startsolutions. This function is documented on page 60.)

## **\stopsolutions**

```

7747 \newcommand\stopsolutions{\bool_set_false:N \c__problems_solutions_bool \solutionsfalse}%\ex

```

(End definition for \stopsolutions. This function is documented on page 60.)

## **exnote**

```

7748 \bool_if:NTF \c__problems_notes_bool {
7749   \newenvironment{exnote}[1][ ]{
7750     \par\smallskip\hrule\smallskip
7751     \noindent\textbf{\prob@note@kw :~ }\small
7752   }{
7753     \smallskip\hrule
7754   }
7755 }{
7756   \excludacomment{exnote}
7757 }

```

## **hint**

```

7758 \bool_if:NTF \c__problems_notes_bool {
7759   \newenvironment{hint}[1][ ]{
7760     \par\smallskip\hrule\smallskip
7761     \noindent\textbf{\prob@hint@kw :~ }\small
7762   }{
7763     \smallskip\hrule
7764   }
7765   \newenvironment{exhint}[1][ ]{
7766     \par\smallskip\hrule\smallskip
7767     \noindent\textbf{\prob@hint@kw :~ }\small

```

```

7768 }{
7769     \smallskip\hrule
7770 }
7771 }{
7772     \excludecomment{hint}
7773     \excludecomment{exhint}
7774 }

```

**gnote**

```

7775 \bool_if:NTF \c__problems_notes_bool {
7776     \newenvironment{gnote}[1][]{
7777         \par\smallskip\hrule\smallskip
7778         \noindent\textbf{\prob@gnote@kw :~ }\small
7779     }{
7780         \smallskip\hrule
7781     }
7782 }{
7783     \excludecomment{gnote}
7784 }

```

## 38.3 Marup for Added Value Services

## 38.4 Multiple Choice Blocks

EdN:12

**mcb**

12

```

7785 \newenvironment{mcb}{
7786     \begin{enumerate}
7787 }{
7788     \end{enumerate}
7789 }

```

we define the keys for the mcb macro

```

7790 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
7791     \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
7792         \bool_set_true:N #1
7793     }{
7794         \bool_set_false:N #1
7795     }
7796 }
7797 \keys_define:nn { problem / mcb }{
7798     id .str_set_x:N = \l__problems_mcc_id_str ,
7799     feedback .tl_set:N = \l__problems_mcc_feedback_tl ,
7800     T .default:n = { false } ,
7801     T .bool_set:N = \l__problems_mcc_t_bool ,
7802     F .default:n = { false } ,
7803     F .bool_set:N = \l__problems_mcc_f_bool ,
7804     Ttext .tl_set:N = \l__problems_mcc_Ttext_tl ,
7805     Ftext .tl_set:N = \l__problems_mcc_Ftext_tl
7806 }
7807 \cs_new_protected:Nn \l__problems_mcc_args:n {

```

---

<sup>12</sup>EdNOTE: MK: maybe import something better here from a dedicated MC package

```

7808 \str_clear:N \l__problems_mcc_id_str
7809 \tl_clear:N \l__problems_mcc_feedback_tl
7810 \bool_set_false:N \l__problems_mcc_t_bool
7811 \bool_set_false:N \l__problems_mcc_f_bool
7812 \tl_clear:N \l__problems_mcc_Ttext_tl
7813 \tl_clear:N \l__problems_mcc_Ftext_tl
7814 \str_clear:N \l__problems_mcc_id_str
7815 \keys_set:nn { problem / mcc }{ #1 }
7816 }

```

**\mcc**

```

7817 \def\mccTrueText{\textbf{\prob@correct@kw!~}}
7818 \def\mccFalseText{\textbf{\prob@wrong@kw!~}}
7819 \newcommand\mcc[2][]{
7820   \l__problems_mcc_args:n{ #1 }
7821   \item[$\Box$] #2
7822   \bool_if:NT \c__problems_solutions_bool{
7823     \\\
7824     \bool_if:NT \l__problems_mcc_t_bool {
7825       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccTrueText\l__problems_mcc_Ttext_tl
7826     }
7827     \bool_if:NT \l__problems_mcc_f_bool {
7828       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccFalseText\l__problems_mcc_Ftext_tl
7829     }
7830     \tl_if_empty:NF \l__problems_mcc_feedback_tl {
7831       \emph{\l__problems_mcc_feedback_tl}
7832     }
7833   }
7834 } %solutions

```

(End definition for \mcc. This function is documented on page 61.)

## 38.5 Filling in Concrete Solutions

**\includeproblem** This is embarrassingly simple, but can grow over time.

```

7835 \newcommand\fillinsol[1]{\quad%
7836   \ifsolutions\textcolor{red}{\#!}\else%
7837   \fbox{\phantom{\huge{\#!}}}%
7838   \fi}

```

(End definition for \includeproblem. This function is documented on page 63.)

## 38.6 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.

```

7839
7840 \keys_define:nn{ problem / inclproblem }{
7841   id      .str_set_x:N = \l__problems_inclprob_id_str,
7842   pts     .tl_set:N    = \l__problems_inclprob_pts_tl,
7843   min     .tl_set:N    = \l__problems_inclprob_min_tl,

```



```

7844 title .tl_set:N = \l__problems_inclprob_title_tl,
7845 refnum .int_set:N = \l__problems_inclprob_refnum_int,
7846 type .tl_set:N = \l__problems_inclprob_type_tl,
7847 mhrepos .str_set_x:N = \l__problems_inclprob_mhrepos_str
7848 }
7849 \cs_new_protected:Nn \l__problems_inclprob_args:n {
7850 \str_clear:N \l__problems_prob_id_str
7851 \tl_clear:N \l__problems_inclprob_pts_tl
7852 \tl_clear:N \l__problems_inclprob_min_tl
7853 \tl_clear:N \l__problems_inclprob_title_tl
7854 \tl_clear:N \l__problems_inclprob_type_tl
7855 \int_zero_new:N \l__problems_inclprob_refnum_int
7856 \str_clear:N \l__problems_inclprob_mhrepos_str
7857 \keys_set:nn { problem / inclproblem }{ #1 }
7858 \tl_if_empty:NT \l__problems_inclprob_pts_tl {
7859 \let\l__problems_inclprob_pts_tl\undefined
7860 }
7861 \tl_if_empty:NT \l__problems_inclprob_min_tl {
7862 \let\l__problems_inclprob_min_tl\undefined
7863 }
7864 \tl_if_empty:NT \l__problems_inclprob_title_tl {
7865 \let\l__problems_inclprob_title_tl\undefined
7866 }
7867 \tl_if_empty:NT \l__problems_inclprob_type_tl {
7868 \let\l__problems_inclprob_type_tl\undefined
7869 }
7870 \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
7871 \let\l__problems_inclprob_refnum_int\undefined
7872 }
7873 }
7874
7875 \cs_new_protected:Nn \l__problems_inclprob_clear: {
7876 \let\l__problems_inclprob_id_str\undefined
7877 \let\l__problems_inclprob_pts_tl\undefined
7878 \let\l__problems_inclprob_min_tl\undefined
7879 \let\l__problems_inclprob_title_tl\undefined
7880 \let\l__problems_inclprob_type_tl\undefined
7881 \let\l__problems_inclprob_refnum_int\undefined
7882 \let\l__problems_inclprob_mhrepos_str\undefined
7883 }
7884 \l__problems_inclprob_clear:
7885
7886 \newcommand\includeproblem[2][]{
7887 \l__problems_inclprob_args:n{ #1 }
7888 \exp_args:No \stex_in_repository:nn\l__problems_inclprob_mhrepos_str{
7889 \stex_html_backend:TF {
7890 \str_clear:N \l_tmpa_str
7891 \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
7892 \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
7893 }
7894 \stex_annotate_invisible:nnn{includeproblem}{
7895 \l_tmpa_str / #2
7896 }{}}
7897 }{

```

```

7898     \begingroup
7899     \inputreftrue
7900     \tl_if_empty:nTF{ ##1 }{
7901       \input{#2}
7902     }{
7903       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
7904     }
7905   \endgroup
7906 }
7907 }
7908 \__problems_inclprob_clear:
7909 }

```

(End definition for `\includeproblem`. This function is documented on page 63.)

## 38.7 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.

```

7910 \AddToHook{enddocument}{
7911   \bool_if:NT \c__problems_pts_bool {
7912     \message{Total:~\arabic{pts}~points}
7913   }
7914   \bool_if:NT \c__problems_min_bool {
7915     \message{Total:~\arabic{min}~minutes}
7916   }
7917 }

```

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

```

7918 \def\pts#1{
7919   \bool_if:NT \c__problems_pts_bool {
7920     \marginpar{#1~\prob@pt@kw}
7921   }
7922 }
7923 \def\min#1{
7924   \bool_if:NT \c__problems_min_bool {
7925     \marginpar{#1~\prob@min@kw}
7926   }
7927 }

```

`\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.

```

7928 \newcounter{pts}
7929 \def\show@pts{
7930   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7931     \bool_if:NT \c__problems_pts_bool {
7932       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
7933       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
7934     }
7935   }{
7936     \tl_if_exist:NT \l__problems_prob_pts_tl {
7937       \bool_if:NT \c__problems_pts_bool {

```

```

7938         \tl_if_empty:NT\l__problems_prob_pts_tl{
7939             \tl_set:Nn \l__problems_prob_pts_tl {0}
7940         }
7941         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
7942         \addtocounter{pts}{\l__problems_prob_pts_tl}
7943     }
7944 }
7945 }
7946 }

```

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

\show@min

```

7947 \newcounter{min}
7948 \def\show@min{
7949     \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7950         \bool_if:NT \c__problems_min_bool {
7951             \marginpar{\l__problems_inclprob_pts_tl\ min}
7952             \addtocounter{min}{\l__problems_inclprob_min_tl}
7953         }
7954     }{
7955         \tl_if_exist:NT \l__problems_prob_min_tl {
7956             \bool_if:NT \c__problems_min_bool {
7957                 \tl_if_empty:NT\l__problems_prob_min_tl{
7958                     \tl_set:Nn \l__problems_prob_min_tl {0}
7959                 }
7960                 \marginpar{\l__problems_prob_min_tl\ min}
7961                 \addtocounter{min}{\l__problems_prob_min_tl}
7962             }
7963         }
7964     }
7965 }
7966 \end{package}

```

(End definition for \show@min. This function is documented on page ??.)

## Chapter 39

# Implementation: The hwexam Package

### 39.1 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.

```
7967 \*package>
7968 \ProvidesExplPackage{hwexam}{2022/05/24}{3.1.0}{homework assignments and exams}
7969 \RequirePackage{13keys2e}
7970
7971 \newif\iftest\testfalse
7972 \DeclareOption{test}{\testtrue}
7973 \newif\ifmultiple\multiplefalse
7974 \DeclareOption{multiple}{\multipletrue}
7975 \DeclareOption{lang}{\PassOptionsToPackage{\CurrentOption}{problem}}
7976 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
7977 \ProcessOptions
```

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

```
7978 \RequirePackage{keyval}[1997/11/10]
7979 \RequirePackage{problem}
```

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

```
7980 \newcommand\hwexam@assignment@kw{Assignment}
7981 \newcommand\hwexam@given@kw{Given}
7982 \newcommand\hwexam@due@kw{Due}
7983 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~blank~for~extra~space}
7984 \newcommand\hwexam@minutes@kw{minutes}
7985 \newcommand\correction@probs@kw{prob.}
7986 \newcommand\correction@pts@kw{total}
7987 \newcommand\correction@reached@kw{reached}
7988 \newcommand\correction@sum@kw{Sum}
7989 \newcommand\correction@grade@kw{grade}
7990 \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

```

7991 \AddToHook{begindocument}{
7992 \ltx@ifpackageloaded{babel}{
7993 \makeatletter
7994 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7995 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
7996 \input{hwexam-ngerman.ldf}
7997 }
7998 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
7999 \input{hwexam-finnish.ldf}
8000 }
8001 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
8002 \input{hwexam-french.ldf}
8003 }
8004 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
8005 \input{hwexam-russian.ldf}
8006 }
8007 \makeatother
8008 }{}
8009 }
8010

```

## 39.2 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.

```

8011 \newcounter{assignment}
8012 %\numberproblemsin{assignment}

```

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

```

8013 \keys_define:nn { hwexam / assignment } {
8014 id .str_set:N = \l_@@_assign_id_str,
8015 number .int_set:N = \l_@@_assign_number_int,
8016 title .tl_set:N = \l_@@_assign_title_tl,
8017 type .tl_set:N = \l_@@_assign_type_tl,
8018 given .tl_set:N = \l_@@_assign_given_tl,
8019 due .tl_set:N = \l_@@_assign_due_tl,
8020 loadmodules .code:n = {
8021 \bool_set_true:N \l_@@_assign_loadmodules_bool
8022 }
8023 }
8024 \cs_new_protected:Nn \_@@_assignment_args:n {
8025 \str_clear:N \l_@@_assign_id_str
8026 \int_set:Nn \l_@@_assign_number_int {-1}
8027 \tl_clear:N \l_@@_assign_title_tl
8028 \tl_clear:N \l_@@_assign_type_tl
8029 \tl_clear:N \l_@@_assign_given_tl
8030 \tl_clear:N \l_@@_assign_due_tl
8031 \bool_set_false:N \l_@@_assign_loadmodules_bool
8032 \keys_set:nn { hwexam / assignment }{ #1 }
8033 }

```

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.

```

8034 \newcommand\given@due[2]{
8035 \bool_lazy_all:nF {
8036 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
8037 { \tl_if_empty_p:V \l_@@_assign_given_tl }
8038 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
8039 { \tl_if_empty_p:V \l_@@_assign_due_tl }
8040 }{ #1 }
8041
8042 \tl_if_empty:NTF \l_@@_inclasssign_given_tl {
8043 \tl_if_empty:NF \l_@@_assign_given_tl {
8044 \hwexam@given@kw\xspace\l_@@_assign_given_tl
8045 }
8046 }{
8047 \hwexam@given@kw\xspace\l_@@_inclasssign_given_tl
8048 }
8049
8050 \bool_lazy_or:nnF {
8051 \bool_lazy_and_p:nn {
8052 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
8053 }{
8054 \tl_if_empty_p:V \l_@@_assign_due_tl
8055 }
8056 }{
8057 \bool_lazy_and_p:nn {
8058 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
8059 }{
8060 \tl_if_empty_p:V \l_@@_assign_due_tl
8061 }
8062 }{ ,~ }
8063
8064 \tl_if_empty:NTF \l_@@_inclasssign_due_tl {
8065 \tl_if_empty:NF \l_@@_assign_due_tl {
8066 \hwexam@due@kw\xspace \l_@@_assign_due_tl
8067 }
8068 }{
8069 \hwexam@due@kw\xspace \l_@@_inclasssign_due_tl
8070 }
8071
8072 \bool_lazy_all:nF {
8073 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
8074 { \tl_if_empty_p:V \l_@@_assign_given_tl }
8075 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
8076 { \tl_if_empty_p:V \l_@@_assign_due_tl }
8077 }{ #2 }
8078 }

```

`\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.

```

8079 \newcommand\assignment@title[3]{
8080 \tl_if_empty:NTF \l_@@_inclasssign_title_tl {
8081 \tl_if_empty:NTF \l_@@_assign_title_tl {
8082 #1
8083 }{
8084 #2\l_@@_assign_title_tl#3
8085 }
8086 }{
8087 #2\l_@@_inclasssign_title_tl#3
8088 }
8089 }

```

(End definition for \assignment@title. This function is documented on page ??.)

**\assignment@number** Like \assignment@title only for the number, and no around part.

```

8090 \newcommand\assignment@number{
8091 \int_compare:nNnTF \l_@@_inclasssign_number_int = {-1} {
8092 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8093 \arabic{assignment}
8094 } {
8095 \int_use:N \l_@@_assign_number_int
8096 }
8097 }{
8098 \int_use:N \l_@@_inclasssign_number_int
8099 }
8100 }

```

(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.

```

8101 \newenvironment{assignment}[1][]{
8102 \_@@_assignment_args:n { #1 }
8103 %\sref@target
8104 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8105 \global\stepcounter{assignment}
8106 }{
8107 \global\setcounter{assignment}{\int_use:N\l_@@_assign_number_int}
8108 }
8109 \setcounter{sproblem}{0}
8110 \renewcommand\prob@label[1]{\assignment@number.##1}
8111 \def\current@section@level{\document@hwexamtype}
8112 %\sref@label{id}{\document@hwexamtype \thesection}
8113 \begin{@assignment}
8114 }{
8115 \end{@assignment}
8116 }

```

In the multi-assignment case we just use the omdoc environment for suitable sectioning.

```

8117 \def\ass@title{
8118 {\protect\document@hwexamtype}\arabic{assignment}
8119 \assignment@title{}\;\;{}{}\; -- \given@due{}\;}
8120 }
8121 \ifmultiple
8122 \newenvironment{@assignment}{
8123 \bool_if:NTF \l_@@_assign_loadmodules_bool {
8124 \begin{sfragment}[loadmodules]{\ass@title}
8125 }{
8126 \begin{sfragment}{\ass@title}
8127 }
8128 }{
8129 \end{sfragment}
8130 }

```

for the single-page case we make a title block from the same components.

```

8131 \else
8132 \newenvironment{@assignment}{
8133 \begin{center}\bf
8134 \Large@title\strut\
8135 \document@hwexamtype}\arabic{assignment}\assignment@title{}\;\;{}{}\;
8136 \large\given@due{--\;\;{}{}\;}\;
8137 \end{center}
8138 }{}
8139 \fi% multiple

```

### 39.3 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.

```

8140 \keys_define:nn { hwexam / inclassignment } {
8141 %id .str_set_x:N = \l_@@_assign_id_str,
8142 number .int_set:N = \l_@@_inclassign_number_int,
8143 title .tl_set:N = \l_@@_inclassign_title_tl,
8144 type .tl_set:N = \l_@@_inclassign_type_tl,
8145 given .tl_set:N = \l_@@_inclassign_given_tl,
8146 due .tl_set:N = \l_@@_inclassign_due_tl,
8147 mhrepos .str_set_x:N = \l_@@_inclassign_mhrepos_str
8148 }
8149 \cs_new_protected:Nn \l_@@_inclassignment_args:n {
8150 \int_set:Nn \l_@@_inclassign_number_int {-1}
8151 \tl_clear:N \l_@@_inclassign_title_tl
8152 \tl_clear:N \l_@@_inclassign_type_tl
8153 \tl_clear:N \l_@@_inclassign_given_tl
8154 \tl_clear:N \l_@@_inclassign_due_tl
8155 \str_clear:N \l_@@_inclassign_mhrepos_str
8156 \keys_set:nn { hwexam / inclassignment }{ #1 }
8157 }
8158 \l_@@_inclassignment_args:n {}
8159
8160 \newcommand\inputassignment[2][]{

```



```

8161 \_@@_inclassassignment_args:n { #1 }
8162 \str_if_empty:NTF \l_@@_inclasssign_mhrepos_str {
8163   \input{#2}
8164 }{
8165   \stex_in_repository:nn{\l_@@_inclasssign_mhrepos_str}{
8166     \input{\mhpath{\l_@@_inclasssign_mhrepos_str}{#2}}
8167   }
8168 }
8169 \_@@_inclassassignment_args:n {}
8170 }
8171 \newcommand\includeassignment[2][]{
8172   \newpage
8173   \inputassignment[#1]{#2}
8174 }

```

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

## 39.4 Typesetting Exams

\quizheading

```

8175 \ExplSyntaxOff
8176 \newcommand\quizheading[1]{%
8177   \def\@tas{#1}%
8178   \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
8179   \ifx\@tas\@empty\else%
8180     \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
8181   \fi%
8182 }
8183 \ExplSyntaxOn

```

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

\testheading

```

8184
8185 \def\hwexamheader{\input{hwexam-default.header}}
8186
8187 \def\hwexamminutes{
8188   \tl_if_empty:NTF \testheading@duration {
8189     {\testheading@min}~\hwexam@minutes@kw
8190   }{
8191     \testheading@duration
8192   }
8193 }
8194
8195 \keys_define:nn { hwexam / testheading } {
8196   min .tl_set:N = \testheading@min,
8197   duration .tl_set:N = \testheading@duration,
8198   reqpts .tl_set:N = \testheading@reqpts,
8199   tools .tl_set:N = \testheading@tools
8200 }
8201 \cs_new_protected:Nn \_@@_testheading_args:n {
8202   \tl_clear:N \testheading@min
8203   \tl_clear:N \testheading@duration

```

```

8204 \tl_clear:N \testheading@reqpts
8205 \tl_clear:N \testheading@tools
8206 \keys_set:nn { hwexam / testheading }{ #1 }
8207 }
8208 \newenvironment{testheading}[1][]{
8209 \_@@_testheading_args:n{ #1 }
8210 \newcount\check@time\check@time=\testheading@min
8211 \advance\check@time by -\theassignment@totalmin
8212 \newif\if@bonuspoints
8213 \tl_if_empty:NTF \testheading@reqpts {
8214 \@bonuspointsfalse
8215 }{
8216 \newcount\bonus@pts
8217 \bonus@pts=\theassignment@totalpts
8218 \advance\bonus@pts by -\testheading@reqpts
8219 \edef\bonus@pts{\the\bonus@pts}
8220 \@bonuspointstrue
8221 }
8222 \edef\check@time{\the\check@time}
8223
8224 \makeatletter\hwexamheader\makeatother
8225 }{
8226 \newpage
8227 }

```

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

\testspace

```

8228 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

8229 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

8230 \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.

```

8231 <@@=problems>
8232 \renewcommand\@problem[3]{
8233 \stepcounter{assignment@probs}
8234 \def\__problemspts{#2}
8235 \ifx\__problemspts\@empty\else
8236 \addtocounter{assignment@totalpts}{#2}
8237 \fi
8238 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
8239 \xdef\correction@probs{\correction@probs & #1}%
8240 \xdef\correction@pts{\correction@pts & #2}
8241 \xdef\correction@reached{\correction@reached &}

```

```

8242 }
8243 <@@=hwexam>

```

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

`\correction@table` This macro generates the correction table

```

8244 \newcounter{assignment@probs}
8245 \newcounter{assignment@totalpts}
8246 \newcounter{assignment@totalmin}
8247 \def\correction@probs{\correction@probs@kw}
8248 \def\correction@pts{\correction@pts@kw}
8249 \def\correction@reached{\correction@reached@kw}
8250 \stepcounter{assignment@probs}
8251 \newcommand\correction@table{
8252 \resizebox{\textwidth}{!}{%
8253 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
8254 &\multicolumn{\theassignment@probs}{c|}{}%
8255 {\footnotesize\correction@forgrading@kw} &\\ \hline
8256 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
8257 \correction@pts & \theassignment@totalpts & \\ \hline
8258 \correction@reached & & \[.7cm]\hline
8259 \end{tabular}}
8260 </package>

```

(End definition for \correction@table. This function is documented on page ??.)

## 39.5 Leftovers

at some point, we may want to reactivate the logos font, then we use

here we define the logos that characterize the assignment

```

\font\bierfont=../assignments/bierglas
\font\denkerfont=../assignments/denker
\font\uhrfont=../assignments/uhr
\font\warnschildfont=../assignments/achtung

\newcommand\bierglas{{\bierfont\char65}}
\newcommand\denker{{\denkerfont\char65}}
\newcommand\uhr{{\uhrfont\char65}}
\newcommand\warnschild{{\warnschildfont\char 65}}
\newcommand\hardA{\warnschild}
\newcommand\longA{\uhr}
\newcommand\thinkA{\denker}
\newcommand\discussA{\bierglas}

```

# Chapter 40

## References

- [Bus+04] Stephen Buswell et al. *The Open Math Standard, Version 2.0*. Tech. rep. The OpenMath Society, 2004. URL: <http://www.openmath.org/standard/om20>.
- [CR99] David Carlisle and Sebastian Rathz. *The graphicx package*. Part of the T<sub>E</sub>X distribution. The Comprehensive T<sub>E</sub>X Archive Network. 1999. URL: <https://www.tug.org/texlive/devsrc/Master/texmf-dist/doc/latex/graphics/graphics.pdf>.
- [DCM03] The DCMI Usage Board. *DCMI Metadata Terms*. DCMI Recommendation. Dublin Core Metadata Initiative, 2003. URL: <http://dublincore.org/documents/dcmi-terms/>.
- [Koh06] Michael Kohlhase. *OMDoc – An open markup format for mathematical documents [Version 1.2]*. LNAI 4180. Springer Verlag, Aug. 2006. URL: <http://omdoc.org/pubs/omdoc1.2.pdf>.
- [LMH] *LMH Scripts*. URL: <https://github.com/sLaTeX/lmhtools>.
- [MMT] *MMT – Language and System for the Uniform Representation of Knowledge*. Project web site. URL: <https://uniformal.github.io/> (visited on 01/15/2019).
- [MRK18] Dennis Müller, Florian Rabe, and Michael Kohlhase. “Theories as Types”. In: *9th International Joint Conference on Automated Reasoning*. Ed. by Didier Galmiche, Stephan Schulz, and Roberto Sebastiani. Springer Verlag, 2018. URL: <https://kwarc.info/kohlhase/papers/ijcar18-records.pdf>.
- [Rab15] Florian Rabe. “The Future of Logic: Foundation-Independence”. In: *Logica Universalis* 10.1 (2015). 10.1007/s11787-015-0132-x; Winner of the Contest “The Future of Logic” at the World Congress on Universal Logic, pp. 1–20.
- [RK13] Florian Rabe and Michael Kohlhase. “A Scalable Module System”. In: *Information & Computation* 0.230 (2013), pp. 1–54. URL: <https://kwarc.info/frabe/Research/mmt.pdf>.
- [RT] *sLaTeX/RusTeX*. URL: <https://github.com/sLaTeX/RusTeX> (visited on 04/22/2022).

---

<sup>13</sup>EdNOTE: we need an un-numbered version sfragment\*

- [SIa] *sLaTeX/sTeX-IDE*. URL: <https://github.com/slatex/sTeX-IDE> (visited on 04/22/2022).
- [SIb] *sLaTeX/stexls-vscode-plugin*. URL: <https://github.com/slatex/stexls-vscode-plugin> (visited on 04/22/2022).
- [SLS] *sLaTeX/stexls*. URL: <https://github.com/slatex/stexls> (visited on 04/22/2022).
- [ST] *sTeX – An Infrastructure for Semantic Preloading of LaTeX Documents*. URL: <https://ctan.org/pkg/stex> (visited on 04/22/2022).
- [sTeX] *sTeX: A semantic Extension of TeX/LaTeX*. URL: <https://github.com/sLaTeX/sTeX> (visited on 05/11/2020).
- [Tana] Till Tantau. *beamer – A LaTeX class for producing presentations and slides*. URL: <http://ctan.org/pkg/beamer> (visited on 01/07/2014).
- [Tanb] Till Tantau. *User Guide to the Beamer Class*. URL: <http://ctan.org/macros/latex/contrib/beamer/doc/beameruserguide.pdf>.
- [TL] *TeX Live*. URL: <http://www.tug.org/texlive/> (visited on 12/11/2012).