

# The sTeX3 Package Collection \*

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

2022-08-11

## 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 3.2 (last revised 2022-08-11)

# 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
3.3.3	Operator Notations	20
3.3.3	Argument Modes	20
3.3.3	Mode-b Arguments	20
3.3.3	Mode-a Arguments	21
3.3.3	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	29
3.4.1	Multilinguality and Translations	29
3.4.2	Simple Inheritance and Namespaces	30
3.4.3	The mathstructure Environment	32
3.4.4	The copymodule Environment	35
3.4.5	The interpretmodule Environment	36
3.5	Primitive Symbols (The sTeX Metatheory)	37
<b>4</b>	<b>Using sTeX Symbols</b>	<b>38</b>
4.1	\symref and its variants	38
4.2	Marking Up Text and On-the-Fly Notations	39

<b>5</b>	<b>sTeX Statements</b>	<b>43</b>
5.1	Definitions, Theorems, Examples, Paragraphs . . . . .	43
5.2	Proofs . . . . .	46
5.3	Highlighting and Presentation Customizations . . . . .	51
<b>6</b>	<b>Cross References</b>	<b>53</b>
<b>7</b>	<b>Additional Packages</b>	<b>55</b>
7.1	Tikzinput: Treating TIKZ code as images . . . . .	55
7.2	Modular Document Structuring . . . . .	56
7.2.1	Introduction . . . . .	56
7.2.2	Package Options . . . . .	56
7.2.3	Document Fragments . . . . .	56
7.2.4	Ending Documents Prematurely . . . . .	58
7.2.5	Global Document Variables . . . . .	58
7.3	Slides and Course Notes . . . . .	58
7.3.1	Introduction . . . . .	58
7.3.2	Package Options . . . . .	59
7.3.3	Notes and Slides . . . . .	59
7.3.4	Customizing Header and Footer Lines . . . . .	60
7.3.5	Frame Images . . . . .	61
7.3.6	Excursions . . . . .	62
7.4	Representing Problems and Solutions . . . . .	63
7.4.1	Introduction . . . . .	63
7.4.2	Problems and Solutions . . . . .	63
7.4.3	Markup for Added-Value Services . . . . .	65
	Multiple Choice Blocks . . . . .	65
	Filling-In Concrete Solutions . . . . .	66
7.4.4	Including Problems . . . . .	67
7.5	Homeworks, Quizzes and Exams . . . . .	68
7.5.1	Introduction . . . . .	68
7.5.2	Package Options . . . . .	68
7.5.3	Assignments . . . . .	68
7.5.4	Including Assignments . . . . .	68
7.5.5	Typesetting Exams . . . . .	69
<b>II</b>	<b>Documentation</b>	<b>70</b>
<b>8</b>	<b>sTeX-Basics</b>	<b>71</b>
8.1	Macros and Environments . . . . .	71
8.1.1	HTML Annotations . . . . .	71
8.1.2	Babel Languages . . . . .	72
8.1.3	Auxiliary Methods . . . . .	72
<b>9</b>	<b>sTeX-MathHub</b>	<b>73</b>
9.1	Macros and Environments . . . . .	73
9.1.1	Files, Paths, URIs . . . . .	73
9.1.2	MathHub Archives . . . . .	74
9.1.3	Using Content in Archives . . . . .	75

<b>10</b>	<b>sTeX-References</b>	<b>76</b>
10.1	Macros and Environments . . . . .	76
10.1.1	Setting Reference Targets . . . . .	76
10.1.2	Using References . . . . .	77
<b>11</b>	<b>sTeX-Modules</b>	<b>78</b>
11.1	Macros and Environments . . . . .	78
11.1.1	The <code>smodule</code> environment . . . . .	80
<b>12</b>	<b>sTeX-Module Inheritance</b>	<b>82</b>
12.1	Macros and Environments . . . . .	82
12.1.1	SMS Mode . . . . .	82
12.1.2	Imports and Inheritance . . . . .	83
<b>13</b>	<b>sTeX-Symbols</b>	<b>85</b>
13.1	Macros and Environments . . . . .	85
<b>14</b>	<b>sTeX-Terms</b>	<b>87</b>
14.1	Macros and Environments . . . . .	87
<b>15</b>	<b>sTeX-Structural Features</b>	<b>89</b>
15.1	Macros and Environments . . . . .	89
15.1.1	Structures . . . . .	89
<b>16</b>	<b>sTeX-Statements</b>	<b>90</b>
16.1	Macros and Environments . . . . .	90
<b>17</b>	<b>sTeX-Proofs: Structural Markup for Proofs</b>	<b>91</b>
<b>18</b>	<b>sTeX-Metatheory</b>	<b>92</b>
18.1	Symbols . . . . .	92
<b>III</b>	<b>Extensions</b>	<b>93</b>
<b>19</b>	<b>Tikzinput: Treating TIKZ code as images</b>	<b>94</b>
19.1	Macros and Environments . . . . .	94
<b>20</b>	<b>document-structure: Semantic Markup for Open Mathematical Documents in L<sup>A</sup>T<sub>E</sub>X</b>	<b>95</b>
<b>21</b>	<b>NotesSlides – Slides and Course Notes</b>	<b>96</b>
<b>22</b>	<b>problem.sty: An Infrastructure for formatting Problems</b>	<b>97</b>
<b>23</b>	<b>hwexam.sty/cls: An Infrastructure for formatting Assignments and Exams</b>	<b>98</b>
<b>IV</b>	<b>Implementation</b>	<b>99</b>

<b>24</b>	<b>STeX-Basics Implementation</b>	<b>100</b>
24.1	The STeXDocument Class . . . . .	100
24.2	Preliminaries . . . . .	101
24.3	Messages and logging . . . . .	101
24.4	HTML Annotations . . . . .	102
24.5	Babel Languages . . . . .	104
24.6	Persistence . . . . .	106
24.7	Auxiliary Methods . . . . .	106
<b>25</b>	<b>STeX-MathHub Implementation</b>	<b>110</b>
25.1	Generic Path Handling . . . . .	110
25.2	PWD and kpsewhich . . . . .	112
25.3	File Hooks and Tracking . . . . .	113
25.4	MathHub Repositories . . . . .	114
25.5	Using Content in Archives . . . . .	119
<b>26</b>	<b>STeX-References Implementation</b>	<b>124</b>
26.1	Document URIs and URLs . . . . .	124
26.2	Setting Reference Targets . . . . .	126
26.3	Using References . . . . .	129
<b>27</b>	<b>STeX-Modules Implementation</b>	<b>135</b>
27.1	The smodule environment . . . . .	139
27.2	Invoking modules . . . . .	145
<b>28</b>	<b>STeX-Module Inheritance Implementation</b>	<b>147</b>
28.1	SMS Mode . . . . .	147
28.2	Inheritance . . . . .	151
<b>29</b>	<b>STeX-Symbols Implementation</b>	<b>157</b>
29.1	Symbol Declarations . . . . .	157
29.2	Notations . . . . .	165
29.3	Variables . . . . .	174
<b>30</b>	<b>STeX-Terms Implementation</b>	<b>182</b>
30.1	Symbol Invocations . . . . .	182
30.2	Terms . . . . .	190
30.3	Notation Components . . . . .	195
30.4	Variables . . . . .	197
30.5	Sequences . . . . .	200
<b>31</b>	<b>STeX-Structural Features Implementation</b>	<b>201</b>
31.1	Imports with modification . . . . .	202
31.2	The feature environment . . . . .	210
31.3	Structure . . . . .	210
<b>32</b>	<b>STeX-Statements Implementation</b>	<b>221</b>
32.1	Definitions . . . . .	221
32.2	Assertions . . . . .	227
32.3	Examples . . . . .	230
32.4	Logical Paragraphs . . . . .	233

<b>33 The Implementation</b>	<b>238</b>
33.1 Proofs . . . . .	238
<b>34 <math>\text{\TeX}</math>-Others Implementation</b>	<b>247</b>
<b>35 <math>\text{\TeX}</math>-Metatheory Implementation</b>	<b>249</b>
<b>36 Tikzinput Implementation</b>	<b>252</b>
<b>37 document-structure.sty Implementation</b>	<b>255</b>
37.1 Package Options . . . . .	255
37.2 Document Structure . . . . .	256
37.3 Front and Backmatter . . . . .	260
37.4 Global Variables . . . . .	262
<b>38 NotesSlides – Implementation</b>	<b>263</b>
38.1 Class and Package Options . . . . .	263
38.2 Notes and Slides . . . . .	265
38.3 Header and Footer Lines . . . . .	269
38.4 Frame Images . . . . .	271
38.5 Sectioning . . . . .	272
38.6 Excursions . . . . .	275
<b>39 The Implementation</b>	<b>277</b>
39.1 Package Options . . . . .	277
39.2 Problems and Solutions . . . . .	278
39.3 Markup for Added Value Services . . . . .	285
39.4 Multiple Choice Blocks . . . . .	285
39.5 Filling in Concrete Solutions . . . . .	286
39.6 Including Problems . . . . .	286
39.7 Reporting Metadata . . . . .	288
<b>40 Implementation: The hwexam Package</b>	<b>290</b>
40.1 Package Options . . . . .	290
40.2 Assignments . . . . .	291
40.3 Including Assignments . . . . .	294
40.4 Typesetting Exams . . . . .	295
40.5 Leftovers . . . . .	297
<b>41 References</b>	<b>298</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  $\text{\texttt{L}^A\text{\texttt{T}}_E\text{\texttt{X}}$  more modularly (object-oriented Math) for creating PDF documents or
2. foundation for authoring active documents in HTML5 instrumented with knowledge management services.

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

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

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

Alternatively, you can install  $\text{\texttt{sTeX}}$  from CTAN, the Comprehensive  $\text{\texttt{T}}_E\text{\texttt{X}}$  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 sTeX Archives (Manual Setup)

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

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

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

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

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

### 2.1.4 The sTeX IDE

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

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

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

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

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

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

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

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

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

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

```

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

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

```
smodule (env.) \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`     The `\define{geometricSeries}` ...  
`\definiendum`     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  $a/b$ .

---

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

---



---

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

---

### 2.2.1 OMDoc/xhtml Conversion

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

#### TODO VSCode Plugin

Using `RuSTeX` [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>
```

...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 ??) 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 `\symdecl{symbolname}`, `\symdef{symbolname}` and some other constructions. Most symbols have a *notation* that can be used via a *semantic macro* `\symbolname` generated by symbol declarations.
4.  $\text{\S}\text{\TeX}$  **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*. `\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].

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

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

### 3.2.1 The Local MathHub-Directory

`\usemodule`, `\importmodule`, `\inputref` etc. allow for including content modularly without having to specify absolute paths, which would differ between users and machines. Instead,  $\text{\S}\text{\TeX}$  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:

---

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

---

`\inputref` `\inputref`[Some/Archive]{some/file} behaves like `\mhinput`, but wraps the input in a `\begingroup ... \endgroup`. When converting to `xhtml`, 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 `\inputref` is likely to be preferred over `\mhinput` because it leads to less duplication in the generated `xhtml`.

---

`\ifinput` Both `\mhinput` and `\inputref` set `\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.

---

`\addmhbibresource` `\addmhbibresource`[Some/Archive]{some/file} searches for a file like `\mhinput` does, but calls `\addbibresource` to the result and looks for the file in the archive root directory directly, rather than the `source` directory. Typical invocations are

- `\addmhbibresource{lib/refs.bib}`, which specifies a bibliography in the `lib` folder in the local archive or
- `\addmhbibresource[HW/meta-inf]{lib/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` (*env.*) 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 sTeX module corresponds to an MMT/OMDOC *theory*. As such it  
 $\hookrightarrow$  gets assigned a module URI (*universal resource identifier*) of the form  
 $\hookrightarrow$  `<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 (\smodulename)}}\par}
3   {\par\noindent\textbf{End of Module (\smodulename)}}}
4
5 \begin{smodule}[type=display,title={Some New Module}]{SomeModule2}
6   Hello World
7 \end{smodule}

```

Output:

```

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

```

### 3.3.2 Declaring New Symbols and Notations

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

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

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

$\hookrightarrow$  `\symdecl` introduces a new OMDoc/MMT constant in the current module (=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  $\text{\TeX}$  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  $\binarysymbol[\text{highlight}]$ :

### 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  $\text{\texttt{STEX}}$  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  $\text{\texttt{TEX}}$ ’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"/>` directly.

### 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  $\text{T}_{\text{E}}\text{X}$ , but applications of binding operators, i.e. symbols with *mode-b* arguments, are translated to *OMBIND*-terms in OMDoc/MMT, rather than *OMA*.

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

### Example 9

Input:

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

Output:

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

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

### Mode-a Arguments

Mode-a arguments represent a *flexary argument sequence*, i.e. a sequence of arguments of arbitrary length. Formally, operators that take arbitrarily many arguments don't "exist", but in informal mathematics, they are ubiquitous. Mode-a arguments allow us to write e.g. `\addition{a,b,c,d,e}` rather than having to write something like `\addition{a}\addition{b}\addition{c}\addition{d}\addition{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=iaai]
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  $\text{\texttt{STeX}}$  (or, rather,  $\text{\texttt{MMT/OMDoc}}$ ) 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  $\text{\texttt{OMDoc/MMT}}$  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  $\TeX$  insert parentheses automatically.

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

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

### Example 18

Input:

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

Output:

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

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

---

`\infprec`  
`\neginfprec`

It is occasionally useful to have “infinitely” high or low precedences to enforce or forbid automated bracketing entirely, 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).

More precisely, each notation takes

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

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

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

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

In the example above:



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

### 3.3.6 Variables

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

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

---

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



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

---

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

### Example 19

Input:

```

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

```

Output:

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

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

TODO: bind=forall/exists

### 3.3.7 Variable Sequences

Variable *sequences* occur quite frequently in informal mathematics, hence they deserve special support. Variable sequences behave like variables in that they disappear at the end of the current T<sub>E</sub>X group and are not exported from modules, but their declaration is quite different.

---

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

This is best shown by example:

### Example 20

Input:

```

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

```

Output:

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

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

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

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

### Example 21

Input:

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

Output:

$a_1 + \dots + a_n$

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

### Example 22

Input:

```

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

```

Output:

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

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

### Example 23

Input:

```

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

```

Output:

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

## 3.4 Module Inheritance and Structures

The  $\text{\TeX}$  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  $\text{\TeX}$ ) and the edges are truth-preserving mappings (called theory morphisms in MMT). We have already seen modules/theories above.

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

### 3.4.1 Multilinguality and Translations

If we load the  $\text{\TeX}$  document class or package with the option `lang=<lang>`,  $\text{\TeX}$  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: `\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`. 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    $\text{lcm}\{a,b\}$  von zwei Zahlen  $a,b$  ist...
6 \end{smodule}
```

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

### 3.4.2 Simple Inheritance and Namespaces

---

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

---

`\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` (*env.*) 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  $\hookrightarrow M \rightarrow$  `<name>-structure`. The *constant* `<name>` is defined as `Mod(<name>-structure)`
- $\hookrightarrow M \rightarrow$  – a *dependent record type with manifest fields*, the fields of which are generated
- $\rightsquigarrow T \rightsquigarrow$  from (and correspond to) the constants in `<name>-structure`.

`\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  $\varMb := \mathsf{mathstruct}\{\varMb{universe}, \varMb{op}!, \varMb{unit}\}$ 
4 be a  $\mathsf{symname}\{\mathsf{monoid}\}$  on  $\mathbb{Z}$  ...

```

Output:

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

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

### 3.4.4 The copymodule Environment

TODO: explain

Given modules:

### Example 29

Input:

```

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

```

Output:

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

### Example 30

Input:

```

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

```

Output:

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

TODO: explain donotclone

### 3.4.5 The interpretmodule Environment

TODO: explain

#### Example 31

Input:

```

1 \begin{smodule}{int}
2   \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 $\mathsf{STEX}$ Metatheory)

The `stex-metatheory` package contains  $\mathsf{STEX}$  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  $\mathsf{STEX}$  module.

We can also see the `stex-metatheory` as a foundation of mathematics in the sense of [Rab15], albeit an informal one (the ones discussed there are all formal foundations). The state of the `stex-metatheory` is necessarily incomplete, and will stay so for a long while: It arises as a collection of empirically useful symbols that are collected as more and more mathematics are encoded in  $\mathsf{STEX}$  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  $\mathsf{STEX}$  collection due to the obiquity of the symbols involved. Note however, that the metatheory is for all practical purposes a “normal”  $\mathsf{STEX}$  module, and the symbols contained “normal”  $\mathsf{STEX}$  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

---

<code>\symref</code> <code>\symname</code>	We have already seen <code>\symname</code> and <code>\symref</code> , the latter being the more general. <code>\symref{&lt;symbolname&gt;}{&lt;code&gt;}</code> marks-up <code>&lt;code&gt;</code> as referencing <code>&lt;symbolname&gt;</code> . Since quite often, the <code>&lt;code&gt;</code> should be (a variant of) the name of the symbol anyway, we also have <code>\symname{&lt;symbolname&gt;}</code> .
---	--

---

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  $m$ .

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

<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.

Output:

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...

If we take features like `\inputref` and `\mhinput` (and the `sfragment`-environment, see [subsection 7.2.1](#)) seriously, and build large documents modularly from individually compiling documents for sections, chapters and so on, cross-referencing becomes an interesting problem.

Say, we have a document `main.tex`, which `\inputrefs` a section `section1.tex`, which references a definition with label `some_definition` in `section2.tex` (subsequently also inputted in `main.tex`). Then the numbering of the definition will depend on the *document context* in which the document fragment `section2.tex` occurs - in `section2.tex` itself (as a standalone document), it might be *Definition 1*, in `main.tex` it might be *Definition 3.1*, and in `section1.tex`, the definition *does not even occur*, so it needs to be referenced by some other text.

What we would want in that instance is an equivalent of `\autoref`, that takes the document context into account to yield something like *Definition 1*, *Definition 3.1* or *Definition 1 in the section on Foo* respectively.

The `\sref` command attempts to do precisely that. Unlike plain `\ref`, `\autoref` etc., `\sref` refers to not just a *label*, but instead a pair consisting of a *label* and the *document* in whose context we want to refer to it. Conversely, every *document* (i.e. standalone compilable `.tex`-file) keeps track of the “names” (*Definition 3.1* etc.) for every label as determined in the context of the document, and stores them in a dedicated file `\jobname.sref`. Additionally, every document has a “reference name” (e.g. “*the section on Foo*”). This allows us to refer to “label  $x$  in document  $D$ ” to yield “*Definition 1 in the section on Foo*”. And of course,  $\TeX$  can decide based on the current document

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

to either refer to the label by its “full name” or directly as e.g. *Definition 3.1* depending on whether the label occurs in the current document anyway (and link to it accordingly).

For that to work, we need to supply (up to) three pieces of information:

- The *label* of the reference target (e.g. `some_definition`),
- (optionally) the *file*/document containing the reference target (e.g. `section2`). This is not strictly necessary, but allows for additional disambiguation between possibly duplicate labels across files, and
- (optionally) the document context, in which we want to refer to the reference target (e.g. `main`).

Additionally, the document in which we want to reference a label needs a title for external references.

---

```
\sref[archive=<archive1>,file=<file>]
{\<label>}[archive=<archive2>,in=<document-context>,title=<title>]
```

---

This command references *<label>* (declared in *<file>* in *<archive1>*). If the object (section, figure, etc.) with that label occurs ultimately in the same document, `\sref` will ignore the second set of optional arguments and simply defer to `\autoref` if that command exists, or `\ref` if the `hyperref` package is not included.

If the referenced object does *not* occur in the current document however, `\sref` will refer to it by the object’s name as it occurs in the file *<document-context>* in *<archive2>*.

For example, the reference to the `sfragment`-environment above will appear as “subsection 7.2.1 (Introduction) in the  $\text{\TeX}$ 3 manual” if you are reading this in the package documentation for `stex-references` directly, but as a linked “subsection 7.2.1” in the full documentation or manual. This is achieved using

```
\sref[file=stex-document-structure]{sec:ds:intro}[in=../stex-manual,title={the \sTeX}3 document]
```

For a further example, the following:

### Part III

will say “Part III” (and link accordingly) in the full documentation, and “Part III (Extensions) in the full  $\text{\TeX}$ 3 documentation” everywhere else. This is achieved using

```
\sref[file=../stex-doc]{part:extends}[in=../stex-doc,title={the full \sTeX}3 document]
```

---

```
\extref\sref[archive=<archive1>,file=<file>]
{\<label>}[archive=<archive2>,in=<document-context>,title=<title>]}
```

---

The `\extref`-command behaves exactly like `\sref`, but takes *required* the document context argument and will always use it for generating the document text, regardless of whether the label occurs in the current document.



## Chapter 5

# ST<sub>E</sub>X 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 ?? 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 ??), `type=` for customization (see ??) 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` **sdefinition** (and **sparagraph** with `type=symdoc`) introduce three new macros: `\definiendum` behaves like `\symref` (and `\definame`/`\Definame` like `\symname`/`\Symname`, respectively), but highlights the referenced symbol as *being defined* in the current definition.

---

$\hookrightarrow$  M  $\rightarrow$  The special `type=symdoc` for **sparagraph** is intended to be used for “informal definitions”, or encyclopedia-style descriptions for symbols.  
 $\hookrightarrow$  M  $\rightarrow$  The MMT system can use those (in lieu of an actual **sdefinition** in scope) to  
 $\hookrightarrow$  T  $\rightarrow$  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   \syndef{universe}[type=\set]{\comp{U}}
3   \syndef{op}[
4     args=2,
5     type=\funtype{\universe,\universe}{\universe},
6     op=\circ
7   ]{#1 \comp{\circ} #2}
8   \syndef{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...

EdN:4

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{spproof}

```

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-*

ulting 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  $\sum_{i=1}^n 2i-1=n^2$  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. □

**sproof** (*env.*) The **sproof** 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.

---

**\spfilea** The **\spfilea** 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 **sproof** 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** (*env.*) 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  $\text{\LaTeX}$  templates come with their own `definition`, `theorem` and similar environments that authors are supposed (or even required) to use. Their concrete syntax however is usually not compatible with all the additional arguments that  $\text{\LaTeX}$  allows for semantic information.

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

---

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

All of these commands take one optional and two proper arguments, i.e. `\stexpatch* [<type>] {<begin-code>} {<end-code>}`.

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

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

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

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

```

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

```

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

```

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

```

<hr/>	
<code>\compemph</code>	Apart from the environments, we can control how $\text{\TeX}$ highlights variables, notation
<code>\varemp</code>	components, <code>\symrefs</code> and <code>\definiendums</code> , respectively.
<code>\symrefemph</code>	To do so, we simply redefine these four macros. For example, to highlight nota-
<code>\defemph</code>	tion components (i.e. everything in a <code>\comp</code> ) in blue, as in this document, we can do
	<code>\def\compemph#1{\textcolor{blue}{#1}}</code> . By default, <code>\compemph</code> et al do nothing.

<hr/>	
<code>\compemph@uri</code>	For each of the four macros, there exists an additional macro that takes the full URI of
<code>\varemp@uri</code>	the relevant symbol currently being highlighted as a second argument. That allows us to
<code>\symrefemph@uri</code>	e.g. use pdf tooltips and links. For example, this document uses <sup>5</sup>
<code>\defemph@uri</code>	
	<pre> 1 \protected\def\symrefemph@uri#1#2{ 2   \pdftooltip{ 3     \symrefemph{#1} 4   }{ 5     URI:~\detokenize{#2} 6   } 7 } </pre>

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

## Chapter 6

# Cross References

If we take features like `\inputref` and `\mhinput` (and the `sfragment`-environment, see [subsection 7.2.1](#)) seriously, and build large documents modularly from individually compiling documents for sections, chapters and so on, cross-referencing becomes an interesting problem.

Say, we have a document `main.tex`, which `\inputrefs` a section `section1.tex`, which references a definition with label `some_definition` in `section2.tex` (subsequently also inputted in `main.tex`). Then the numbering of the definition will depend on the *document context* in which the document fragment `section2.tex` occurs - in `section2.tex` itself (as a standalone document), it might be *Definition 1*, in `main.tex` it might be *Definition 3.1*, and in `section1.tex`, the definition *does not even occur*, so it needs to be referenced by some other text.

What we would want in that instance is an equivalent of `\autoref`, that takes the document context into account to yield something like *Definition 1*, *Definition 3.1* or *Definition 1 in the section on Foo* respectively.

The `\sref` command attempts to do precisely that. Unlike plain `\ref`, `\autoref` etc., `\sref` refers to not just a *label*, but instead a pair consisting of a *label* and the *document* in whose context we want to refer to it. Conversely, every *document* (i.e. standalone compilable `.tex`-file) keeps track of the “names” (*Definition 3.1* etc.) for every label as determined in the context of the document, and stores them in a dedicated file `\jobname.sref`. Additionally, every document has a “reference name” (e.g. “*the section on Foo*”). This allows us to refer to “label *x* in document *D*” to yield “*Definition 1 in the section on Foo*”. And of course, `TEX` can decide based on the current document to either refer to the label by its “full name” or directly as e.g. *Definition 3.1* depending on whether the label occurs in the current document anyway (and link to it accordingly).

For that to work, we need to supply (up to) three pieces of information:

- The *label* of the reference target (e.g. `some_definition`),
- (optionally) the *file*/document containing the reference target (e.g. `section2`). This is not strictly necessary, but allows for additional disambiguation between possibly duplicate labels across files, and
- (optionally) the document context, in which we want to refer to the reference target (e.g. `main`).

Additionally, the document in which we want to reference a label needs a title for external references.

---

**\sref** `\sref[archive=<archive1>,file=<file>]  
{<label>}[archive=<archive2>,in=<document-context>,title=<title>]`

---

This command references *<label>* (declared in *<file>* in *<archive1>*). If the object (section, figure, etc.) with that label occurs ultimately in the same document, `\sref` will ignore the second set of optional arguments and simply defer to `\autoref` if that command exists, or `\ref` if the `hyperref` package is not included.

If the referenced object does *not* occur in the current document however, `\sref` will refer to it by the object's name as it occurs in the file *<document-context>* in *<archive2>*.

For example, the reference to the `sfragment`-environment above will appear as “subsection 7.2.1 (Introduction) in the `gTeX3` manual” if you are reading this in the package documentation for `stex-references` directly, but as a linked “subsection 7.2.1” in the full documentation or manual. This is achieved using

```
\sref[file=stex-document-structure]{sec:ds:intro}[in=./stex-manual,title={the \sTeX}]
```

For a further example, the following:

### Part III

will say “Part III” (and link accordingly) in the full documentation, and “Part III (Extensions) in the full `gTeX3` documentation” everywhere else. This is achieved using

```
\sref[file=./stex-doc]{part:extends}[in=./stex-doc,title={the full \sTeX}3 document]
```

---

**\extref** `\sref[archive=<archive1>,file=<file>]  
{<label>}[archive=<archive2>,in=<document-context>,title=<title>]}`

---

The `\extref`-command behaves exactly like `\sref`, but takes *required* the document context argument and will always use it for generating the document text, regardless of whether the label occurs in the current document.

## Chapter 7

# Additional Packages

### 7.1 Tikzinput: Treating TIKZ code as images

---

**image** The behavior of the `tikzinput` 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** 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.

---

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

---



---

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

---

## 7.2 Modular Document Structuring

### 7.2.1 Introduction

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

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

### 7.2.2 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>

### 7.2.3 Document Fragments

`sfragment` (*env.*) The structure of the document is given by nested `sfragment` environments. In the  $\text{\LaTeX}$  route, the `sfragment` environment is flexibly mapped to sectioning commands, inducing the proper sectioning level from the nesting of `sfragment` environments. Correspondingly, the `sfragment` environment takes an optional key/value argument for metadata followed by a regular argument for the (section) title of the sfragment. The optional metadata argument has the keys `id` for an identifier, `creators` and `contributors` for the Dublin Core metadata [DCM03]. The option `short` allows to give a short title for the generated section. If the title contains semantic macros, 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` (*env.*) 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{kwar}
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.

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

## 7.2.4 Ending Documents Prematurely

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

`\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.

## 7.2.5 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.

<hr/> <code>\setSGvar</code> <hr/>	<code>\setSGvar{&lt;vname&gt;}{&lt;text&gt;}</code> to set the global variable <code>&lt;vname&gt;</code> to <code>&lt;text&gt;</code> and <code>\useSGvar{&lt;vname&gt;}</code> to reference it.
<code>\useSGvar</code>	

<hr/> <code>\ifSGvar</code> <hr/>	With <code>\ifSGvar</code> we can test for the contents of a global variable: the macro call <code>\ifSGvar{&lt;vname&gt;}{&lt;val&gt;}{&lt;ctext&gt;}</code> tests the content of the global variable <code>&lt;vname&gt;</code> , only if (after expansion) it is equal to <code>&lt;val&gt;</code> , the conditional text <code>&lt;ctext&gt;</code> is formatted.
-----------------------------------	---

## 7.3 Slides and Course Notes

### 7.3.1 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.

### 7.3.2 Package Options

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

<code>slides</code>	The options <code>slides</code> and <code>notes</code> switch between slides mode and notes mode (see <a href="#">subsection 7.3.3</a> ).
<code>notes</code>	
<code>sectocframes</code>	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.
<code>frameimages</code>	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.
<code>fiboxed</code>	

### 7.3.3 Notes and Slides

`frame` (*env.*) Slides are represented with the `frame` environment just like in the `beamer` class, see [\[Tanb\]](#) for details.

`note` (*env.*) 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 L<sup>A</sup>T<sub>E</sub>X becomes confused and throws error messages that are difficult to decipher.

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

```

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

```

```

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

```

---

**\ifnotes** Note the use of the `\ifnotes` conditional, which allows different treatment between `notes` and `slides` mode – manually setting `\notesttrue` or `\notestfalse` 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 `notestslides` 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}`.

`nparagraph` (*env.*) There are some environments that tend to occur at the top-level of `note` environments.

`nparagraph` (*env.*) We make convenience versions of these: e.g. the `nparagraph` environment is just an

`ndefinition` (*env.*) `sparagraph` inside a `note` environment (but looks nicer in the source, since it avoids one

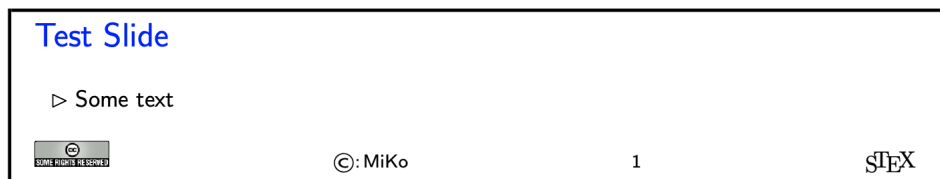
`nexample` (*env.*) level of source indenting). Similarly, we have the `nfragment`, `ndefinition`, `nexample`,

`nsproof` (*env.*) `nsproof`, and `nassertion` environments.

`nassertion` (*env.*)

### 7.3.4 Customizing Header and Footer Lines

The `notestslides` package and class comes with a simple default theme named `sTeX` that provided by the `beamerthemesTeX`. 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.

---

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

---



---

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

---



---

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

---

### 7.3.5 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.

---

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

---

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

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

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


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

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

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

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

---

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

---

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

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

---

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

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



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

## 7.4 Representing Problems and Solutions

### 7.4.1 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.

### 7.4.2 Problems and Solutions

---

<code>solutions</code>	The <code>problem</code> package takes the options <code>solutions</code> (should solutions be output?), <code>notes</code>
<code>notes</code>	(should the problem notes be presented?), <code>hints</code> (do we give the hints?), <code>gnotes</code> (do we
<code>hints</code>	show grading notes?), <code>pts</code> (do we display the points awarded for solving the problem?),
<code>gnotes</code>	<code>min</code> (do we display the estimated minutes for problem soling). If theses are specified, then
<code>pts</code>	the corresponding auxiliary parts of the problems are output, otherwise, they remain
<code>min</code>	invisible.
<code>boxed</code>	The <code>boxed</code> option specifies that problems should be formatted in framed boxes so
<code>test</code>	that they are more visible in the text. Finally, the <code>test</code> option signifies that we are in

---

a test situation, so this option does not show the solutions (of course), but leaves space for the students to solve them.

`problem (env.)` The main environment provided by the `problempackage` is (surprise surprise) the `problem` environment. It is used to mark up problems and exercises. The environment takes an optional `KeyVal` argument with the keys `id` as an identifier that can be reference later, `pts` for the points to be gained from this exercise in homework or quiz situations, `min` for the estimated minutes needed to solve the problem, and finally `title` for an informative title of the problem.

#### 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 7.4.1 (Fitting Elephants)**  
 How many Elephants can you fit into a Volkswagen beetle?  


---

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


---

**Note:** Justify your answer  


---

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


---

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


---



---

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

---

`\startsolutions` Sometimes we would like to locally override the `solutions` option we have given to  
`\stopsolutions` 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.

### 7.4.3 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` (*env.*) 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 7.4.2 (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=Nooooooooo,feedback=that is for Java]{public static void}
9   \end{mcb}
10 \end{sproblem}

```

Output:

**Problem 7.4.3 (Functions)**

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

☐ def

☐ 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 7.4.4 (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 7.4.5 (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>

## 7.4.4 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 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.

<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.

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

## 7.5 Homeworks, Quizzes and Exams

### 7.5.1 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.

### 7.5.2 Package Options

<hr/> <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>	
<hr/> <code>multiple</code>	Furthermore, the <code>hwexam</code> package takes the option <code>multiple</code> that allows to combine multiple assignment sheets into a compound document (the assignment sheets are treated as section, there is a table of contents, etc.).
<code>test</code>	Finally, there is the option <code>test</code> that modifies the behavior to facilitate formatting tests. Only in <code>test</code> mode, the macros <code>\testspace</code> , <code>\testnewpage</code> , and <code>\testemptypage</code> have an effect: they generate space for the students to solve the given problems. Thus they can be left in the $\text{\LaTeX}$ source.

### 7.5.3 Assignments

<code>assignment</code> ( <i>env.</i> )	This package supplies the <code>assignment</code> environment that groups problems into assignment sheets. It takes an optional KeyVal 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>	

### 7.5.4 Including Assignments

<hr/> <code>\inputassignment</code> <hr/>	The <code>\inputassignment</code> 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 <code>assignment</code> environment in the included file). The keys <code>number</code> , <code>title</code> , <code>type</code> , <code>given</code> , and <code>due</code> are just as for the <code>assignment</code> environment and (if given) overwrite the ones specified in the <code>assignment</code> environment in the included file.
---	---

### 7.5.5 Typesetting Exams

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

`testheading (env.)` 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.

```
reqpts1 \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)

2022-08-11

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

Write the solutions to the sheet.

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

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

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

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

To be used for grading, do not write here														
prob.	7.4.1	7.4.2	7.4.3	7.4.4	7.4.5	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

<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 8

## STEX-Basics

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

### 8.1 Macros and Environments

---

<code>\sTeX</code>	Both print this ST <sub>E</sub> X logo.
<code>\stex</code>	

---

---

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

---

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

#### 8.1.1 HTML Annotations

---

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

---

---

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

---

---

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

---

---

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

---

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

---

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

---

Annotates the HTML generated by `⟨content⟩` with

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

### 8.1.2 Babel Languages

---

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

---

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

### 8.1.3 Auxiliary Methods

---

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

---

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

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

---

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

---

## Chapter 9

# STEX-MathHub

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

### 9.1 Macros and Environments

---

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

---

#### 9.1.1 Files, Paths, URIs

---

<code>\stex_path_from_string:Nn</code>	<code>\stex_path_from_string:Nn</code> $\langle path-variable \rangle$ $\{ \langle string \rangle \}$ 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>	The inverse; turns a path into a string and stores it in the second argument variable, or
<code>\stex_path_to_string:N</code>	leaves it in the input stream.

---

---

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

---

---

<code>\stex_path_if_absolute_p:N</code> *	
<code>\stex_path_if_absolute:N<math>\underline{T}</math></code> *	

---

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

---

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

---

---

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

---



---

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

---

### 9.1.2 MathHub Archives

---

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

---

1. The `mathhub` package option, or
2. the `\mathhub-macro`, if it has been defined before the `\usepackage{stex}`-statement, or
3. the `MATHHUB` system variable, 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 `{<repository-name>}` (or not, if `{<repository-name>}` is empty), and passes its ID on to `{<code>}` as `#1`. Switches back to the previous repository after executing `{<code>}`.



### 9.1.3 Using Content in Archives

<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/> <code>\inputref</code> <hr/> <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-mode</code> , inserting a <i>reference</i> to the file instead. Both also set <code>\ifinputref</code> to true.
<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/> <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/> <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/> <code>\mhgraphics</code> <hr/> <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/> <code>\lstinputmhlisting</code> <hr/> <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 10

# STEX-References

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

### 10.1 Macros and Environments

---

<code>\stex_get_document_uri:</code>	Computes the current document uri from the current archive's <b>narr</b> -field and its location relative to the archive's <b>source</b> -directory. Reference targets are computed from this URI and the reference-id.
--------------------------------------	---

---

<code>\l_stex_current_docns_str</code>	Stores its result in <code>\l_stex_current_docns_str</code>
--	---

---

<code>\stex_get_document_url:</code>	Computes the current URL from the current archive's <b>docurl</b> -field and its location relative to the archive's <b>source</b> -directory. Reference targets are computed from this URL and the reference-id, if this document is only included in SMS mode.
--------------------------------------	---

---

<code>\l_stex_current_docurl_str</code>	Stores its result in <code>\l_stex_current_docurl_str</code>
---	--

#### 10.1.1 Setting Reference Targets

---

<code>\stex_ref_new_doc_target:n</code>	<code>\stex_ref_new_doc_target:n{&lt;id&gt;}</code> Sets a new reference target with id <code>&lt;id&gt;</code> .
---	--

---

<code>\stex_ref_new_sym_target:n</code>	<code>\stex_ref_new_sym_target:n{&lt;uri&gt;}</code> Sets a new reference target for the symbol <code>&lt;uri&gt;</code> .
---	---

### 10.1.2 Using References

---

`\sref` `\sref[<opt-args>]{<id>}`

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

---

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

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

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

---

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

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

# Chapter 11

## STEX-Modules

This sub package contains code related to Modules

### 11.1 Macros and Environments

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

<hr/> <hr/> <code>\c_stex_module_&lt;URI&gt;_prop</code>	A property list with the following fields: <ul style="list-style-type: none"><li><code>name</code> The <i>name</i> of the module,</li><li><code>ns</code> the <i>namespace</i> in field <code>ns</code>,</li><li><code>file</code> the <i>file</i> containing the module, as a sequence of path fragments</li><li><code>lang</code> the module's <i>language</i>,</li><li><code>sig</code> the language of the signature module, if the current file is a translation from some other language,</li><li><code>deprecate</code> if this module is deprecated, the module that replaces it,</li><li><code>meta</code> the metatheory of the module.</li></ul>
--	---

<hr/> <hr/> <code>\c_stex_module_&lt;URI&gt;_code</code>	The code to execute when this module is activated (i.e. imported), e.g. to set all the semantic macros, notations, etc.
--	---

<hr/> <hr/> <code>\c_stex_module_&lt;URI&gt;_constants</code>	The names of all constants declared in the module
---	---

<hr/> <hr/> <code>\c_stex_module_&lt;URI&gt;_constants</code>	The full URIs of all modules imported in this module
---	--

---

<code>\l_stex_current_module_str</code>	<code>\l_stex_current_module_str</code> always contains the URI of the current module (if existent).
---	--

---



---

<code>\l_stex_all_modules_seq</code>	Stores full URIs for all modules currently in scope.
--------------------------------------	--

---



---

<code>\stex_if_in_module_p: *</code>	Conditional for whether we are currently in a module
<code>\stex_if_in_module:TF *</code>	

---



---

<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.

---

<code>\stex_add_to_current_module:n</code>	
<code>\STEXexport</code>	

---

Adds the provided tokens to the `_code` control sequence of the current module.

`\stex_add_to_current_module:n` is used internally, `\STEXexport` is intended for users and additionally executes the provided code immediately.

---

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

---

Adds the declaration with the provided name to the `_constants` control sequence of the current module.

---

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

---

Adds the module with the provided full URI to the `_imports` control sequence of the current module.

---

<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>
--------------------------------------	---

---



---

<code>\stex_do_up_to_module:n</code>	Code that is <i>exported</i> from module (such as symbol declarations) should be local <i>to the current module</i> . For that reason, ideally all symbol declarations and similar commands should be called directly in the module environment, however, that is not always feasible, e.g. in structural features or <code>sparapraphs</code> . <code>\stex_do_up_to_module</code> therefore executes the provided code repeatedly in an <code>\aftergroup</code> up until the group level is equal to that of the innermost smodule environment.
--------------------------------------	--

---

---

`\stex_modules_current_namespace:`

---

Computes the current namespace as follows:

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

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

### 11.1.1 The `smodule` environment

`module (env.) \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.

---

---

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

---

## Chapter 12

# STEX-Module Inheritance

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

### 12.1 Macros and Environments

#### 12.1.1 SMS Mode

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

---

`\g_stex_smsmode_allowedmacros_tl`

---

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

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

---

`\g_stex_smsmode_allowedmacros_escape_tl`

---

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

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

---

`\g_stex_smsmode_allowedenvs_seq`

---

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

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

---

`\stex_if_smsmode_p:` ★ Tests whether SMS mode is currently active.  
`\stex_if_smsmode:` *TF* ★

---



---

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

---

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

---

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

---

## 12.1.2 Imports and Inheritance

---

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

---

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

---

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

---

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

---

$\backslash\text{stex\_import\_module\_uri:nn}$	$\backslash\text{stex\_import\_module\_uri:nn } \{ \langle \text{archive-ID} \rangle \} \{ \langle \text{module-path} \rangle \}$
---	---

---

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

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

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

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

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

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

2. Otherwise:

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

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

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

If a module by that namespace exists, it is returned. Otherwise, we call  $\backslash\text{stex\_require\_module:nn}$  on the **source** directory of the archive to find the file.

---

$\backslash\text{l\_stex\_import\_name\_str}$ $\backslash\text{l\_stex\_import\_archive\_str}$ $\backslash\text{l\_stex\_import\_path\_str}$ $\backslash\text{l\_stex\_import\_ns\_str}$	stores the result in these four variables.
---	--

---



---

$\backslash\text{stex\_import\_require\_module:nnnn}$	$\{ \langle \text{ns} \rangle \} \{ \langle \text{archive-ID} \rangle \} \{ \langle \text{path} \rangle \} \{ \langle \text{name} \rangle \}$
---	---

---

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

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

# Chapter 13

## STEX-Symbols

Code related to symbol declarations and notations

### 13.1 Macros and Environments

---

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

---

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

# Chapter 14

## STEX-Terms

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

### 14.1 Macros and Environments

---

<code>\STEXsymbol</code>	Uses <code>\stex_get_symbol:n</code> to find the symbol denoted by the first argument and passes the result on to <code>\stex_invoke_symbol:n</code>
--------------------------	--

---

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

---

<code>\stex_invoke_symbol:n</code>	Executes a semantic macro. Outside of math mode or if followed by <code>*</code> , it continues to <code>\stex_term_custom:nn</code> . In math mode, it uses the default or optionally provided notation of the associated symbol.
------------------------------------	--

If followed by `!`, 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}`.

---

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

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.

---

<code>\STEXInternalTermMathArgiii</code>	<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/> <code>\STEXInternalTermMathAssocArgiiii</code> <hr/>	<code>\stex_term_arg:nnn⟨int⟩⟨prec⟩⟨notation⟩⟨type⟩⟨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/> <code>\infpref</code> <code>\neginfpref</code> <hr/>	Maximal and minimal notation precedences.
<hr/> <code>\dobrackets</code> <hr/>	<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/> <code>\withbrackets</code> <hr/>	<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/> <code>\stex_term_custom:nn</code> <hr/>	<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/> <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> <hr/>	<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/> <code>\STEXinvisible</code> <hr/>	Exports its argument as OMDOC (invisible), but does not produce PDF output. Useful e.g. for semantic macros that take arguments that are not part of the symbolic notation.
<hr/> <code>\ellipses</code> <hr/>	TODO

## Chapter 15

# TeX-Structural Features

Code related to structural features

### 15.1 Macros and Environments

#### 15.1.1 Structures

`mathstructure` (*env.*) TODO

## Chapter 16

# TeX-Statements

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

### 16.1 Macros and Environments

`symboldoc (env.) \begin{<symboldoc>}{<symbols>} <text> \end{<symboldoc>}`

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



## Chapter 17

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

## Chapter 18

# sTeX-Metatheory

### 18.1 Symbols

**Part III**  
**Extensions**

## Chapter 19

# Tikzinput: Treating TIKZ code as images

### 19.1 Macros and Environments

## Chapter 20

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

## Chapter 21

# NotesSlides – Slides and Course Notes

## Chapter 22

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

## Chapter 23

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



Part IV

# Implementation

## Chapter 24

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

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

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

```
1 <*cls>
2
3 %%%%%%%%% basics.dtx %%%%%%%%%
4
5 \RequirePackage{expl3,l3keys2e}
6 \ProvidesExplClass{stex}{2022/08/08}{3.2.0}{sTeX document class}
7
8 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
9 \ProcessOptions
10
11 \bool_set_true:N \c_stex_document_class_bool
12
13 \RequirePackage{stex}
14
15 \stex_html_backend:TF {
16   \LoadClass{article}
17 }{
18   \LoadClass[border=1px,varwidth,crop=false]{standalone}
19   \setlength\textwidth{15cm}
20 }
21 \RequirePackage{standalone}
22
23
24 \clist_if_empty:NT \c_stex_languages_clist {
25   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
26   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
27   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
28   \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
29     \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
30       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
```

```

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

```

## 24.2 Preliminaries

```

44 <*package>
45
46 %%%%%%%%% basics.dtx %%%%%%%%%
47
48 \RequirePackage{expl3,l3keys2e,ltxcmds}
49 \ProvidesExplPackage{stex}{2022/08/08}{3.2.0}{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~3.2.0~*^^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 71.)

## 24.3 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 71.)

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}

```

## 24.4 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 71.)

`\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 71.)

`\stex_annotate_html:nn`

`\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>ML</sub>, R<sub>U</sub>S<sub>T</sub>E<sub>X</sub>, p<sub>D</sub>F<sub>L</sub>A<sub>T</sub>E<sub>X</sub>).

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

```

132 \ifcsname if@rustex\endcsname\else
133   \expandafter\newif\csname if@rustex\endcsname
134   \@rustexfalse
135 \fi
136 \ifcsname if@latexml\endcsname\else
137   \expandafter\newif\csname if@latexml\endcsname
138   \@latexmlfalse
139 \fi
140 \tl_if_exist:NF\stex@backend{
141   \if@rustex
142     \def\stex@backend{rustex}
143   \else
144     \if@latexml
145       \def\stex@backend{latexml}
146     \else

```

```

147     \cs_if_exist:NTF\HCode{
148       \def\stex@backend{tex4ht}
149     }{
150       \def\stex@backend{pdflatex}
151     }
152     \fi
153     \fi
154   }
155   \input{stex-backend-\stex@backend.cfg}
156
157   \newif\ifstexhtml
158   \stex_html_backend:TF\stexhtmltrue\stexhtmlfalse
159

```

(End definition for `\stex_annotate:nnn`, `\stex_annotate_invisible:n`, and `\stex_annotate_invisible:nnn`. These functions are documented on page 72.)

## 24.5 Babel Languages

```

160 <@=stex_language>

```

`\c_stex_languages_prop`  
`\c_stex_language_abbrevs_prop`

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

```

161 \exp_args:NNx \prop_const_from_keyval:Nn \c_stex_languages_prop { \tl_to_str:n {
162   en = english ,
163   de = ngerman ,
164   ar = arabic ,
165   bg = bulgarian ,
166   ru = russian ,
167   fi = finnish ,
168   ro = romanian ,
169   tr = turkish ,
170   fr = french
171 }}
172
173 \exp_args:NNx \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop { \tl_to_str:n {
174   english   = en ,
175   ngerman   = de ,
176   arabic    = ar ,
177   bulgarian = bg ,
178   russian   = ru ,
179   finnish   = fi ,
180   romanian  = ro ,
181   turkish   = tr ,
182   french    = fr
183 }}
184 % todo: chinese simplified (zhs)
185 %       chinese traditional (zht)

```

(End definition for `\c_stex_languages_prop` and `\c_stex_language_abbrevs_prop`. These variables are documented on page 72.)

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

```

186 \cs_new_protected:Nn \stex_set_language:Nn {
187   \str_set:Nx \l_tmpa_str {#2}
188   \prop_get:NoNT \c_stex_languages_prop \l_tmpa_str #1 {

```

```

189 \ifx\@onlypreamble\@notprerr
190 \ltx@ifpackageloaded{babel}{
191 \exp_args:No \selectlanguage #1
192 }{}
193 \else
194 \exp_args:No \str_if_eq:nnTF #1 {turkish} {
195 \RequirePackage[#1,shorthands=:!]{babel}
196 }{
197 \RequirePackage[#1]{babel}
198 }
199 \fi
200 }
201 }
202
203 \clist_if_empty:NF \c_stex_languages_clist {
204 \bool_set_false:N \l_tmpa_bool
205 \clist_clear:N \l_tmpa_clist
206 \clist_map_inline:Nn \c_stex_languages_clist {
207 \str_set:Nx \l_tmpa_str {#1}
208 \str_if_eq:nnT {#1}{tr}{
209 \bool_set_true:N \l_tmpa_bool
210 }
211 \prop_get:NoNTF \c_stex_languages_prop \l_tmpa_str \l_tmpa_str {
212 \clist_put_right:No \l_tmpa_clist \l_tmpa_str
213 } {
214 \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
215 }
216 }
217 \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
218 \bool_if:NTF \l_tmpa_bool {
219 \RequirePackage[\clist_use:Nn \l_tmpa_clist,,shorthands=:!]{babel}
220 }{
221 \RequirePackage[\clist_use:Nn \l_tmpa_clist,]{babel}
222 }
223 }
224
225 \AtBeginDocument{
226 \stex_html_backend:T {
227 \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
228 \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
229 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
230 \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
231 \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
232 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
233 \stex_debug:nn{basics} {Language~\l_tmpa_str~
234 inferred~from~file~name}
235 \stex_annotate_invisible:nnn{language}{ \l_tmpa_str }{}
236 }
237 }
238 }
239

```

## 24.6 Persistence

```

240 <@@=stex_persist>
241 \bool_if:NTF \c_stex_persist_mode_bool {
242   \def \stex_persist:n #1 {}
243   \def \stex_persist:x #1 {}
244 }{
245   \bool_if:NTF \c_stex_persist_write_mode_bool {
246     \iow_new:N \c__stex_persist_iow
247     \iow_open:Nn \c__stex_persist_iow{\jobname.sms}
248     \AtEndDocument{
249       \iow_close:N \c__stex_persist_iow
250     }
251     \cs_new_protected:Nn \stex_persist:n {
252       \tl_set:Nn \l_tmpa_tl { #1 }
253       \regex_replace_all:nnN { \cP\# } { \cO\# } \l_tmpa_tl
254       \regex_replace_all:nnN { \ } { \~ } \l_tmpa_tl
255       \exp_args:NNo \iow_now:Nn \c__stex_persist_iow \l_tmpa_tl
256     }
257     \cs_generate_variant:Nn \stex_persist:n {x}
258   }{
259     \def \stex_persist:n #1 {}
260     \def \stex_persist:x #1 {}
261   }
262 }

```

## 24.7 Auxiliary Methods

`\stex_deactivate_macro:Nn`

```

263 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
264   \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
265   \def#1{
266     \msg_error:nnnn{stex}{error/deactivated-macro}{\detokenize{#1}}{#2}
267   }
268 }

```

(End definition for `\stex_deactivate_macro:Nn`. This function is documented on page 72.)

`\stex_reactivate_macro:N`

```

269 \cs_new_protected:Nn \stex_reactivate_macro:N {
270   \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname
271 }

```

(End definition for `\stex_reactivate_macro:N`. This function is documented on page 72.)

`\ignorespacesandpars`

```

272 \protected\def\ignorespacesandpars{
273   \begingroup\catcode13=10\relax
274   \@ifnextchar\par{
275     \endgroup\expandafter\ignorespacesandpars\@gobble
276   }{
277     \endgroup
278   }
279 }

```



```

280
281 \cs_new_protected:Nn \stex_copy_control_sequence:NNN {
282   \tl_set:Nx \_tmp_args_tl {\cs_argument_spec:N #2}
283   \exp_args:NNo \tl_remove_all:Nn \_tmp_args_tl \c_hash_str
284   \int_set:Nn \l_tmpa_int {\tl_count:N \_tmp_args_tl}
285
286   \tl_clear:N \_tmp_args_tl
287   \int_step_inline:nn \l_tmpa_int {
288     \tl_put_right:Nx \_tmp_args_tl {{\exp_not:n{####}\exp_not:n{##1}}}
289   }
290
291   \tl_set:Nn #3 {\cs_generate_from_arg_count:NNnn #1 \cs_set:Npn}
292   \tl_put_right:Nx #3 { {\int_use:N \l_tmpa_int}{
293     \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
294     \exp_after:wN\exp_after:wN\exp_after:wN\exp_after:wN {
295       \exp_after:wN #2 \_tmp_args_tl
296     }
297   }}
298 }
299 \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {cNN}
300 \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {NcN}
301 \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {ccN}
302
303 \cs_new_protected:Nn \stex_copy_control_sequence_ii:NNN {
304   \tl_set:Nx \_tmp_args_tl {\cs_argument_spec:N #2}
305   \exp_args:NNo \tl_remove_all:Nn \_tmp_args_tl \c_hash_str
306   \int_set:Nn \l_tmpa_int {\tl_count:N \_tmp_args_tl}
307
308   \tl_clear:N \_tmp_args_tl
309   \int_step_inline:nn \l_tmpa_int {
310     \tl_put_right:Nx \_tmp_args_tl {{\exp_not:n{#####}\exp_not:n{##1}}}
311   }
312
313   \edef \_tmp_args_tl {
314     \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
315     \exp_after:wN\exp_after:wN\exp_after:wN {
316       \exp_after:wN #2 \_tmp_args_tl
317     }
318   }
319
320   \exp_after:wN \def \exp_after:wN \_tmp_args_tl
321   \exp_after:wN ##\exp_after:wN 1 \exp_after:wN ##\exp_after:wN 2
322   \exp_after:wN { \_tmp_args_tl }
323
324   \edef \_tmp_args_tl {
325     \exp_after:wN \exp_not:n \exp_after:wN {
326       \_tmp_args_tl {####1}{####2}
327     }
328   }
329
330   \tl_set:Nn #3 {\cs_generate_from_arg_count:NNnn #1 \cs_set:Npn}
331   \tl_put_right:Nx #3 { {\int_use:N \l_tmpa_int}{
332     \exp_after:wN\exp_not:n\exp_after:wN{\_tmp_args_tl}
333   }}

```

```

334 }
335
336 \cs_generate_variant:Nn \stex_copy_control_sequence_ii:NNN {cNN}
337 \cs_generate_variant:Nn \stex_copy_control_sequence_ii:NNN {NcN}
338 \cs_generate_variant:Nn \stex_copy_control_sequence_ii:NNN {ccN}

```

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

\MMTrule

```

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

```

```

384 }
385
386 \cs_new_protected:Nn \stex_par: {
387   \mode_if_vertical:F{
388     \if@minipage\else\if@nobreak\else\par\fi\fi
389   }
390 }
391
392 \end{package}

```

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

## Chapter 25

# STEX -MathHub Implementation

```
393 <*package>
394
395 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
396
397 <@@=stex_path>
398
399 Warnings and error messages
400 \msg_new:nnn{stex}{error/norepository}{
401   No~archive~#1~found~in~#2
402 }
403 \msg_new:nnn{stex}{error/notinarchive}{
404   Not~currently~in~an~archive,~but~\detokenize{#1}~
405   needs~one!
406 }
407 \msg_new:nnn{stex}{error/nofile}{
408   \detokenize{#1}~could~not~find~file~#2
409 }
410 \msg_new:nnn{stex}{error/twofiles}{
411   \detokenize{#1}~found~two~candidates~for~#2
412 }
```

### 25.1 Generic Path Handling

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

`\stex_path_from_string:Nn`

```
411 \cs_new_protected:Nn \stex_path_from_string:Nn {
412   \stex_debug:nn{files}{#2}
413   \str_set:Nx \l_tmpa_str { #2 }
414   \str_if_empty:NTF \l_tmpa_str {
415     \seq_clear:N #1
416   }{
417     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
418     \sys_if_platform_windows:T{
```

```

419 \seq_clear:N \l_tmpa_tl
420 \seq_map_inline:Nn #1 {
421   \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
422   \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
423 }
424 \seq_set_eq:NN #1 \l_tmpa_tl
425 }
426 \stex_path_canonicalize:N #1
427 }
428 \stex_debug:nn{files}{Yields: \stex_path_to_string:N#1}
429 }
430

```

(End definition for `\stex_path_from_string:Nn`. This function is documented on page 73.)

```

\stex_path_to_string:NN
\stex_path_to_string:N
431 \cs_new_protected:Nn \stex_path_to_string:NN {
432   \exp_args:NNe \str_set:Nn #2 { \seq_use:Nn #1 / }
433 }
434
435 \cs_new:Nn \stex_path_to_string:N {
436   \seq_use:Nn #1 /
437 }

```

(End definition for `\stex_path_to_string:NN` and `\stex_path_to_string:N`. These functions are documented on page 73.)

```

\c__stex_path_dot_str . and .., respectively.
\c__stex_path_up_str
438 \str_const:Nn \c__stex_path_dot_str {.}
439 \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.

```

440 \cs_new_protected:Nn \stex_path_canonicalize:N {
441   \seq_if_empty:NF #1 {
442     \seq_clear:N \l_tmpa_seq
443     \seq_get_left:NN #1 \l_tmpa_tl
444     \str_if_empty:NT \l_tmpa_tl {
445       \seq_put_right:Nn \l_tmpa_seq {}
446     }
447     \seq_map_inline:Nn #1 {
448       \str_set:Nn \l_tmpa_tl { ##1 }
449       \str_if_eq:NNF \l_tmpa_tl \c__stex_path_dot_str {
450         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
451           \seq_if_empty:NTF \l_tmpa_seq {
452             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
453               \c__stex_path_up_str
454             }
455           }{
456             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
457             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
458               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
459                 \c__stex_path_up_str

```

```

460     }
461   }{
462     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
463   }
464 }
465 }{
466   \str_if_empty:NF \l_tmpa_tl {
467     \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
468   }
469 }
470 }
471 }
472 \seq_gset_eq:NN #1 \l_tmpa_seq
473 }
474 }

```

(End definition for `\stex_path_canonicalize:N`. This function is documented on page 73.)

`\stex_path_if_absolute_p:N`  
`\stex_path_if_absolute:NTF`

```

475 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
476   \seq_if_empty:NTF #1 {
477     \prg_return_false:
478   }{
479     \seq_get_left:NN #1 \l_tmpa_tl
480     \sys_if_platform_windows:TF{
481       \str_if_in:NnTF \l_tmpa_tl {:}{
482         \prg_return_true:
483       }{
484         \prg_return_false:
485       }
486     }{
487       \str_if_empty:NTF \l_tmpa_tl {
488         \prg_return_true:
489       }{
490         \prg_return_false:
491       }
492     }
493   }
494 }

```

(End definition for `\stex_path_if_absolute:NTF`. This function is documented on page 73.)

## 25.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

495 \str_new:N\l_stex_kpsewhich_return_str
496 \cs_new_protected:Nn \stex_kpsewhich:n {\begingroup
497   \catcode'\ =12
498   \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
499   \tl_gset_eq:NN \l_tmpa_tl \l_tmpa_tl
500   \endgroup
501   \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
502   \tl_trim_spaces:N \l_stex_kpsewhich_return_str
503 }

```

(End definition for `\stex_kpsewhich:n`. This function is documented on page 73.)

We determine the PWD

`\c_stex_pwd_seq`  
`\c_stex_pwd_str`

```

504 \sys_if_platform_windows:TF{
505   \begingroup\escapechar=-1\catcode'\=12
506   \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
507   \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
508   \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_
509   }}{
510   \stex_kpsewhich:n{-var-value~PWD}
511   }
512
513 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
514 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
515 \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 73.)

## 25.3 File Hooks and Tracking

516 `<@@=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

```

517 \seq_gclear_new:N\g__stex_files_stack

```

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`  
`\c_stex_mainfile_str`

```

518 \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
519 \stex_path_from_string:Nn \c_stex_mainfile_seq
520   \c_stex_mainfile_str

```

(End definition for `\c_stex_mainfile_seq` and `\c_stex_mainfile_str`. These variables are documented on page 73.)

`\g_stex_currentfile_seq`

```

521 \seq_gclear_new:N\g_stex_currentfile_seq

```

(End definition for `\g_stex_currentfile_seq`. This variable is documented on page 74.)

`\stex_filestack_push:n`

```

522 \cs_new_protected:Nn \stex_filestack_push:n {
523   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
524   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
525     \stex_path_from_string:Nn\g_stex_currentfile_seq{
526       \c_stex_pwd_str/#1
527     }

```

```

528 }
529 \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
530 \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
531 \stex_get_document_uri:
532 }

```

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

`\stex_filestack_pop:`

```

533 \cs_new_protected:Nn \stex_filestack_pop: {
534   \seq_if_empty:NF\g__stex_files_stack{
535     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
536   }
537   \seq_if_empty:NTF\g__stex_files_stack{
538     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
539   }{
540     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
541     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
542   }
543   \stex_get_document_uri:
544 }

```

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

Hooks for the current file:

```

545 \AddToHook{file/before}{
546   \tl_if_empty:NTF\CurrentFilePath{
547     \stex_filestack_push:n{\CurrentFile}
548   }{
549     \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
550   }
551 }
552 \AddToHook{file/after}{
553   \stex_filestack_pop:
554 }

```

## 25.4 MathHub Repositories

```

555 <@=stex_mathhub>

```

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

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

```

556 \str_if_empty:NTF\mathhub{
557   \sys_if_platform_windows:TF{
558     \begingroup\escapechar=-1\catcode'\=12
559     \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str MATHHUB\c_percent_str}
560     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
561     \exp_args:NNx\str_if_eq:onT\l_stex_kpsewhich_return_str{\c_percent_str MATHHUB\c_percent_str}
562     \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_kpsewhich_return_str}}
563   }{
564     \stex_kpsewhich:n{-var-value-MATHHUB}
565   }
566   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
567 }

```



```

568 \str_if_empty:NT \c_stex_mathhub_str {
569   \sys_if_platform_windows:TF{
570     \begingroup\escapechar=-1\catcode'\=12
571     \exp_args:Nx\stex_kpsewhich:n{-var-value-HOME}
572     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
573     \exp_args:NNx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_s
574   }{
575     \stex_kpsewhich:n{-var-value-HOME}
576   }
577   \ior_open:NnT \g_tmpa_ior{\l_stex_kpsewhich_return_str / .stex / mathhub.path}{
578     \begingroup\escapechar=-1\catcode'\=12
579     \ior_str_get:NN \g_tmpa_ior \l_tmpa_str
580     \sys_if_platform_windows:T{
581       \exp_args:NNx\str_replace_all:Nnn\l_tmpa_str{\c_backslash_str}/
582     }
583     \str_gset_eq:NN \c_stex_mathhub_str\l_tmpa_str
584     \endgroup
585     \ior_close:N \g_tmpa_ior
586   }
587 }
588 \str_if_empty:NTF\c_stex_mathhub_str{
589   \msg_warning:nn{stex}{warning/nomathhub}
590 }{
591   \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
592   \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
593 }
594 }{
595   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
596   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
597     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
598       \c_stex_pwd_str/\mathhub
599     }
600   }
601   \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
602   \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
603 }

```

(End definition for `\mathhub`, `\c_stex_mathhub_seq`, and `\c_stex_mathhub_str`. These variables are documented on page 74.)

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

```

604 \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
605   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
606     \str_set:Nx \l_tmpa_str { #1 }
607     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
608     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
609     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
610     \__stex_mathhub_find_manifest:N \l_tmpa_seq
611     \seq_if_empty:NTF \l_stex_mathhub_manifest_file_seq {
612       \msg_error:nnxx{stex}{error/norepository}{#1}{
613         \stex_path_to_string:N \c_stex_mathhub_str
614       }
615       \input{Fatal-Error!}

```

```

616     } {
617     \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
618     }
619   }
620 }

```

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

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

```

621 \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:

```

622 \cs_new_protected:Nn \__stex_mathhub_find_manifest:N {
623   \seq_set_eq:NN\l_tmpa_seq #1
624   \bool_set_true:N\l_tmpa_bool
625   \bool_while_do:Nn \l_tmpa_bool {
626     \seq_if_empty:NTF \l_tmpa_seq {
627       \bool_set_false:N\l_tmpa_bool
628     }{
629       \file_if_exist:nTF{
630         \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
631       }{
632         \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
633         \bool_set_false:N\l_tmpa_bool
634       }{
635         \file_if_exist:nTF{
636           \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
637         }{
638           \seq_put_right:Nn\l_tmpa_seq{META-INF}
639           \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
640           \bool_set_false:N\l_tmpa_bool
641         }{
642           \file_if_exist:nTF{
643             \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
644           }{
645             \seq_put_right:Nn\l_tmpa_seq{meta-inf}
646             \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
647             \bool_set_false:N\l_tmpa_bool
648           }{
649             \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
650           }
651         }
652       }
653     }
654   }
655   \seq_set_eq:NN\l__stex_mathhub_manifest_file_seq\l_tmpa_seq
656 }

```

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

\c\_stex\_mathhub\_manifest\_ior

File variable used for MANIFEST-files

```

657 \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:

```

658 \cs_new_protected:Nn \_stex_mathhub_parse_manifest:n {
659   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
660   \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
661   \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
662     \str_set:Nn \l_tmpa_str {##1}
663     \exp_args:NNo \seq_set_split:Nnn
664       \l_tmpb_seq \c_colon_str \l_tmpa_str
665     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
666       \exp_args:NNe \str_set:Nn \l_tmpb_tl {
667         \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
668       }
669       \exp_args:No \str_case:nnTF \l_tmpa_tl {
670         {id} {
671           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
672             { id } \l_tmpb_tl
673         }
674         {narration-base} {
675           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
676             { narr } \l_tmpb_tl
677         }
678         {url-base} {
679           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
680             { docurl } \l_tmpb_tl
681         }
682         {source-base} {
683           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
684             { ns } \l_tmpb_tl
685         }
686         {ns} {
687           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
688             { ns } \l_tmpb_tl
689         }
690         {dependencies} {
691           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
692             { deps } \l_tmpb_tl
693         }
694       }{}{}
695     }{}
696   }
697   \ior_close:N \c__stex_mathhub_manifest_ior
698   \stex_persist:x {
699     \prop_set_from_keyval:cn{ c_stex_mathhub_#1_manifest_prop }{
700       \exp_after:wN \prop_to_keyval:N \csname c_stex_mathhub_#1_manifest_prop\endcsname
701     }
702   }
703 }
```

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

`\stex_set_current_repository:n`

```

704 \cs_new_protected:Nn \stex_set_current_repository:n {
```

```

705 \stex_require_repository:n { #1 }
706 \prop_set_eq:Nc \l_stex_current_repository_prop {
707   c_stex_mathhub_#1_manifest_prop
708 }
709 }

```

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

`\stex_require_repository:n`

```

710 \cs_new_protected:Nn \stex_require_repository:n {
711   \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
712     \stex_debug:nn{mathhub}{Opening~archive:~#1}
713     \__stex_mathhub_do_manifest:n { #1 }
714   }
715 }

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

716 %\prop_new:N \l_stex_current_repository_prop
717 \bool_if:NF \c_stex_persist_mode_bool {
718   \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
719   \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
720     \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
721   } {
722     \__stex_mathhub_parse_manifest:n { main }
723     \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
724     \l_tmpa_str
725     \prop_set_eq:cN { c_stex_mathhub_#1_manifest_prop }
726     \c_stex_mathhub_main_manifest_prop
727     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
728     \stex_debug:nn{mathhub}{Current~repository:~
729     \prop_item:Nn \l_stex_current_repository_prop {id}
730   }
731 }
732 }

```

(End definition for `\l_stex_current_repository_prop`. This variable is documented on page 74.)

`\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.

```

733 \cs_new_protected:Nn \stex_in_repository:nn {
734   \str_set:Nx \l_tmpa_str { #1 }
735   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
736   \str_if_empty:NTF \l_tmpa_str {
737     \prop_if_exist:NTF \l_stex_current_repository_prop {
738       \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
739       \exp_args:Ne \l_tmpa_cs{
740         \prop_item:Nn \l_stex_current_repository_prop { id }
741       }
742     }{
743       \l_tmpa_cs{}
744     }
745   }{

```

```

746 \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}
747 \stex_require_repository:n \l_tmpa_str
748 \str_set:Nx \l_tmpa_str { #1 }
749 \exp_args:Nne \use:nn {
750   \stex_set_current_repository:n \l_tmpa_str
751   \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
752 }{
753   \stex_debug:nn{mathhub}{switching~back~to:~
754     \prop_if_exist:NTF \l_stex_current_repository_prop {
755       \prop_item:Nn \l_stex_current_repository_prop { id }::~
756       \meaning\l_stex_current_repository_prop
757     }{
758       no~repository
759     }
760   }
761   \prop_if_exist:NTF \l_stex_current_repository_prop {
762     \stex_set_current_repository:n {
763       \prop_item:Nn \l_stex_current_repository_prop { id }
764     }
765   }{
766     \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
767   }
768 }
769 }
770 }

```

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

## 25.5 Using Content in Archives

`\mhpath`

```

771 \def \mhpath #1 #2 {
772   \exp_args:Ne \tl_if_empty:nTF{#1}{
773     \c_stex_mathhub_str /
774     \prop_item:Nn \l_stex_current_repository_prop { id }
775     / source / #2
776   }{
777     \c_stex_mathhub_str / #1 / source / #2
778   }
779 }

```

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

`\inputref`

`\mhinput`

```

780 \newif \ifinputref \inputreffalse
781
782 \cs_new_protected:Nn \__stex_mathhub_mhinput:nn {
783   \stex_in_repository:nn {#1} {
784     \ifinputref
785       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
786     \else
787       \inputreftrue
788       \input{ \c_stex_mathhub_str / ##1 / source / #2 }

```

```

789     \inputreffalse
790   \fi
791 }
792 }
793 \NewDocumentCommand \mhinput { 0{} m}{
794   \_stex_mathhub_mhinput:nn{ #1 }{ #2 }
795 }
796
797 \cs_new_protected:Nn \_stex_mathhub_inputref:nn {
798   \stex_in_repository:nn {#1} {
799     \stex_html_backend:TF {
800       \str_clear:N \l_tmpa_str
801       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
802         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
803       }
804
805       \tl_if_empty:nTF{ ##1 }{
806         \IfFileExists{#2}{
807           \stex_annotate_invisible:nnn{inputref}{
808             \l_tmpa_str / #2
809           }{}
810         }{
811           \input{#2}
812         }
813       }{
814         \IfFileExists{ \c_stex_mathhub_str / ##1 / source / #2 }{
815           \stex_annotate_invisible:nnn{inputref}{
816             \l_tmpa_str / #2
817           }{}
818         }{
819           \input{ \c_stex_mathhub_str / ##1 / source / #2 }
820         }
821       }
822
823     }{
824       \begingroup
825       \inputreftrue
826       \tl_if_empty:nTF{ ##1 }{
827         \input{#2}
828       }{
829         \input{ \c_stex_mathhub_str / ##1 / source / #2 }
830       }
831     } \endgroup
832   }
833 }
834 }
835 \NewDocumentCommand \inputref { 0{} m}{
836   \_stex_mathhub_inputref:nn{ #1 }{ #2 }
837 }

```

(End definition for `\inputref` and `\mhinput`. These functions are documented on page 75.)

`\addmhbibresource`

```

838 \cs_new_protected:Nn \_stex_mathhub_mhbibresource:nn {

```

```

839 \stex_in_repository:nn {#1} {
840   \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
841 }
842 }
843 \newcommand\addmhbibresource[2][]{
844   \__stex_mathhub_mhbibresource:nn{ #1 }{ #2 }
845 }

```

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

### \libinput

```

846 \cs_new_protected:Npn \libinput #1 {
847   \prop_if_exist:NF \l_stex_current_repository_prop {
848     \msg_error:nnn{stex}{error/notinarchive}\libinput
849   }
850   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
851     \msg_error:nnn{stex}{error/notinarchive}\libinput
852   }
853   \seq_clear:N \l__stex_mathhub_libinput_files_seq
854   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
855   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
856
857   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
858     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}
859     \IfFileExists{ \l_tmpa_str }{
860       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
861     }{}
862     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
863     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
864   }
865
866   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
867   \IfFileExists{ \l_tmpa_str }{
868     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
869   }{}
870
871   \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
872     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
873   }{
874     \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
875       \input{ ##1 }
876     }
877   }
878 }

```

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

### \libusepackage

```

879 \NewDocumentCommand \libusepackage {0{ } m} {
880   \prop_if_exist:NF \l_stex_current_repository_prop {
881     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
882   }
883   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
884     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
885   }

```

```

886 \seq_clear:N \l__stex_mathhub_libinput_files_seq
887 \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
888 \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
889
890 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
891   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2}
892   \IfFileExists{ \l_tmpa_str.sty }{
893     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
894   }{
895     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
896     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
897   }
898
899 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2}
900 \IfFileExists{ \l_tmpa_str.sty }{
901   \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
902 }{
903
904 \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
905   \msg_error:nxxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
906 }{
907   \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
908     \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
909       \usepackage[#1]{ #1 }
910     }
911   }{
912     \msg_error:nxxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
913   }
914 }
915 }

```

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

`\mhgraphics`  
`\cmhgraphics`

```

916
917 \AddToHook{begindocument}{
918 \ltx@ifpackageloaded{graphicx}{
919   \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
920   \providecommand\mhgraphics[2] [] {%
921     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
922     \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}
923   \providecommand\cmhgraphics[2] [] {\begin{center}\mhgraphics[#1]{#2}\end{center}}
924 }{

```

(End definition for `\mhgraphics` and `\cmhgraphics`. These functions are documented on page 75.)

`\lstinputmhlisting`  
`\clstinputmhlisting`

```

925 \ltx@ifpackageloaded{listings}{
926   \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
927   \newcommand\lstinputmhlisting[2] [] {%
928     \def\lst@mhrepos{}\setkeys{lst}{#1}%
929     \lstinputlisting[#1]{\mhp\lst@mhrepos{#2}}
930   \newcommand\clstinputmhlisting[2] [] {\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
931 }{
932 }

```



933

934 `\end{package}`

*(End definition for `\lstinputmhlisting` and `\clstinputmhlisting`. These functions are documented on page 75.)*

## Chapter 26

# STEX -References Implementation

```
935 <*package>
936
937 %%%%%%%%% stex-references.dtx %%%%%%%%%
938
939 <@@=stex_refs>
    Warnings and error messages
940 \msg_new:nnn{stex}{error/extrefmissing}{
941   Missing~in~or~cite~value~for~\detokenize{\extref}!
942 }
943 \msg_new:nnn{stex}{warning/smsmissing}{
944   .sref~file~#1~doesn't~exist!
945 }
946 \msg_new:nnn{stex}{warning/smslabelmissing}{
947   No~label~#2~in~.sref~file~#1!
948 }
```

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

```
949 \iow_new:N \c__stex_refs_refs_iow
950 \AtBeginDocument{
951   \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
952 }
953 \AtEndDocument{
954   \iow_close:N \c__stex_refs_refs_iow
955 }
```

### 26.1 Document URIs and URLs

`\l_stex_current_docns_str`

```
956 \str_new:N \l_stex_current_docns_str
```

*(End definition for `\l_stex_current_docns_str`. This variable is documented on page 76.)*

`\stex_get_document_uri:`

```
957 \cs_new_protected:Nn \stex_get_document_uri: {
```

```

958 \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
959 \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
960 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
961 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
962 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
963
964 \str_clear:N \l_tmpa_str
965 \prop_if_exist:NT \l_stex_current_repository_prop {
966   \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
967     \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
968   }
969 }
970
971 \str_if_empty:NTF \l_tmpa_str {
972   \str_set:Nx \l_stex_current_docns_str {
973     file:/\stex_path_to_string:N \l_tmpa_seq
974   }
975 }{
976   \bool_set_true:N \l_tmpa_bool
977   \bool_while_do:Nn \l_tmpa_bool {
978     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
979     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
980       {source} { \bool_set_false:N \l_tmpa_bool }
981     }{}{
982       \seq_if_empty:NT \l_tmpa_seq {
983         \bool_set_false:N \l_tmpa_bool
984       }
985     }
986   }
987
988   \seq_if_empty:NTF \l_tmpa_seq {
989     \str_gset_eq:NN \l_stex_current_docns_str \l_tmpa_str
990   }{
991     \str_gset:Nx \l_stex_current_docns_str {
992       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
993     }
994   }
995 }
996 %\stex_get_document_url:
997 }

```

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

`\l_stex_current_docurl_str`

```

998 \str_new:N \l_stex_current_docurl_str

```

(End definition for `\l_stex_current_docurl_str`. This variable is documented on page 76.)

`\stex_get_document_url:`

```

999 \cs_new_protected:Nn \stex_get_document_url: {
1000   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1001   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1002   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1003   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1004   \seq_put_right:No \l_tmpa_seq \l_tmpb_str

```

```

1005
1006 \str_clear:N \l_tmpa_str
1007 \prop_if_exist:NT \l_stex_current_repository_prop {
1008   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
1009     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
1010       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
1011     }
1012   }
1013 }
1014
1015 \str_if_empty:NTF \l_tmpa_str {
1016   \str_set:Nx \l_stex_current_docurl_str {
1017     file:/\stex_path_to_string:N \l_tmpa_seq
1018   }
1019 }{
1020   \bool_set_true:N \l_tmpa_bool
1021   \bool_while_do:Nn \l_tmpa_bool {
1022     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1023     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1024       {source} { \bool_set_false:N \l_tmpa_bool }
1025     }{}{
1026       \seq_if_empty:NT \l_tmpa_seq {
1027         \bool_set_false:N \l_tmpa_bool
1028       }
1029     }
1030   }
1031
1032   \seq_if_empty:NTF \l_tmpa_seq {
1033     \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
1034   }{
1035     \str_set:Nx \l_stex_current_docurl_str {
1036       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
1037     }
1038   }
1039 }
1040 }

```

(End definition for `\stex_get_document_url`.. This function is documented on page 76.)

## 26.2 Setting Reference Targets

```

1041 \str_const:Nn \c__stex_refs_url_str{URL}
1042 \str_const:Nn \c__stex_refs_ref_str{REF}
1043 \str_new:N \l__stex_refs_curr_label_str
1044 % @currentlabel -> number
1045 % @currentlabelname -> title
1046 % @currentHref -> name.number <- id of some kind
1047 % @currentcounter <- name/id
1048 % \#autorefname <- "Section"
1049 % \theH# -> \arabic{section}
1050 % \the# -> number
1051 % \hyper@makecurrent{#}
1052 \int_new:N \l__stex_refs_unnamed_counter_int

```

Restoring references from .sref-files

\STEXInternalSrefRestoreTarget

1053 \cs\_new\_protected:Npn \STEXInternalSrefRestoreTarget #1#2#3#4#5 {}

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

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

```

1054 \seq_new:N \g_stex_ref_files_seq
1055
1056 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
1057   %\stex_get_document_uri:
1058   \str_clear:N \l__stex_refs_curr_label_str
1059   \str_set:Nx \l_tmpa_str { #1 }
1060   \str_if_empty:NT \l_tmpa_str {
1061     \int_gincr:N \l__stex_refs_unnamed_counter_int
1062     \str_set:Nx \l_tmpa_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
1063   }
1064   \str_set:Nx \l__stex_refs_curr_label_str {
1065     \l_stex_current_docns_str?\l_tmpa_str
1066   }
1067
1068   \exp_args:Noo \STEXInternalAuxAddDocRef\l_stex_current_docns_str\l_tmpa_str
1069
1070   %\seq_if_exist:cF{g__stex_refs_labels_\l_tmpa_str_seq}{
1071   %   \seq_new:c {g__stex_refs_labels_\l_tmpa_str_seq}
1072   %}
1073   %\seq_if_in:coF{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str {
1074   %   \seq_gput_right:co{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str
1075   %}
1076
1077
1078   \stex_if_smsmode:TF {
1079     %\stex_get_document_url:
1080     %\str_gset_eq:cN {sref_url_\l__stex_refs_curr_label_str_str}\l_stex_current_docurl_str
1081     %\str_gset_eq:cN {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_url_str
1082   }{
1083     \iow_now:Nx \c__stex_refs_refs_iow {
1084       \STEXInternalSrefRestoreTarget
1085       {\l_stex_current_docns_str}
1086       {\l_tmpa_str}
1087       {\@currentcounter}
1088       {\@currentlabel}
1089       {\tl_if_exist:NT\@currentlabelname{\exp_args:No\unexpanded\@currentlabelname}}
1090     }
1091     %\iow_now:Nx \c__stex_refs_refs_iow {
1092     %   {\l_stex_current_docns_str?\l_tmpa_str}~==~{\use:c{\@currentcounter autorefname}~\@currentlabelname}
1093     \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
1094     \immediate\write\@auxout{\STEXInternalAuxAddDocRef{\l_stex_current_docns_str}{\l_tmpa_str}}
1095     %\str_gset:cx {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_ref_str
1096   }
1097 }
1098 \NewDocumentCommand \slabel {m} {\stex_ref_new_doc_target:n {#1}}

```

(End definition for \stex\_ref\_new\_doc\_target:n. This function is documented on page 76.)

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

```

1099 \cs_new_protected:Npn \STEXInternalAuxAddDocRef #1 #2 {
1100   \exp_args:Nnx \seq_if_in:NnTF \g_stex_ref_files_seq {\detokenize{#1}} {
1101     \exp_args:Nnx \seq_if_in:cnF{g_stex_ref_ #1 _seq}{\detokenize{#2}}{
1102       \exp_args:Nnx \seq_gput_left:cn{g_stex_ref_ #1 _seq}{\detokenize{#2}}
1103     }
1104   }{
1105     \exp_args:Nnx \seq_gput_right:Nn \g_stex_ref_files_seq {\detokenize{#1}}
1106     %\seq_if_exist:cF{g_stex_ref_ #1 _seq}{
1107       \seq_new:c{g_stex_ref_ #1 _seq} % <- seq_new throws errors??
1108     %}
1109     \exp_args:Nnx \seq_gput_left:cn{g_stex_ref_ #1 _seq}{\detokenize{#2}}
1110   }
1111
1112   %\str_set:Nn \l_tmpa_str {#1?#2}
1113   %\str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
1114   %\seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
1115     % \seq_new:c {g__stex_refs_labels_#2_seq}
1116     %}
1117   %\seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
1118     % \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
1119     %}
1120 }

```

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

```

1121 \AtEndDocument{
1122   \def\STEXInternalAuxAddDocRef#1 #2 {}{}
1123 }

```

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

```

1124 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
1125
1126   % \stex_if_smsmode:TF {
1127   %   \str_if_exist:cF{sref_sym_#1_type}{
1128   %     \stex_get_document_url:
1129   %     \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str
1130   %     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
1131   %   }
1132   % }{
1133   %   \str_if_empty:NF \l__stex_refs_curr_label_str {
1134   %     \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
1135   %     \immediate\write\@auxout{
1136   %       \exp_not:N\expandafter\def\exp_not:N\csname \exp_not:N\detokenize{sref_sym_#1_label}
1137   %       \l__stex_refs_curr_label_str
1138   %     }
1139   %   }
1140   % }
1141 % }
1142 }

```

(End definition for \stex\_ref\_new\_sym\_target:n. This function is documented on page 76.)

## 26.3 Using References

`\sref` Optional arguments:

```

1143
1144 \keys_define:nn { stex / sref / 1 } {
1145   archive .str_set_x:N = \l__stex_refs_repo_str,
1146   file     .str_set_x:N = \l__stex_refs_file_str,
1147   % TODO get rid of this
1148   fallback .code:n = {},
1149   pre      .code:n = {},
1150   post     .code:n = {}
1151 }
1152 \cs_new_protected:Nn \__stex_refs_args_i:n {
1153   \str_clear:N \l__stex_refs_repo_str
1154   \str_clear:N \l__stex_refs_file_str
1155   \keys_set:nn { stex / sref / 1 } { #1 }
1156 }
1157 \keys_define:nn { stex / sref / 2 } {
1158   in       .str_set_x:N = \l__stex_refs_in_str,
1159   archive  .str_set_x:N = \l__stex_refs_repob_str,
1160   title    .tl_set:N = \l__stex_refs_title_tl
1161 }
1162 \cs_new_protected:Nn \__stex_refs_args_ii:n {
1163   \str_clear:N \l__stex_refs_in_str
1164   \tl_clear:N \l__stex_refs_title_tl
1165   \str_clear:N \l__stex_refs_repob_str
1166   \keys_set:nn { stex / sref / 2 } { #1 }
1167 }

```

The actual macro:

```

1168 \NewDocumentCommand \sref { 0{} m 0{} }{
1169   \__stex_refs_args_i:n{#1}
1170   \__stex_refs_args_ii:n{#3}
1171   \str_clear:N \l__stex_refs_uri_str
1172   \__stex_refs_find_uri:n{#2}
1173   \__stex_refs_do_sref:n{#2}
1174 }
1175 \NewDocumentCommand \extref { 0{} m m }{
1176   \__stex_refs_args_i:n{#1}
1177   \__stex_refs_args_ii:n{#3}
1178   \str_if_empty:NT \l__stex_refs_in_str {
1179     \msg_error:nn{stex}{error/extrefmissing}
1180   }
1181   \str_clear:N \l__stex_refs_uri_str
1182   \__stex_refs_find_uri:n{#2}
1183   \__stex_refs_do_sref_in:n{#2}
1184 }
1185
1186 \cs_new_protected:Nn \__stex_refs_find_uri:n {
1187   \stex_debug:nn{sref}{File: \l__stex_refs_file_str^^JRepo:\l__stex_refs_repo_str}
1188   \str_if_empty:NTF \l__stex_refs_file_str {
1189     \seq_if_exist:cT{g_stex_ref_\l_stex_current_docns_str_seq}{
1190       \seq_map_inline:cn{g_stex_ref_\l_stex_current_docns_str_seq}{
1191         \str_if_eq:nnT{#1}{##1}{

```

```

1192         \str_set_eq:NN \l__stex_refs_uri_str \l_stex_current_docns_str
1193         \seq_map_break:
1194     }
1195 }
1196 }
1197 \str_if_empty:NF \l__stex_refs_uri_str {
1198     \seq_map_inline:Nn \g_stex_ref_files_seq {
1199         \seq_map_inline:cn{g_stex_ref_###1_seq}{
1200             \str_if_eq:nnT{#1}{###1}{
1201                 \str_set:Nn \l__stex_refs_uri_str {##1}
1202                 \seq_map_break:n{\seq_map_break:}
1203             }
1204         }
1205     }
1206 }
1207 }{
1208     \str_if_empty:NTF \l__stex_refs_repo_str {
1209         \prop_if_exist:NTF \l_stex_current_repository_prop {
1210             \prop_get:NnN \l_stex_current_repository_prop { ns } \l__stex_refs_uri_str
1211             \str_set:Nx \l__stex_refs_uri_str {\l__stex_refs_uri_str / \l__stex_refs_file_str}
1212             \stex_path_from_string:Nn \l_tmpb_seq \l__stex_refs_uri_str
1213             \str_set:Nx \l__stex_refs_uri_str {\stex_path_to_string:N \l_tmpb_seq}
1214         }{
1215             \stex_path_from_string:Nn \l_tmpb_seq {
1216                 \stex_path_to_string:N \g_stex_currentfile_seq/ .. / \l__stex_refs_file_str
1217             }
1218             \str_set:Nx \l__stex_refs_uri_str {file:/\stex_path_to_string:N \l_tmpb_seq}
1219         }
1220     }{
1221         \stex_require_repository:n \l__stex_refs_repo_str
1222         \prop_get:cnN { c_stex_mathhub_\l__stex_refs_repo_str _manifest_prop } { ns } \l__stex_refs_uri_str
1223         \str_set:Nx \l__stex_refs_uri_str {\l__stex_refs_uri_str / \l__stex_refs_file_str}
1224         \stex_path_from_string:Nn \l_tmpb_seq \l__stex_refs_uri_str
1225         \str_set:Nx \l__stex_refs_uri_str {\stex_path_to_string:N \l_tmpb_seq}
1226     }
1227 }
1228 }
1229
1230 \cs_new_protected:Nn \__stex_refs_do_autoref:n{
1231     \cs_if_exist:cTF{autoref}{
1232         \exp_args:Nx\autoref{sref_#1}
1233     }{
1234         \exp_args:Nx\ref{sref_#1}
1235     }
1236 }
1237
1238 \cs_new_protected:Nn \__stex_refs_do_sref:n {
1239     \str_if_empty:NTF \l__stex_refs_uri_str {
1240         \str_if_empty:NTF \l__stex_refs_in_str {
1241             \__stex_refs_do_autoref:n{#1}
1242         }{
1243             \__stex_refs_do_sref_in:n{#1}
1244         }
1245     }{

```



```

1246 \exp_args:NNo \seq_if_in:NnTF \g_stex_ref_files_seq \l__stex_refs_uri_str {
1247 \exp_args:Nnx \seq_if_in:cnTF{g_stex_ref_\l__stex_refs_uri_str_seq}{\detokenize{#1}}{
1248 \__stex_refs_do_autoref:n{\l__stex_refs_uri_str?#1}
1249 }{
1250 \str_if_empty:NTF \l__stex_refs_in_str {
1251 \__stex_refs_do_autoref:n{#1}
1252 }{
1253 \__stex_refs_do_sref_in:n{#1}
1254 }
1255 }
1256 }{
1257 \str_if_empty:NTF \l__stex_refs_in_str {
1258 \__stex_refs_do_autoref:n{#1}
1259 }{
1260 \__stex_refs_do_sref_in:n{#1}
1261 }
1262 }
1263 }
1264 }
1265
1266 \cs_new_protected:Nn \__stex_refs_restore_target:nnnnn {
1267 \str_if_empty:NTF \l__stex_refs_uri_str {
1268 \exp_args:No \str_if_eq:nnT \l__stex_refs_id_str {#2}{
1269 \tl_set:Nn \l__stex_refs_return_tl {
1270 \use:c{#3autorefname}~#4\tl_if_empty:nF{#5}{~(5)}~in~
1271 \tl_if_empty:nTF\l__stex_refs_title_tl{
1272 ???
1273 }\l__stex_refs_title_tl
1274 }
1275 }
1276 }{
1277 \stex_debug:nn{sref}{\l__stex_refs_uri_str}{~ == ~ #1 ~ ?}
1278 \exp_args:No \str_if_eq:nnT \l__stex_refs_uri_str {#1}{
1279 \stex_debug:nn{sref}{\l__stex_refs_id_str~ == ~ #2 ~ ?}
1280 \exp_args:No \str_if_eq:nnT \l__stex_refs_id_str {#2}{
1281 \stex_debug:nn{sref}{success!}
1282 \tl_set:Nn \l__stex_refs_return_tl {
1283 \use:c{#3autorefname}~#4\tl_if_empty:nF{#5}{~(5)}~in~
1284 \tl_if_empty:nTF\l__stex_refs_title_tl{
1285 ???
1286 }\l__stex_refs_title_tl
1287 }
1288 \endinput
1289 }
1290 }
1291 }
1292 }
1293
1294 \cs_new_protected:Nn \__stex_refs_do_sref_in:n {
1295 \stex_debug:nn{sref}{In: \l__stex_refs_in_str^^JRepo:\l__stex_refs_repo_str}
1296 \stex_debug:nn{sref}{URI: \l__stex_refs_uri_str?#1}
1297 %\msg_warning:nnn{stex}{warning/smsmissing}{<filename>}
1298 \begingroup\catcode13=9\relax\catcode10=9\relax
1299 \str_if_empty:NTF \l__stex_refs_repob_str {

```

```

1300 \prop_if_exist:NTF \l_stex_current_repository_prop {
1301   \str_set:Nx \l_tmpa_str {
1302     \c_stex_mathhub_str /
1303     \prop_item:Nn \l_stex_current_repository_prop { id }
1304     / source / \l__stex_refs_in_str .sref
1305   }
1306 }{
1307   \str_set:Nx \l_tmpa_str {
1308     \stex_path_to_string:N \g_stex_currentfile_seq/ .. / \l__stex_refs_in_str . sref
1309   }
1310 }
1311 }{
1312   \str_set:Nx \l_tmpa_str {
1313     \c_stex_mathhub_str / \l__stex_refs_repob_str
1314     / source / \l__stex_refs_in_str . sref
1315   }
1316 }
1317 \stex_path_from_string:Nn \l_tmpb_seq \l_tmpa_str
1318 \stex_path_to_string:NN \l_tmpb_seq \l_tmpa_str
1319 \stex_debug:nn{sref}{File: \l_tmpa_str}
1320 \exp_args:No \IfFileExists \l_tmpa_str {
1321   \tl_clear:N \l__stex_refs_return_tl
1322   \str_set:Nn \l__stex_refs_id_str {#1}
1323   \let\STEXInternalSrefRestoreTarget\__stex_refs_restore_target:nnnnn
1324   \use:c{@ @ input}{\l_tmpa_str}
1325   \exp_args:No \tl_if_empty:nTF \l__stex_refs_return_tl {
1326     \exp_args:Nnno \msg_warning:nnnn{stex}{warning/smslabelmissing}\l_tmpa_str{#1}
1327     \__stex_refs_do_autoref:n{
1328       \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
1329     }
1330   }{
1331     \l__stex_refs_return_tl
1332   }
1333 }{
1334   \exp_args:Nnno \msg_warning:nnnn{stex}{warning/smsmissing}\l_tmpa_str
1335   \__stex_refs_do_autoref:n{
1336     \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
1337   }
1338 }
1339 \endgroup
1340 }
1341
1342 % \__stex_refs_args:n { #1 }
1343 % \str_if_empty:NTF \l__stex_refs_indocument_str {
1344 %   \str_set:Nx \l_tmpa_str { #2 }
1345 %   \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
1346 %   \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
1347 %     \seq_if_exist:CTF{g__stex_refs_labels_\l_tmpa_str_seq}{
1348 %       \seq_get_left:cnF {g__stex_refs_labels_\l_tmpa_str_seq} \l_tmpa_str {
1349 %         \str_clear:N \l_tmpa_str
1350 %       }
1351 %     }{
1352 %       \str_clear:N \l_tmpa_str
1353 %     }

```

```

1354 %   }{
1355 %       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1356 %       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1357 %       \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
1358 %       \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str_seq}{
1359 %           \str_set_eq:NN \l_tmpc_str \l_tmpa_str
1360 %           \str_clear:N \l_tmpa_str
1361 %           \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str_seq} {
1362 %               \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
1363 %                   \str_range:nnn { ##1 }{-\l_tmpa_int}{ -1 }
1364 %               }{
1365 %                   \seq_map_break:n {
1366 %                       \str_set:Nn \l_tmpa_str { ##1 }
1367 %                   }
1368 %               }
1369 %           }
1370 %       }{
1371 %           \str_clear:N \l_tmpa_str
1372 %       }
1373 %   }
1374 %   \str_if_empty:NTF \l_tmpa_str {
1375 %       \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_li
1376 %   }{
1377 %       \str_if_eq:cNTF {sref_\l_tmpa_str_type} \c__stex_refs_ref_str {
1378 %           \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1379 %               \cs_if_exist:cTF{autoref}{
1380 %                   \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1381 %               }{
1382 %                   \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1383 %               }
1384 %           }{
1385 %               \ltx@ifpackageloaded{hyperref}{
1386 %                   \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
1387 %               }{
1388 %                   \l__stex_refs_linktext_tl
1389 %               }
1390 %           }
1391 %       }{
1392 %           \ltx@ifpackageloaded{hyperref}{
1393 %               \href{\use:c{sref_url_\l_tmpa_str_str}}{\tl_if_empty:NTF \l__stex_refs_linktext_
1394 %           }{
1395 %               \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_ref
1396 %           }
1397 %       }
1398 %   }
1399 %   }{
1400 %       % TODO
1401 %   }
1402 %}

```

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

**\srefsym**

```

1403 \NewDocumentCommand \srefsym { 0{} m}{

```

```

1404 \stex_get_symbol:n { #2 }
1405 \__stex_refs_sym_aux:nn{#1}{\l_stex_get_symbol_uri_str}
1406 }
1407
1408 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1409
1410 % \str_if_exist:cTF {sref_sym_#2 _label_str }{
1411 % \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1412 % }{
1413 % \__stex_refs_args:n { #1 }
1414 % \str_if_empty:NTF \l__stex_refs_indocument_str {
1415 % \tl_if_exist:cTF{sref_sym_#2 _type}{
1416 % % doc uri in \l_tmpb_str
1417 % \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1418 % \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1419 % % reference
1420 % \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1421 % \cs_if_exist:cTF{autoref}{
1422 % \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1423 % }{
1424 % \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1425 % }
1426 % }{
1427 % \ltx@ifpackageloaded{hyperref}{
1428 % \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1429 % }{
1430 % \l__stex_refs_linktext_tl
1431 % }
1432 % }
1433 % }{
1434 % % URL
1435 % \ltx@ifpackageloaded{hyperref}{
1436 % \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl
1437 % }{
1438 % \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_
1439 % }
1440 % }
1441 % }{
1442 % \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_
1443 % }
1444 % }{
1445 % % TODO
1446 % }
1447 % }
1448 }

```

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

`\srefsymuri`

```

1449 \cs_new_protected:Npn \srefsymuri #1 #2 { % TODO
1450 #2%\__stex_refs_sym_aux:nn[linktext={#2}]{#1}
1451 }

```

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

```

1452 </package>

```

## Chapter 27

# STEX -Modules Implementation

```
1453 <*package>
1454
1455 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1456
1457 <@@=stex_modules>
1458
1459     Warnings and error messages
1458 \msg_new:nnn{stex}{error/unknownmodule}{
1459     No~module~#1~found
1460 }
1461 \msg_new:nnn{stex}{error/syntax}{
1462     Syntax~error:~#1
1463 }
1464 \msg_new:nnn{stex}{error/siglanguage}{
1465     Module~#1~declares~signature~#2,~but~does~not~
1466     declare~its~language
1467 }
1468 \msg_new:nnn{stex}{warning/deprecated}{
1469     #1~is~deprecated;~please~use~#2~instead!
1470 }
1471
1472 \msg_new:nnn{stex}{error/conflictingmodules}{
1473     Conflicting~imports~for~module~#1
1474 }
1475
\l_stex_current_module_str The current module:
1475 \str_new:N \l_stex_current_module_str
1476
(End definition for \l_stex_current_module_str. This variable is documented on page 79.)

\l_stex_all_modules_seq Stores all available modules
1476 \seq_new:N \l_stex_all_modules_seq
1477
(End definition for \l_stex_all_modules_seq. This variable is documented on page 79.)
```

```

\stex_if_in_module_p:
\stex_if_in_module:TF
1477 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1478   \str_if_empty:NTF \l_stex_current_module_str
1479   \prg_return_false: \prg_return_true:
1480 }

(End definition for \stex_if_in_module:TF. This function is documented on page 79.)

```

```

\stex_if_module_exists_p:n
\stex_if_module_exists:nTF
1481 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1482   \prop_if_exist:cTF { c_stex_module_#1_prop }
1483   \prg_return_true: \prg_return_false:
1484 }

(End definition for \stex_if_module_exists:nTF. This function is documented on page 79.)

```

```

\stex_add_to_current_module:n
\STEXexport
Only allowed within modules:
1485 \cs_new_protected:Nn \stex_execute_in_module:n { \stex_if_in_module:T {
1486   \stex_add_to_current_module:n { #1 }
1487   \stex_do_up_to_module:n { #1 }
1488 }}
1489 \cs_generate_variant:Nn \stex_execute_in_module:n {x}
1490
1491 \cs_new_protected:Nn \stex_add_to_current_module:n {
1492   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str_code} { #1 }
1493 }
1494 \cs_generate_variant:Nn \stex_add_to_current_module:n {x}
1495 \cs_new_protected:Npn \STEXexport {
1496   \ExplSyntaxOn
1497   \__stex_modules_export:n
1498 }
1499 \cs_new_protected:Nn \__stex_modules_export:n {
1500   \ignorespacesandpars#1\ExplSyntaxOff
1501   \stex_add_to_current_module:n { \ignorespacesandpars#1}
1502   \stex_smsmode_do:
1503 }
1504 \let \stex_module_export_helper:n \use:n
1505 \stex_deactivate_macro:Nn \STEXexport {module~environments}

(End definition for \stex_add_to_current_module:n and \STEXexport. These functions are documented
on page 79.)

```

```

\stex_add_constant_to_current_module:n
1506 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1507   \str_set:Nx \l_tmpa_str { #1 }
1508   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str_constants} { \l_tmpa_str }
1509 }

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

```

```

\stex_add_import_to_current_module:n
1510 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1511   \str_set:Nx \l_tmpa_str { #1 }
1512   \exp_args:Nno

```

```

1513 \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1514 \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1515 }
1516 }

```

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

`\stex_collect_imports:n`

```

1517 \cs_new_protected:Nn \stex_collect_imports:n {
1518 \seq_clear:N \l_stex_collect_imports_seq
1519 \__stex_modules_collect_imports:n {#1}
1520 }
1521 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1522 \seq_map_inline:cn {c_stex_module_#1_imports} {
1523 \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1524 \__stex_modules_collect_imports:n { ##1 }
1525 }
1526 }
1527 \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1528 \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1529 }
1530 }

```

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

`\stex_do_up_to_module:n`

```

1531 \int_new:N \l__stex_modules_group_depth_int
1532 \cs_new_protected:Nn \stex_do_up_to_module:n {
1533 \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1534 #1
1535 }{
1536 #1
1537 \expandafter \tl_gset:Nn
1538 \csname l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1539 \expandafter\expandafter\expandafter\endcsname
1540 \expandafter\expandafter\expandafter { \csname
1541 l__stex_modules_aftergroup_\l_stex_current_module_str _tl\endcsname #1 }
1542 \aftergroup\__stex_modules_aftergroup_do:
1543 }
1544 }
1545 \cs_generate_variant:Nn \stex_do_up_to_module:n {x}
1546 \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1547 \stex_debug:nn{aftergroup}{\cs_meaning:c{
1548 l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1549 }}}
1550 \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1551 \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1552 \tl_gclear:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1553 }{
1554 \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1555 \aftergroup\__stex_modules_aftergroup_do:
1556 }
1557 }
1558 \cs_new_protected:Nn \stex_reset_up_to_module:n {
1559 \expandafter\let\csname l__stex_modules_aftergroup_#1_tl\endcsname\undefined

```

1560 }

(End definition for \stex\_do\_up\_to\_module:n. This function is documented on page 79.)

\stex\_modules\_compute\_namespace:nN Computes the appropriate namespace from the top-level namespace of a repository (#1) and a file path (#2).

1561

(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.

```
1562 \str_new:N \l_stex_module_ns_str
1563 \str_new:N \l_stex_module_subpath_str
1564 \cs_new_protected:Nn \__stex_modules_compute_namespace:nN {
1565   \seq_set_eq:NN \l_tmpa_seq #2
1566   % split off file extension
1567   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str % <- filename
1568   \exp_args:Nnno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1569   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str % <- filename without suffixes
1570   \seq_put_right:No \l_tmpa_seq \l_tmpb_str % <- file path including name without suffixes
1571
1572   \bool_set_true:N \l_tmpa_bool
1573   \bool_while_do:Nn \l_tmpa_bool {
1574     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1575     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1576       {source} { \bool_set_false:N \l_tmpa_bool }
1577     }{}{
1578       \seq_if_empty:NT \l_tmpa_seq {
1579         \bool_set_false:N \l_tmpa_bool
1580       }
1581     }
1582   }
1583
1584   \stex_path_to_string:NN \l_tmpa_seq \l_stex_module_subpath_str
1585   % \l_tmpa_seq <- sub-path relative to archive
1586   \str_if_empty:NTF \l_stex_module_subpath_str {
1587     \str_set:Nx \l_stex_module_ns_str {#1}
1588   }{
1589     \str_set:Nx \l_stex_module_ns_str {
1590       #1/\l_stex_module_subpath_str
1591     }
1592   }
1593 }
1594
1595 \cs_new_protected:Nn \stex_modules_current_namespace: {
1596   \str_clear:N \l_stex_module_subpath_str
1597   \prop_if_exist:NTF \l_stex_current_repository_prop {
1598     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1599     \__stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1600   }{
1601     % split off file extension
1602     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1603     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
```



```

1604 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1605 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1606 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1607 \str_set:Nx \l_stex_module_ns_str {
1608   file:/\stex_path_to_string:N \l_tmpa_seq
1609 }
1610 }
1611 }

```

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

## 27.1 The smodule environment

smodule arguments:

```

1612 \keys_define:nn { stex / module } {
1613   title      .tl_set:N      = \smodulename ,
1614   type       .str_set_x:N   = \smodulename ,
1615   id         .str_set_x:N   = \smoduleid ,
1616   deprecate  .str_set_x:N   = \l_stex_module_deprecate_str ,
1617   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1618   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1619   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1620   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1621   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1622   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1623   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1624 }
1625
1626 \cs_new_protected:Nn \__stex_modules_args:n {
1627   \str_clear:N \smodulename
1628   \str_clear:N \smodulename
1629   \str_clear:N \smoduleid
1630   \str_clear:N \l_stex_module_ns_str
1631   \str_clear:N \l_stex_module_deprecate_str
1632   \str_clear:N \l_stex_module_lang_str
1633   \str_clear:N \l_stex_module_sig_str
1634   \str_clear:N \l_stex_module_creators_str
1635   \str_clear:N \l_stex_module_contributors_str
1636   \str_clear:N \l_stex_module_meta_str
1637   \str_clear:N \l_stex_module_srccite_str
1638   \keys_set:nn { stex / module } { #1 }
1639 }
1640
1641 % module parameters here? In the body?
1642

```

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

```

1643 \cs_new_protected:Nn \stex_module_setup:nn {
1644   \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1645   \str_set:Nx \l_stex_module_name_str { #2 }
1646   \__stex_modules_args:n { #1 }

```

First, we set up the name and namespace of the module.

Are we in a nested module?

```

1647 \stex_if_in_module:TF {
1648   % Nested module
1649   \prop_get:cnN {c_stex_module_\l_stex_current_module_str _prop}
1650   { ns } \l_stex_module_ns_str
1651   \str_set:Nx \l_stex_module_name_str {
1652     \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
1653     { name } / \l_stex_module_name_str
1654   }
1655   \str_if_empty:NT \l_stex_module_lang_str {
1656     \str_set:Nx \l_stex_module_lang_str {
1657       \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
1658       { lang }
1659     }
1660   }
1661 }{
1662   % not nested:
1663   \str_if_empty:NT \l_stex_module_ns_str {
1664     \stex_modules_current_namespace:
1665     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1666       / {\l_stex_module_ns_str}
1667     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1668     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1669       \str_set:Nx \l_stex_module_ns_str {
1670         \stex_path_to_string:N \l_tmpa_seq
1671       }
1672     }
1673   }
1674 }

```

Next, we determine the language of the module:

```

1675 \str_if_empty:NT \l_stex_module_lang_str {
1676   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1677   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1678   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1679   \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
1680     \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
1681       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
1682     }
1683   }
1684   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1685   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
1686     \seq_pop_right:NN \l_tmpa_seq \l_stex_module_lang_str
1687     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1688       inferred~from~file~name}
1689   }
1690 }
1691
1692 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
1693   \exp_args:NNo \stex_set_language:Nn \l_tmpa_str \l_stex_module_lang_str
1694 }}

```

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

```

1695 \str_if_empty:NTF \l_stex_module_sig_str {
1696   \exp_args:Nnx \prop_gset_from_keyval:cn {
1697     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1698   } {
1699     name      = \l_stex_module_name_str ,
1700     ns        = \l_stex_module_ns_str ,
1701     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1702     lang      = \l_stex_module_lang_str ,
1703     sig       = \l_stex_module_sig_str ,
1704     deprecate = \l_stex_module_deprecate_str ,
1705     meta      = \l_stex_module_meta_str
1706   }
1707   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1708   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1709   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _copymodules}
1710   \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1711   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1712 \str_if_empty:NT \l_stex_module_meta_str {
1713   \str_set:Nx \l_stex_module_meta_str {
1714     \c_stex_metatheory_ns_str ? Metatheory
1715   }
1716 }
1717 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1718   \bool_set_true:N \l_stex_in_meta_bool
1719   \exp_args:Nx \stex_add_to_current_module:n {
1720     \bool_set_true:N \l_stex_in_meta_bool
1721     \stex_activate_module:n {\l_stex_module_meta_str}
1722     \bool_set_false:N \l_stex_in_meta_bool
1723   }
1724   \stex_activate_module:n {\l_stex_module_meta_str}
1725   \bool_set_false:N \l_stex_in_meta_bool
1726 }
1727 }{
1728   \str_if_empty:NT \l_stex_module_lang_str {
1729     \msg_error:nnxx{stex}{error/siglanguage}{
1730       \l_stex_module_ns_str?\l_stex_module_name_str
1731     }{\l_stex_module_sig_str}
1732   }
1733   \stex_debug:nn{modules}{Signature~\l_stex_module_sig_str~for~\l_stex_module_ns_str?\l_st
1734   \stex_if_module_exists:nTF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1735     \stex_debug:nn{modules}{(already exists)}
1736   }{
1737     \stex_debug:nn{modules}{(needs loading)}
1738     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1739     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1740     \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1741     \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1742     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1743     \str_set:Nx \l_tmpa_str {
1744       \stex_path_to_string:N \l_tmpa_seq /

```

```

1745     \l_tmpa_str . \l_stex_module_sig_str .tex
1746   }
1747   \IfFileExists \l_tmpa_str {
1748     \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1749       \str_clear:N \l_stex_current_module_str
1750       \seq_clear:N \l_stex_all_modules_seq
1751       \stex_debug:nn{modules}{Loading~signature}
1752     }
1753   }{
1754     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1755   }
1756 }
1757 \stex_if_smsmode:F {
1758   \stex_activate_module:n {
1759     \l_stex_module_ns_str ? \l_stex_module_name_str
1760   }
1761 }
1762 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1763 }
1764 \str_if_empty:NF \l_stex_module_deprecate_str {
1765   \msg_warning:nnxx{stex}{warning/deprecated}{
1766     Module~\l_stex_current_module_str
1767   }{
1768     \l_stex_module_deprecate_str
1769   }
1770 }
1771 \seq_put_right:Nx \l_stex_all_modules_seq {
1772   \l_stex_module_ns_str ? \l_stex_module_name_str
1773 }
1774 \tl_clear:c{l__stex_modules_aftergroup_\l_stex_module_ns_str ? \l_stex_module_name_str _tl}
1775 }

```

(End definition for `\stex_module_setup:nn`. This function is documented on page 80.)

**smodule** (*env.*) The module environment.

```

\__stex_modules_begin_module: implements \begin{smodule}
1776 \cs_new_protected:Nn \__stex_modules_begin_module: {
1777   \stex_reactivate_macro:N \STEXexport
1778   \stex_reactivate_macro:N \importmodule
1779   \stex_reactivate_macro:N \symdecl
1780   \stex_reactivate_macro:N \notation
1781   \stex_reactivate_macro:N \symdef
1782 }
1783 \stex_debug:nn{modules}{
1784   New~module:\\
1785   Namespace:~\l_stex_module_ns_str\\
1786   Name:~\l_stex_module_name_str\\
1787   Language:~\l_stex_module_lang_str\\
1788   Signature:~\l_stex_module_sig_str\\
1789   Metatheory:~\l_stex_module_meta_str\\
1790   File:~\stex_path_to_string:N \g_stex_currentfile_seq
1791 }
1792

```

```

1793 \stex_if_do_html:T{
1794   \begin{stex_annotate_env} {theory} {
1795     \l_stex_module_ns_str ? \l_stex_module_name_str
1796   }
1797
1798   \stex_annotate_invisible:nnn{header}{} {
1799     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1800     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1801     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1802       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1803     }
1804     \str_if_empty:NF \smoduletype {
1805       \stex_annotate:nnn{type}{\smoduletype}{}
1806     }
1807   }
1808 }
1809 % TODO: Inherit metatheory for nested modules?
1810 }
1811 \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}`

```

1812 \cs_new_protected:Nn \_stex_modules_end_module: {
1813   \stex_debug:nn{modules}{Closing module~\prop_item:cn {c_stex_module\_l_stex_current_module}
1814   \stex_reset_up_to_module:n \l_stex_current_module_str
1815   \stex_if_smsmode:T {
1816     \stex_persist:x {
1817       \prop_set_from_keyval:cn{c_stex_module\_l_stex_current_module_str _prop}{
1818         \exp_after:wN \prop_to_keyval:N \csname c_stex_module\_l_stex_current_module_str _pr
1819       }
1820       \seq_set_from_clist:cn{c_stex_module\_l_stex_current_module_str _constants}{
1821         \seq_use:cn{c_stex_module\_l_stex_current_module_str _constants},
1822       }
1823       \seq_set_from_clist:cn{c_stex_module\_l_stex_current_module_str _imports}{
1824         \seq_use:cn{c_stex_module\_l_stex_current_module_str _imports},
1825       }
1826       \tl_set:cn {c_stex_module\_l_stex_current_module_str _code}
1827     }
1828     \exp_after:wN \let \exp_after:wN \l_tmpa_tl \csname c_stex_module\_l_stex_current_module
1829     \exp_after:wN \stex_persist:n \exp_after:wN { \exp_after:wN { \l_tmpa_tl } }
1830   }
1831 }

```

(End definition for `\_stex_modules_end_module:.`)

The core environment

```

1832 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1833 \NewDocumentEnvironment { smodule } { 0 } { m } {
1834   \stex_module_setup:nn{#1}{#2}
1835   %\par
1836   \stex_if_smsmode:F{
1837     \tl_if_empty:NF \smoduletitle {
1838       \exp_args:No \stex_document_title:n \smoduletitle
1839     }

```

```

1840 \tl_clear:N \l_tmpa_tl
1841 \clist_map_inline:Nn \smodulotype {
1842   \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1843     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1844   }
1845 }
1846 \tl_if_empty:NTF \l_tmpa_tl {
1847   \__stex_modules_smodule_start:
1848 }{
1849   \l_tmpa_tl
1850 }
1851 }
1852 \__stex_modules_begin_module:
1853 \str_if_empty:NF \smoduleid {
1854   \stex_ref_new_doc_target:n \smoduleid
1855 }
1856 \stex_smsmode_do:
1857 } {
1858   \__stex_modules_end_module:
1859   \stex_if_smsmode:F {
1860     \end{stex_annotate_env}
1861     \clist_set:Nn \l_tmpa_clist \smodulotype
1862     \tl_clear:N \l_tmpa_tl
1863     \clist_map_inline:Nn \l_tmpa_clist {
1864       \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1865         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1866       }
1867     }
1868     \tl_if_empty:NTF \l_tmpa_tl {
1869       \__stex_modules_smodule_end:
1870     }{
1871       \l_tmpa_tl
1872     }
1873   }
1874 }

```

### **\stexpatchmodule**

```

1875 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1876 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1877
1878 \newcommand\stexpatchmodule[3] [] {
1879   \str_set:Nx \l_tmpa_str{ #1 }
1880   \str_if_empty:NTF \l_tmpa_str {
1881     \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1882     \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1883   }{
1884     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1885     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1886   }
1887 }

```

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

## 27.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n 1888 \NewDocumentCommand \STEXModule { m } {
1889   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1890   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1891   \tl_set:Nn \l_tmpa_tl {
1892     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1893   }
1894   \seq_map_inline:Nn \l_stex_all_modules_seq {
1895     \str_set:Nn \l_tmpb_str { ##1 }
1896     \str_if_eq:eeT { \l_tmpa_str } {
1897       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1898     } {
1899       \seq_map_break:n {
1900         \tl_set:Nn \l_tmpa_tl {
1901           \stex_invoke_module:n { ##1 }
1902         }
1903       }
1904     }
1905   }
1906   \l_tmpa_tl
1907 }
1908
1909 \cs_new_protected:Nn \stex_invoke_module:n {
1910   \stex_debug:nn{modules}{Invoking~module~#1}
1911   \peek_charcode_remove:NTF ! {
1912     \__stex_modules_invoke_uri:nN { #1 }
1913   } {
1914     \peek_charcode_remove:NTF ? {
1915       \__stex_modules_invoke_symbol:nn { #1 }
1916     } {
1917       \msg_error:nnx{stex}{error/syntax}{
1918         ?~or~!~expected~after~
1919         \c_backslash_str STEXModule{#1}
1920       }
1921     }
1922   }
1923 }
1924
1925 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1926   \str_set:Nn #2 { #1 }
1927 }
1928
1929 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1930   \stex_invoke_symbol:n{#1?#2}
1931 }

```

(End definition for `\STEXModule` and `\stex_invoke_module:n`. These functions are documented on page 80.)

```

\stex_activate_module:n
1932 \bool_new:N \l_stex_in_meta_bool
1933 \bool_set_false:N \l_stex_in_meta_bool

```

```

1934 \cs_new_protected:Nn \stex_activate_module:n {
1935   \stex_debug:nn{modules}{Activating~module~#1}
1936   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1937     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1938     \use:c{ c_stex_module_#1_code }
1939   }
1940 }

```

*(End definition for \stex\_activate\_module:n. This function is documented on page 81.)*

```

1941 \</package>

```



## Chapter 28

# STEX -Module Inheritance Implementation

```
1942 <*package>
1943
1944 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1945
```

### 28.1 SMS Mode

```
1946 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1947 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1948 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1949 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1950
1951 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1952   \makeatletter
1953   \makeatother
1954   \ExplSyntaxOn
1955   \ExplSyntaxOff
1956   \rustexBREAK
1957 }
1958
1959 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1960   \symdef
1961   \importmodule
1962   \notation
1963   \symdecl
1964   \STEXexport
1965   \inlineass
1966   \inlinedef
1967   \inlineex
1968   \endinput
1969   \setnotation
```

```

1970 \copynotation
1971 \assign
1972 \renamedekl
1973 \donotcopy
1974 \instantiate
1975 \textsymdecl
1976 }
1977
1978 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1979   \tl_to_str:n {
1980     smodule,
1981     copymodule,
1982     interpretmodule,
1983     realization,
1984     sdefinition,
1985     sexample,
1986     sassertion,
1987     sparagraph,
1988     mathstructure
1989   }
1990 }

```

(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 82.)

`\stex_if_smsmode_p:`

```

\stex_if_smsmode:TF
1991 \bool_new:N \g__stex_smsmode_bool
1992 \bool_set_false:N \g__stex_smsmode_bool
1993 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1994   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1995 }

```

(End definition for `\stex_if_smsmode:TF`. This function is documented on page 82.)

`\_stex_smsmode_in_smsmode:nn`

```

1996 \cs_new_protected:Nn \_stex_smsmode_in_smsmode:nn { \stex_suppress_html:n {
1997   \vbox_set:Nn \l_tmpa_box {
1998     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1999     \bool_gset_true:N \g__stex_smsmode_bool
2000     #2
2001     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
2002   }
2003   \box_clear:N \l_tmpa_box
2004 } }

```

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

`\stex_file_in_smsmode:nn`

```

2005 \quark_new:N \q__stex_smsmode_break
2006
2007 \NewDocumentCommand \_stex_smsmode_importmodule: { 0{} m } {
2008   \seq_gput_right:Nn \l__stex_smsmode_importmodules_seq {{#1}{#2}}
2009   \stex_smsmode_do:
2010 }
2011

```

```

2012 \cs_new_protected:Nn \__stex_smsmode_module:nn {
2013   \__stex_modules_args:n{#1}
2014   \stex_if_in_module:F {
2015     \str_if_empty:NF \l_stex_module_sig_str {
2016       \stex_modules_current_namespace:
2017       \str_set:Nx \l_stex_module_name_str { #2 }
2018       \stex_if_module_exists:nF{\l_stex_module_ns_str?\l_stex_module_name_str}{
2019         \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
2020         \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
2021         \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
2022         \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
2023         \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
2024         \str_set:Nx \l_tmpa_str {
2025           \stex_path_to_string:N \l_tmpa_seq /
2026           \l_tmpa_str . \l_stex_module_sig_str .tex
2027         }
2028         \IfFileExists \l_tmpa_str {
2029           \exp_args:NNx \seq_gput_right:Nn \l__stex_smsmode_sigmodules_seq \l_tmpa_str
2030         }{
2031           \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
2032         }
2033       }
2034     }
2035   }
2036 }
2037
2038 \prg_new_conditional:Nnn \__stex_smsmode_check_import_pair:nn {T,F,TF} {
2039   %\stex_debug:nn{import-pair}{\detokenize{#1}~{#2}}
2040   \tl_if_empty:nTF{#1}{
2041     \prop_if_exist:NTF \l_stex_current_repository_prop
2042     {
2043       %\stex_debug:nn{import-pair}{in repository \prop_item:Nn \l_stex_current_repository_
2044       \prg_return_true:
2045     } {
2046       \seq_set_split:Nnn \l_tmpa_seq ? {#2}
2047       \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
2048       \tl_if_empty:NT \l_tmpa_tl {
2049         \seq_pop_left:NN \l_tmpa_seq \l_tmpa_tl
2050       }
2051       %\stex_debug:nn{import-pair}{\seq_use:Nn \l_tmpa_seq,~of~length~\seq_count:N \l_tmpa
2052       \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1
2053       \prg_return_true: \prg_return_false:
2054     }
2055   }\prg_return_true:
2056 }
2057
2058 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
2059   \stex_filestack_push:n{#1}
2060   \seq_gclear:N \l__stex_smsmode_importmodules_seq
2061   \seq_gclear:N \l__stex_smsmode_sigmodules_seq
2062   % ----- new -----
2063   \__stex_smsmode_in_smsmode:nn{#1}{
2064     \let\importmodule\__stex_smsmode_importmodule:
2065     \let\stex_module_setup:nn\__stex_smsmode_module:nn

```

```

2066 \let\__stex_modules_begin_module:\relax
2067 \let\__stex_modules_end_module:\relax
2068 \seq_clear:N \g_stex_smsmode_allowedenvs_seq
2069 \exp_args:NNx \seq_put_right:Nn \g_stex_smsmode_allowedenvs_seq {\tl_to_str:n{smodule}}
2070 \tl_clear:N \g_stex_smsmode_allowedmacros_tl
2071 \tl_clear:N \g_stex_smsmode_allowedmacros_escape_tl
2072 \tl_put_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\importmodule}
2073 \everyeof{\q__stex_smsmode_break\noexpand}
2074 \expandafter\expandafter\expandafter
2075 \stex_smsmode_do:
2076 \csname @ @ input\endcsname "#1"\relax
2077
2078 \seq_map_inline:Nn \l__stex_smsmode_sigmodules_seq {
2079   \stex_filestack_push:n{##1}
2080   \expandafter\expandafter\expandafter
2081   \stex_smsmode_do:
2082   \csname @ @ input\endcsname "##1"\relax
2083   \stex_filestack_pop:
2084 }
2085 }
2086 % ----- new -----
2087 \__stex_smsmode_in_smsmode:nn{#1} {
2088   #2
2089   % ----- new -----
2090   \begingroup
2091   %\stex_debug:nn{smsmode}{Here:~\seq_use:Nn\l__stex_smsmode_importmodules_seq, }
2092   \seq_map_inline:Nn \l__stex_smsmode_importmodules_seq {
2093     \__stex_smsmode_check_import_pair:nnT ##1 { \begingroup
2094       \stex_import_module_uri:nn ##1
2095       \stex_import_require_module:nnnn
2096       \l_stex_import_ns_str
2097       \l_stex_import_archive_str
2098       \l_stex_import_path_str
2099       \l_stex_import_name_str \endgroup
2100     }
2101   }
2102   \endgroup
2103   \stex_debug:nn{smsmode}{Actually~loading~file~#1}
2104   % ----- new -----
2105   \everyeof{\q__stex_smsmode_break\noexpand}
2106   \expandafter\expandafter\expandafter
2107   \stex_smsmode_do:
2108   \csname @ @ input\endcsname "#1"\relax
2109 }
2110 \stex_filestack_pop:
2111 }

```

(End definition for `\stex_file_in_smsmode:nn`. This function is documented on page 83.)

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

```

2112 \cs_new_protected:Npn \stex_smsmode_do: {
2113   \stex_if_smsmode:T {
2114     \__stex_smsmode_do:w

```

```

2115 }
2116 }
2117 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
2118   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
2119     \expandafter\if\expandafter\relax\noexpand#1
2120     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
2121     \else\expandafter\__stex_smsmode_do:w\fi
2122   }{
2123     \__stex_smsmode_do:w % #1
2124   }
2125 }
2126 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
2127   \cs_if_eq:NNTF #1 \q__stex_smsmode_break {
2128     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
2129       #1\__stex_smsmode_do:w
2130     }{
2131       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
2132         #1
2133       }{
2134         \cs_if_eq:NNTF \begin #1 {
2135           \__stex_smsmode_check_begin:n
2136         }{
2137           \cs_if_eq:NNTF \end #1 {
2138             \__stex_smsmode_check_end:n
2139           }{
2140             \__stex_smsmode_do:w
2141           }
2142         }
2143       }
2144     }
2145   }
2146 }
2147
2148 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
2149   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
2150     \begin{#1}
2151   }{
2152     \__stex_smsmode_do:w
2153   }
2154 }
2155 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
2156   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
2157     \end{#1}\__stex_smsmode_do:w
2158   }{
2159     \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
2160   }
2161 }

```

(End definition for `\stex_smsmode_do:.` This function is documented on page 83.)

## 28.2 Inheritance

```

2162 <@@=stex_importmodule>

```

`\stex_import_module_uri:nn`

```
2163 \cs_new_protected:Nn \stex_import_module_uri:nn {
2164   \str_set:Nx \l_stex_import_archive_str { #1 }
2165   \str_set:Nn \l_stex_import_path_str { #2 }
2166
2167   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
2168   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
2169   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
2170
2171   \stex_modules_current_namespace:
2172   \bool_lazy_all:nTF {
2173     {\str_if_empty_p:N \l_stex_import_archive_str}
2174     {\str_if_empty_p:N \l_stex_import_path_str}
2175     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
2176   }{
2177     \str_set_eq:NN \l_stex_import_path_str \l_stex_module_subpath_str
2178     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
2179   }{
2180     \str_if_empty:NT \l_stex_import_archive_str {
2181       \prop_if_exist:NT \l_stex_current_repository_prop {
2182         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
2183       }
2184     }
2185     \str_if_empty:NTF \l_stex_import_archive_str {
2186       \str_if_empty:NF \l_stex_import_path_str {
2187         \stex_path_from_string:Nn \l_tmpb_seq {
2188           \l_stex_module_ns_str / .. / \l_stex_import_path_str
2189         }
2190         \str_set:Nx \l_stex_import_ns_str {\stex_path_to_string:N \l_tmpb_seq}
2191         \str_replace_once:Nnn \l_stex_import_ns_str {file://} {file:///}
2192       }
2193     }{
2194       \stex_require_repository:n \l_stex_import_archive_str
2195       \prop_get:cnN { c_stex_mathhub_\l_stex_import_archive_str_manifest_prop } { ns }
2196       \l_stex_import_ns_str
2197       \str_if_empty:NF \l_stex_import_path_str {
2198         \str_set:Nx \l_stex_import_ns_str {
2199           \l_stex_import_ns_str / \l_stex_import_path_str
2200         }
2201       }
2202     }
2203   }
2204 }
```

(End definition for `\stex_import_module_uri:nn`. This function is documented on page 84.)

`\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`.

```
2205 \str_new:N \l_stex_import_name_str
2206 \str_new:N \l_stex_import_archive_str
2207 \str_new:N \l_stex_import_path_str
2208 \str_new:N \l_stex_import_ns_str
```

(End definition for `\l_stex_import_name_str` and others. These variables are documented on page 84.)

```

\stex_import_require_module:nnnnn {{\ns}} {{\archive-ID}} {{\path}} {{\name}}
2209 \cs_new_protected:Nn \stex_import_require_module:nnnnn {
2210 \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
2211
2212 \stex_debug:nn{requiremodule}{Here:\~1:\~2:\~3:\~4}
2213
2214 \exp_args:NNxx \seq_set_split:Nnn \l_tmpa_seq {\tl_to_str:n{/}} {#4}
2215 \seq_get_left:NN \l_tmpa_seq \l_tmpc_str
2216
2217 %\stex_debug:nn{requiremodule}{Top~module:\l_tmpc_str}
2218
2219 % archive
2220 \str_set:Nx \l_tmpa_str { #2 }
2221 \str_if_empty:NTF \l_tmpa_str {
2222 \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
2223 \seq_put_right:Nn \l_tmpa_seq {...}
2224 } {
2225 \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
2226 \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
2227 \seq_put_right:Nn \l_tmpa_seq { source }
2228 }
2229
2230 % path
2231 \str_set:Nx \l_tmpb_str { #3 }
2232 \str_if_empty:NTF \l_tmpb_str {
2233 \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / \l_tmpc_str }
2234
2235 \ltx@ifpackageloaded{babel} {
2236 \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
2237 { \language } \l_tmpb_str {
2238 \msg_error:nnx{stex}{error/unknownlanguage}{\language}
2239 }
2240 } {
2241 \str_clear:N \l_tmpb_str
2242 }
2243
2244 \stex_debug:nn{modules}{Checking~a1~\l_tmpa_str.\l_tmpb_str.tex}
2245 \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2246 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2247 }{
2248 \stex_debug:nn{modules}{Checking~a2~\l_tmpa_str.tex}
2249 \IfFileExists{ \l_tmpa_str.tex }{
2250 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2251 }{
2252 % try english as default
2253 \stex_debug:nn{modules}{Checking~a3~\l_tmpa_str.en.tex}
2254 \IfFileExists{ \l_tmpa_str.en.tex }{
2255 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2256 }{
2257 \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2258 }
2259 }
2260 }
2261

```

```

2262 } {
2263   \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
2264   \seq_concat:NNN \l_tmpb_seq \l_tmpa_seq \l_tmpb_seq
2265
2266   \ltx@ifpackageloaded{babel} {
2267     \exp_args:NnX \prop_get:NnNF \c_stex_language_abbrevs_prop
2268       { \language } \l_tmpb_str {
2269       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
2270     }
2271   } {
2272     \str_clear:N \l_tmpb_str
2273   }
2274
2275   \stex_path_canonicalize:N \l_tmpb_seq
2276   \stex_path_to_string:NN \l_tmpb_seq \l_tmpa_str
2277
2278   \stex_debug:nn{modules}{Checking~b1~\l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex}
2279   \IfFileExists{ \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }{
2280     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }
2281   }{
2282     \stex_debug:nn{modules}{Checking~b2~\l_tmpa_str/\l_tmpc_str.tex}
2283     \IfFileExists{ \l_tmpa_str/\l_tmpc_str.tex }{
2284       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.tex }
2285     }{
2286       % try english as default
2287       \stex_debug:nn{modules}{Checking~b3~\l_tmpa_str/\l_tmpc_str.en.tex}
2288       \IfFileExists{ \l_tmpa_str/\l_tmpc_str.en.tex }{
2289         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.en.tex }
2290       }{
2291         \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.\l_tmpb_str.tex}
2292         \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2293           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2294         }{
2295           \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.tex}
2296           \IfFileExists{ \l_tmpa_str.tex }{
2297             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2298           }{
2299             % try english as default
2300             \stex_debug:nn{modules}{Checking~b5~\l_tmpa_str.en.tex}
2301             \IfFileExists{ \l_tmpa_str.en.tex }{
2302               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2303             }{
2304               \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2305             }
2306           }
2307         }
2308       }
2309     }
2310   }
2311 }
2312
2313 \str_if_eq:eeF{\g__stex_importmodule_file_str}{\seq_use:Nn \g_stex_currentfile_seq /}{
2314   \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
2315     \seq_clear:N \l_stex_all_modules_seq

```



```

2316     \str_clear:N \l_stex_current_module_str
2317     \str_set:Nx \l_tmpb_str { #2 }
2318     \str_if_empty:NF \l_tmpb_str {
2319         \stex_set_current_repository:n { #2 }
2320     }
2321     \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
2322 }
2323
2324 \stex_if_module_exists:nF { #1 ? #4 } {
2325     \msg_error:nnx{stex}{error/unknownmodule}{
2326         #1?#4~(in~file~\g__stex_importmodule_file_str)
2327     }
2328 }
2329 }
2330
2331 }
2332 \stex_activate_module:n { #1 ? #4 }
2333 }

```

(End definition for `\stex_import_require_module:nnnn`. This function is documented on page 84.)

## `\importmodule`

```

2334 \NewDocumentCommand \importmodule { 0{} m } {
2335     \stex_import_module_uri:nn { #1 } { #2 }
2336     \stex_debug:nn{modules}{Importing~module:~
2337         \l_stex_import_ns_str ? \l_stex_import_name_str
2338     }
2339     \stex_import_require_module:nnnn
2340     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2341     { \l_stex_import_path_str } { \l_stex_import_name_str }
2342     \stex_if_smsmode:F {
2343         \stex_annotate_invisible:nnn
2344         {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
2345     }
2346     \exp_args:Nx \stex_add_to_current_module:n {
2347         \stex_import_require_module:nnnn
2348         { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2349         { \l_stex_import_path_str } { \l_stex_import_name_str }
2350     }
2351     \exp_args:Nx \stex_add_import_to_current_module:n {
2352         \l_stex_import_ns_str ? \l_stex_import_name_str
2353     }
2354     \stex_smsmode_do:
2355     \ignorespacesandpars
2356 }
2357 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

(End definition for `\importmodule`. This function is documented on page 83.)

## `\usemodule`

```

2358 \NewDocumentCommand \usemodule { 0{} m } {
2359     \stex_if_smsmode:F {
2360         \stex_import_module_uri:nn { #1 } { #2 }
2361         \stex_import_require_module:nnnn
2362         { \l_stex_import_ns_str } { \l_stex_import_archive_str }

```

```

2363 { \l_stex_import_path_str } { \l_stex_import_name_str }
2364 \stex_annotate_invisible:nnn
2365 {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
2366 }
2367 \stex_smsmode_do:
2368 \ignorespacesandpars
2369 }

```

(End definition for \usemodule. This function is documented on page 83.)

```

2370 \cs_new_protected:Nn \stex_csl_to_imports:Nn {
2371   \tl_if_empty:nF{#2}{
2372     \clist_set:Nn \l_tmpa_clist {#2}
2373     \clist_map_inline:Nn \l_tmpa_clist {
2374       \tl_if_head_eq_charcode:nNTF {##1} [{
2375         #1 ##1
2376       }{
2377         #1{##1}
2378       }
2379     }
2380   }
2381 }
2382 \cs_generate_variant:Nn \stex_csl_to_imports:Nn {No}
2383
2384
2385 \end{package}

```

## Chapter 29

# STEX -Symbols Implementation

```
2386 <*package>
2387
2388 %%%%%%%%%% symbols.dtx %%%%%%%%%%
2389
2390 \msg_new:nnn{stex}{error/wrongargs}{
2391   args~value~in~symbol~declaration~for~#1~
2392   needs~to~be~i,~a,~b~or~B,~but~#2~given
2393 }
2394 \msg_new:nnn{stex}{error/unknownsymbol}{
2395   No~symbol~#1~found!
2396 }
2397 \msg_new:nnn{stex}{error/seqlength}{
2398   Expected~#1~arguments;~got~#2!
2399 }
2400 \msg_new:nnn{stex}{error/unknownnotation}{
2401   Unknown~notation~#1~for~#2!
2402 }
```

### 29.1 Symbol Declarations

```
2403 <@@=stex_symdecl>
\stex_all_symbols:n Map over all available symbols
2404 \cs_new_protected:Nn \stex_all_symbols:n {
2405   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
2406   \seq_map_inline:Nn \l_stex_all_modules_seq {
2407     \seq_map_inline:cn{c_stex_module_##1_constants}{
2408       \__stex_symdecl_all_symbols_cs{##1?####1}
2409     }
2410   }
2411 }
```

(End definition for `\stex_all_symbols:n`. This function is documented on page 86.)

## **\STEXsymbol**

```
2412 \NewDocumentCommand \STEXsymbol { m } {  
2413   \stex_get_symbol:n { #1 }  
2414   \exp_args:No  
2415   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }  
2416 }
```

(End definition for \STEXsymbol. This function is documented on page 87.)

symdecl arguments:

```
2417 \keys_define:nn { stex / symdecl } {  
2418   name      .str_set_x:N = \l_stex_symdecl_name_str ,  
2419   local     .bool_set:N = \l_stex_symdecl_local_bool ,  
2420   args      .str_set_x:N = \l_stex_symdecl_args_str ,  
2421   type      .tl_set:N = \l_stex_symdecl_type_tl ,  
2422   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,  
2423   align     .str_set:N = \l_stex_symdecl_align_str , % TODO(?)  
2424   gfc       .str_set:N = \l_stex_symdecl_gfc_str , % TODO(?)  
2425   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)  
2426   def       .tl_set:N = \l_stex_symdecl_definiens_tl ,  
2427   reorder   .str_set_x:N = \l_stex_symdecl_reorder_str ,  
2428   argnames  .clist_set:N = \l_stex_symdecl_argnames_clist ,  
2429   assoc     .choices:nn =  
2430     {bin,binl,binr,pre,conj,pwconj}  
2431     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}  
2432 }  
2433  
2434 \bool_new:N \l_stex_symdecl_make_macro_bool  
2435  
2436 \cs_new_protected:Nn \__stex_symdecl_args:n {  
2437   \str_clear:N \l_stex_symdecl_name_str  
2438   \str_clear:N \l_stex_symdecl_args_str  
2439   \str_clear:N \l_stex_symdecl_deprecate_str  
2440   \str_clear:N \l_stex_symdecl_reorder_str  
2441   \str_clear:N \l_stex_symdecl_assoctype_str  
2442   \bool_set_false:N \l_stex_symdecl_local_bool  
2443   \tl_clear:N \l_stex_symdecl_type_tl  
2444   \tl_clear:N \l_stex_symdecl_definiens_tl  
2445   \clist_clear:N \l_stex_symdecl_argnames_clist  
2446  
2447   \keys_set:nn { stex / symdecl } { #1 }  
2448 }
```

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

```
2449  
2450 \NewDocumentCommand \symdecl { s m O{} } {  
2451   \__stex_symdecl_args:n { #3 }  
2452   \IfBooleanTF #1 {  
2453     \bool_set_false:N \l_stex_symdecl_make_macro_bool  
2454   } {  
2455     \bool_set_true:N \l_stex_symdecl_make_macro_bool  
2456   }  
2457   \stex_symdecl_do:n { #2 }
```

```

2458 \stex_smsmode_do:
2459 }
2460
2461 \cs_new_protected:Nn \stex_symdecl_do:nn {
2462   \__stex_symdecl_args:n{#1}
2463   \bool_set_false:N \l_stex_symdecl_make_macro_bool
2464   \stex_symdecl_do:n{#2}
2465 }
2466
2467 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

(End definition for \symdecl. This function is documented on page 85.)

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

```

2468 \cs_new_protected:Nn \stex_symdecl_do:n {
2469   \stex_if_in_module:F {
2470     % TODO throw error? some default namespace?
2471   }
2472
2473   \str_if_empty:NT \l_stex_symdecl_name_str {
2474     \str_set:Nx \l_stex_symdecl_name_str { #1 }
2475   }
2476
2477   \prop_if_exist:cT { l_stex_symdecl_
2478     \l_stex_current_module_str ?
2479     \l_stex_symdecl_name_str
2480   _prop
2481 }{
2482   % TODO throw error (beware of circular dependencies)
2483 }
2484
2485 \prop_clear:N \l_tmpa_prop
2486 \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
2487 \seq_clear:N \l_tmpa_seq
2488 \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
2489 \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
2490
2491 \str_if_empty:NT \l_stex_symdecl_deprecate_str {
2492   \str_if_empty:NF \l_stex_module_deprecate_str {
2493     \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
2494   }
2495 }
2496 \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
2497
2498 \exp_args:No \stex_add_constant_to_current_module:n {
2499   \l_stex_symdecl_name_str
2500 }
2501
2502 % arity/args
2503 \int_zero:N \l_tmpb_int
2504
2505 \bool_set_true:N \l_tmpa_bool
2506 \str_map_inline:Nn \l_stex_symdecl_args_str {
2507   \token_case_meaning:NnF ##1 {

```

```

2508     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2509     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2510     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2511     {\tl_to_str:n a} {
2512         \bool_set_false:N \l_tmpa_bool
2513         \int_incr:N \l_tmpb_int
2514     }
2515     {\tl_to_str:n B} {
2516         \bool_set_false:N \l_tmpa_bool
2517         \int_incr:N \l_tmpb_int
2518     }
2519 }{
2520     \msg_error:nnxx{stex}{error/wrongargs}{
2521         \l_stex_current_module_str ?
2522         \l_stex_symdecl_name_str
2523     }{##1}
2524 }
2525 }
2526
2527 \bool_if:NTF \l_tmpa_bool {
2528     % possibly numeric
2529     \str_if_empty:NTF \l_stex_symdecl_args_str {
2530         \prop_put:Nnn \l_tmpa_prop { args } {}
2531         \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2532     }{
2533         \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
2534         \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2535         \str_clear:N \l_tmpa_str
2536         \int_step_inline:nn \l_tmpa_int {
2537             \str_put_right:Nn \l_tmpa_str i
2538         }
2539         \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
2540     }
2541 } {
2542     \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2543     \prop_put:Nnx \l_tmpa_prop { arity }
2544     { \str_count:N \l_stex_symdecl_args_str }
2545 }
2546 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2547
2548 \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
2549     \prop_put:Nnx \l_tmpa_prop { defined }{ false }
2550 }{
2551     \prop_put:Nnx \l_tmpa_prop { defined }{ true }
2552 }
2553
2554 % argnames
2555
2556 \clist_clear:N \l_tmpa_clist
2557 \int_step_inline:nn {\prop_item:Nn \l_tmpa_prop {arity}} {
2558     \clist_if_empty:NTF \l_stex_symdecl_argnames_clist {
2559         \clist_put_right:Nn \l_tmpa_clist {##1}
2560     }{
2561         \clist_pop:NN \l_stex_symdecl_argnames_clist \l_tmpa_tl

```

```

2562     \exp_args:NNx \clist_put_right:Nn \l_tmpa_clist {\c_dollar_str\l_tmpa_tl}
2563   }
2564 }
2565 \prop_put:Nnx \l_tmpa_prop {argnames} {\clist_use:Nn \l_tmpa_clist ,}
2566
2567 % semantic macro
2568
2569 \bool_if:NT \l_stex_symdecl_make_macro_bool {
2570   \exp_args:Nx \stex_do_up_to_module:n {
2571     \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2572       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2573     }}
2574   }
2575 }
2576
2577 \stex_debug:nn{symbols}{New~symbol:~
2578   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2579   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2580   Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
2581   Definiens:~\exp_not:o {\l_stex_symdecl_definiens_tl}
2582 }
2583
2584 % circular dependencies require this:
2585 \stex_if_do_html:T {
2586   \stex_annotate_invisible:nnn {symdecl} {
2587     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2588   } {
2589     \tl_if_empty:NF \l_stex_symdecl_type_tl {
2590       \stex_annotate_invisible:nnn{type}{\l_stex_symdecl_type_tl$}
2591     }
2592     \stex_annotate_invisible:nnn{args}{\prop_item:Nn \l_tmpa_prop { args }}{}
2593     \stex_annotate_invisible:nnn{macroname}{#1}{}
2594     \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2595       \stex_annotate_invisible:nnn{definiens}{}
2596       {\l_stex_symdecl_definiens_tl$}
2597     }
2598     \str_if_empty:NF \l_stex_symdecl_assoctype_str {
2599       \stex_annotate_invisible:nnn{assoctype}{\l_stex_symdecl_assoctype_str}{}
2600     }
2601     \str_if_empty:NF \l_stex_symdecl_reorder_str {
2602       \stex_annotate_invisible:nnn{reorderargs}{\l_stex_symdecl_reorder_str}{}
2603     }
2604   }
2605 }
2606 \prop_if_exist:cF {
2607   l_stex_symdecl_
2608   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2609   _prop
2610 } {
2611   \bool_if:NTF \l_stex_symdecl_local_bool \stex_do_up_to_module:x \stex_execute_in_module:
2612     \_stex_symdecl_restore_symbol:nnnnnnn
2613     {\l_stex_symdecl_name_str}
2614     { \prop_item:Nn \l_tmpa_prop {args} }
2615     { \prop_item:Nn \l_tmpa_prop {arity} }

```

```

2616     { \prop_item:Nn \l_tmpa_prop {assocs} }
2617     { \prop_item:Nn \l_tmpa_prop {defined} }
2618     {\bool_if:NT \l_stex_symdecl_make_macro_bool {#1} }
2619     {\l_stex_current_module_str}
2620     { \prop_item:Nn \l_tmpa_prop {argnames} }
2621   }
2622 }
2623 }
2624 \cs_new_protected:Nn \__stex_symdecl_restore_symbol:nnnnnnnn {
2625   \prop_clear:N \l_tmpa_prop
2626   \prop_put:Nnn \l_tmpa_prop { module } { #7 }
2627   \prop_put:Nnn \l_tmpa_prop { name } { #1 }
2628   \prop_put:Nnn \l_tmpa_prop { args } { #2 }
2629   \prop_put:Nnn \l_tmpa_prop { arity } { #3 }
2630   \prop_put:Nnn \l_tmpa_prop { assocs } { #4 }
2631   \prop_put:Nnn \l_tmpa_prop { defined } { #5 }
2632   \prop_put:Nnn \l_tmpa_prop { argnames } { #8 }
2633   \tl_if_empty:nF{#6}{
2634     \tl_set:cx{#6}{\stex_invoke_symbol:n{\detokenize{#7 ? #1}}}
2635   }
2636   \prop_set_eq:cN{\l_stex_symdecl_ \detokenize{#7 ? #1} _prop}\l_tmpa_prop
2637   \seq_clear:c{\l_stex_symdecl_ \detokenize{#7 ? #1} _notations}
2638 }

```

(End definition for \stex\_symdecl\_do:n. This function is documented on page 86.)

## \textsymdecl

```

2639
2640 \keys_define:nn { stex / textsymdecl } {
2641   name      .str_set_x:N = \l__stex_symdecl_name_str ,
2642   type      .tl_set:N    = \l__stex_symdecl_type_tl
2643 }
2644
2645 \cs_new_protected:Nn \_stex_textsymdecl_args:n {
2646   \str_clear:N \l__stex_symdecl_name_str
2647   \tl_clear:N \l__stex_symdecl_type_tl
2648   \clist_clear:N \l_stex_symdecl_argnames_clist
2649   \keys_set:nn { stex / textsymdecl } { #1 }
2650 }
2651
2652 \NewDocumentCommand \textsymdecl {m O{} m} {
2653   \_stex_textsymdecl_args:n { #2 }
2654   \str_if_empty:NTF \l__stex_symdecl_name_str {
2655     \__stex_symdecl_args:n{name=#1,#2}
2656   }{
2657     \__stex_symdecl_args:n{#2}
2658   }
2659   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2660   \stex_symdecl_do:n{#1-sym}
2661   \stex_execute_in_module:n{
2662     \cs_set_nopar:cpn{#1name}{
2663       \ifvmode\hbox_unpack:N\c_empty_box\fi
2664       \ifmmode\hbox{#3}\else#3\fi\xspace
2665     }

```



```

2666 \cs_set_nopar:cpn{#1}{
2667   \ifmmode\csname#1-sym\expandafter\endcsname\else
2668   \ifvmode\hbox_unpack:N\c_empty_box\fi
2669   \symref{#1-sym}{#3}\expandafter\xspace
2670   \fi
2671 }
2672 }
2673 \stex_execute_in_module:x{
2674   \__stex_notation_restore_notation:nnnnn
2675   {\l_stex_current_module_str?\tl_if_empty:NTF\l__stex_symdecl_name_str{#1}\l__stex_symdecl_name_str{#1}}{0}
2676   {\exp_not:n{\STEXInternalTermMathOMSiiii{\STEXInternalCurrentSymbolStr}{}}{\neginfpref}{\neginfpref}}
2677   \comp{\hbox{#3}}\STEXInternalSymbolAfterInvokationTL
2678   }}}
2679 {}
2680 {}
2681 }
2682 \stex_smsmode_do:
2683 }

```

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

`\stex_get_symbol:n`

```

2684 \str_new:N \l_stex_get_symbol_uri_str
2685
2686 \cs_new_protected:Nn \stex_get_symbol:n {
2687   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2688     \tl_set:Nn \l_tmpa_tl { #1 }
2689     \__stex_symdecl_get_symbol_from_cs:
2690   }{
2691     % argument is a string
2692     % is it a command name?
2693     \cs_if_exist:cTF { #1 }{
2694       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2695       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2696       \str_if_empty:NTF \l_tmpa_str {
2697         \exp_args:Nx \cs_if_eq:NTF {
2698           \tl_head:N \l_tmpa_tl
2699         } \stex_invoke_symbol:n {
2700           \__stex_symdecl_get_symbol_from_cs:
2701         }{
2702           \__stex_symdecl_get_symbol_from_string:n { #1 }
2703         }
2704       } {
2705         \__stex_symdecl_get_symbol_from_string:n { #1 }
2706       }
2707     }{
2708       % argument is not a command name
2709       \__stex_symdecl_get_symbol_from_string:n { #1 }
2710       % \l_stex_all_symbols_seq
2711     }
2712   }
2713   \str_if_eq:eeF {
2714     \prop_item:cn {
2715       \l_stex_symdecl\l_stex_get_symbol_uri_str _prop

```

```

2716     }{ deprecate }
2717 }{}{
2718     \msg_warning:nnxx{stex}{warning/deprecated}{
2719         Symbol~\l_stex_get_symbol_uri_str
2720     }{
2721         \prop_item:cn {l_stex_symdecl_l\l_stex_get_symbol_uri_str _prop}{ deprecate }
2722     }
2723 }
2724 }
2725
2726 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2727     \tl_set:Nn \l_tmpa_tl {
2728         \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2729     }
2730     \str_set:Nn \l_tmpa_str { #1 }
2731
2732     %\int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2733
2734     \str_if_in:NnTF \l_tmpa_str ? {
2735         \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
2736         \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
2737         \str_set:Nx \l_tmpb_str {\seq_use:Nn \l_tmpa_seq ?}
2738     }{
2739         \str_clear:N \l_tmpb_str
2740     }
2741     \str_if_empty:NNTF \l_tmpb_str {
2742         \seq_map_inline:Nn \l_stex_all_modules_seq {
2743             \seq_map_inline:cn{c_stex_module_##1_constants}{
2744                 \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2745                     \seq_map_break:n{\seq_map_break:n{
2746                         \tl_set:Nn \l_tmpa_tl {
2747                             \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2748                         }
2749                     }}
2750                 }
2751             }
2752         }
2753     }{
2754         \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpb_str }
2755         \seq_map_inline:Nn \l_stex_all_modules_seq {
2756             \str_if_eq:eeT{ \l_tmpb_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2757                 \seq_map_inline:cn{c_stex_module_##1_constants}{
2758                     \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2759                         \seq_map_break:n{\seq_map_break:n{
2760                             \tl_set:Nn \l_tmpa_tl {
2761                                 \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2762                             }
2763                         }}
2764                     }
2765                 }
2766             }
2767         }
2768     }
2769

```

```

2770 \l_tmpa_tl
2771 }
2772
2773 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
2774   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2775     { \tl_tail:N \l_tmpa_tl }
2776   \tl_if_single:NTF \l_tmpa_tl {
2777     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2778       \exp_after:wN \str_set:Nn \exp_after:wN
2779         \l_stex_get_symbol_uri_str \l_tmpa_tl
2780     }{
2781       % TODO
2782       % tail is not a single group
2783     }
2784   }{
2785     % TODO
2786     % tail is not a single group
2787   }
2788 }

```

(End definition for `\stex_get_symbol:n`. This function is documented on page 86.)

## 29.2 Notations

```

2789 <@@=stex_notation>
      notation arguments:
2790 \keys_define:nn { stex / notation } {
2791   % lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2792   variant .tl_set_x:N   = \l__stex_notation_variant_str ,
2793   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2794   op        .tl_set:N   = \l__stex_notation_op_tl ,
2795   primary   .bool_set:N = \l__stex_notation_primary_bool ,
2796   primary   .default:n   = {true} ,
2797   hints     .str_set_x:N = \l__stex_notation_hints_str,
2798   unknown   .code:n      = \str_set:Nx
2799     \l__stex_notation_variant_str \l_keys_key_str
2800 }
2801
2802 \cs_new_protected:Nn \_stex_notation_args:n {
2803   % \str_clear:N \l__stex_notation_lang_str
2804   \str_clear:N \l__stex_notation_variant_str
2805   \str_clear:N \l__stex_notation_prec_str
2806   \str_clear:N \l__stex_notation_hints_str
2807   \tl_clear:N \l__stex_notation_op_tl
2808   \bool_set_false:N \l__stex_notation_primary_bool
2809
2810   \keys_set:nn { stex / notation } { #1 }
2811 }

```

**\notation**

```

2812 \NewDocumentCommand \notation { s m O{} } {
2813   \_stex_notation_args:n { #3 }
2814   \tl_clear:N \l_stex_symdecl_definiens_tl

```

```

2815 \stex_get_symbol:n { #2 }
2816 \tl_set:Nn \l_stex_notation_after_do_tl {
2817   \__stex_notation_final:
2818   \IfBooleanTF#1{
2819     \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2820   }{}
2821   \stex_smsmode_do:\ignorespacesandpars
2822 }
2823 \stex_notation_do:nnnnn
2824 { \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str _prop } { args } }
2825 { \prop_item:cn { l_stex_symdecl\l_stex_get_symbol_uri_str _prop } { arity } }
2826 { \l__stex_notation_variant_str }
2827 { \l__stex_notation_prec_str }
2828 }
2829 \stex_deactivate_macro:Nn \notation {module~environments}

```

(End definition for \notation. This function is documented on page 86.)

\stex\_notation\_do:nnnnn

```

2830 \seq_new:N \l__stex_notation_precedences_seq
2831 \tl_new:N \l__stex_notation_opprec_tl
2832 \int_new:N \l__stex_notation_currarg_int
2833 \tl_new:N \STEXInternalSymbolAfterInvokationTL
2834
2835 \cs_new_protected:Nn \stex_notation_do:nnnnn {
2836   \let\STEXInternalCurrentSymbolStr\relax
2837   \seq_clear:N \l__stex_notation_precedences_seq
2838   \tl_clear:N \l__stex_notation_opprec_tl
2839   \str_set:Nx \l__stex_notation_args_str { #1 }
2840   \str_set:Nx \l__stex_notation_arity_str { #2 }
2841   \str_set:Nx \l__stex_notation_suffix_str { #3 }
2842   \str_set:Nx \l__stex_notation_prec_str { #4 }
2843
2844   % precedences
2845   \str_if_empty:NTF \l__stex_notation_prec_str {
2846     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2847       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2848     }{
2849       \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2850     }
2851   } {
2852     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2853       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2854       \int_step_inline:nn { \l__stex_notation_arity_str } {
2855         \exp_args:NNo
2856         \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2857       }
2858     }{
2859       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2860       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2861         \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2862         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2863           \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2864             \l_tmpa_seq {\tl_to_str:n{x}} } { \l_tmpa_str }

```

```

2865         \seq_map_inline:Nn \l_tmpa_seq {
2866             \seq_put_right:Nn \l__stex_notation_precedences_seq { ##1 }
2867         }
2868     }
2869 }{
2870     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2871         \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2872     }{
2873         \tl_set:No \l__stex_notation_opprec_tl { 0 }
2874     }
2875 }
2876 }
2877 }
2878
2879 \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2880 \int_step_inline:nn { \l__stex_notation_arity_str } {
2881     \seq_pop_left:NnF \l_tmpa_seq \l_tmpb_seq {
2882         \exp_args:NNo
2883         \seq_put_right:No \l__stex_notation_precedences_seq {
2884             \l__stex_notation_opprec_tl
2885         }
2886     }
2887 }
2888 \tl_clear:N \l_stex_notation_dummyargs_tl
2889
2890 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2891     \exp_args:NNe
2892     \cs_set:Npn \l_stex_notation_macrocode_cs {
2893         \STEXInternalTermMathOMSiiii { \STEXInternalCurrentSymbolStr }
2894         { \l__stex_notation_suffix_str }
2895         { \l__stex_notation_opprec_tl }
2896         { \exp_not:n { #5 } }
2897     }
2898     \l_stex_notation_after_do_tl
2899 }{
2900     \str_if_in:NnTF \l__stex_notation_args_str b {
2901         \exp_args:Nne \use:nn
2902         {
2903             \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2904             \cs_set:Npn \l__stex_notation_arity_str } { {
2905                 \STEXInternalTermMathOMBiiii { \STEXInternalCurrentSymbolStr }
2906                 { \l__stex_notation_suffix_str }
2907                 { \l__stex_notation_opprec_tl }
2908                 { \exp_not:n { #5 } }
2909             }}
2910     }{
2911         \str_if_in:NnTF \l__stex_notation_args_str B {
2912             \exp_args:Nne \use:nn
2913             {
2914                 \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2915                 \cs_set:Npn \l__stex_notation_arity_str } { {
2916                     \STEXInternalTermMathOMBiiii { \STEXInternalCurrentSymbolStr }
2917                     { \l__stex_notation_suffix_str }
2918                     { \l__stex_notation_opprec_tl }

```

```

2919         { \exp_not:n { #5 } }
2920     } }
2921 }{
2922     \exp_args:Nne \use:nn
2923     {
2924         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2925         \cs_set:Npn \l__stex_notation_arity_str } { {
2926             \STEXInternalTermMathOMAiiai { \STEXInternalCurrentSymbolStr }
2927             { \l__stex_notation_suffix_str }
2928             { \l__stex_notation_opprec_tl }
2929             { \exp_not:n { #5 } }
2930         } }
2931     }
2932 }
2933
2934 \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2935 \int_zero:N \l__stex_notation_currarg_int
2936 \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2937 \__stex_notation_arguments:
2938 }
2939 }

```

(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

```

2940 \cs_new_protected:Nn \__stex_notation_arguments: {
2941     \int_incr:N \l__stex_notation_currarg_int
2942     \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2943         \l_stex_notation_after_do_tl
2944     }{
2945         \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2946         \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2947         \str_if_eq:NnTF \l_tmpa_str a {
2948             \__stex_notation_argument_assoc:nn{a}
2949         }{
2950             \str_if_eq:NnTF \l_tmpa_str B {
2951                 \__stex_notation_argument_assoc:nn{B}
2952             }{
2953                 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpb_str
2954                 \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2955                     { \STEXInternalTermMathArgiii
2956                       { \l_tmpa_str\int_use:N \l__stex_notation_currarg_int }
2957                       { \l_tmpb_str }
2958                       { ###\int_use:N \l__stex_notation_currarg_int }
2959                     }
2960                 }
2961                 \__stex_notation_arguments:
2962             }
2963         }
2964     }
2965 }

```

(End definition for `\__stex_notation_arguments:.`)

\\_stex\_notation\_argument\_assoc:nn

```

2966 \cs_new_protected:Nn \_stex_notation_argument_assoc:nn {
2967
2968   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2969     {\l_stex_notation_arity_str}{
2970       #2
2971     }
2972   \int_zero:N \l_tmpa_int
2973   \tl_clear:N \l_tmpa_tl
2974   \str_map_inline:Nn \l_stex_notation_args_str {
2975     \int_incr:N \l_tmpa_int
2976     \tl_put_right:Nx \l_tmpa_tl {
2977       \str_if_eq:nnTF {##1}{a}{ {} }{
2978         \str_if_eq:nnTF {##1}{B}{ {} }{
2979           {\_stex_term_arg:nn{##1}\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa_int}
2980         }
2981       }
2982     }
2983   }
2984   \exp_after:wN\exp_after:wN\exp_after:wN \def
2985   \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2986   \exp_after:wN\exp_after:wN\exp_after:wN ##
2987   \exp_after:wN\exp_after:wN\exp_after:wN 1
2988   \exp_after:wN\exp_after:wN\exp_after:wN ##
2989   \exp_after:wN\exp_after:wN\exp_after:wN 2
2990   \exp_after:wN\exp_after:wN\exp_after:wN {
2991     \exp_after:wN \exp_after:wN \exp_after:wN
2992     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2993       \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2994     }
2995   }
2996
2997   \seq_pop_left:NN \l_stex_notation_remaining_precs_seq \l_tmpa_str
2998   \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
2999     \STEXInternalTermMathAssocArgiiii
3000     { \int_use:N \l_stex_notation_currarg_int }
3001     { \l_tmpa_str }
3002     { ####\int_use:N \l_stex_notation_currarg_int }
3003     { \l_tmpa_cs {####1} {####2} }
3004     {#1}
3005   } }
3006   \_stex_notation_arguments:
3007 }

```

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

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

```

3008 \cs_new_protected:Nn \_stex_notation_restore_notation:nnnnn {
3009   \cs_generate_from_arg_count:cNnn{stex_notation_\detokenize{#1} \c_hash_str \detokenize{#2}
3010   \cs_set_nopar:Npn {#3}{#4}
3011   \tl_if_empty:nF {#5}{
3012     \tl_set:cn{stex_op_notation_\detokenize{#1} \c_hash_str \detokenize{#2}_cs}{ \comp{ #5 }
3013   }
3014   \seq_if_exist:cT { l_stex_symdecl_\detokenize{#1} _notations }{

```

```

3015     \seq_put_right:cx { l_stex_symdecl_\detokenize{#1} _notations } { \detokenize{#2} }
3016   }
3017 }
3018
3019 \cs_new_protected:Nn \__stex_notation_final: {
3020
3021   \stex_execute_in_module:x {
3022     \__stex_notation_restore_notation:nnnnn
3023     {\l_stex_get_symbol_uri_str}
3024     {\l__stex_notation_suffix_str}
3025     {\l__stex_notation_arity_str}
3026     {
3027       \exp_after:wN \exp_after:wN \exp_after:wN
3028       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
3029       { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \STEXInt
3030     }
3031     {\exp_args:No \exp_not:n \l__stex_notation_op_tl }
3032   }
3033
3034   \stex_debug:nn{symbols}{
3035     Notation~\l__stex_notation_suffix_str
3036     ~for~\l_stex_get_symbol_uri_str^^J
3037     Operator~precedence:~\l__stex_notation_opprec_tl^^J
3038     Argument~precedences:~
3039     \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
3040     Notation: \cs_meaning:c {
3041       stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
3042       \l__stex_notation_suffix_str
3043       _cs
3044     }
3045   }
3046   % HTML annotations
3047   \stex_if_do_html:T {
3048     \stex_annotate_invisible:nnn { notation }
3049     { \l_stex_get_symbol_uri_str } {
3050       \stex_annotate_invisible:nnn { notationfragment }
3051       { \l__stex_notation_suffix_str }{}
3052     }
3053     \stex_annotate_invisible:nnn { precedence }
3054     { \l__stex_notation_prec_str }{}
3055
3056     \int_zero:N \l_tmpa_int
3057     \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
3058     \tl_clear:N \l_tmpa_tl
3059     \int_step_inline:nn { \l__stex_notation_arity_str }{
3060       \int_incr:N \l_tmpa_int
3061       \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
3062       \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_rema
3063       \str_if_eq:VnTF \l_tmpb_str a {
3064         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3065           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{},
3066           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{},
3067         } }
3068       }{
3069         \str_if_eq:VnTF \l_tmpb_str B {

```



```

3069         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3070             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{ } ,
3071             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{ }
3072         } }
3073     }{
3074         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3075             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{ }
3076         } }
3077     }
3078 }
3079 }
3080 \stex_annotate_invisible:nnn { notationcomp }{ }{
3081     \str_set:Nx \STEXInternalCurrentSymbolStr {\l_stex_get_symbol_uri_str }
3082     $ \exp_args:Nno \use:nn { \use:c {
3083         stex_notation_ \STEXInternalCurrentSymbolStr
3084         \c_hash_str \l__stex_notation_suffix_str _cs
3085     } } { \l_tmpa_tl } $
3086 }
3087 \tl_if_empty:NF \l__stex_notation_op_tl {
3088     \stex_annotate_invisible:nnn { notationopcomp }{ }{
3089         $\l__stex_notation_op_tl$
3090     }
3091 }
3092 }
3093 }
3094 }

```

(End definition for `\_stex_notation_final:.`)

## `\setnotation`

```

3095 \keys_define:nn { stex / setnotation } {
3096     % lang .tl_set_x:N = \l__stex_notation_lang_str ,
3097     variant .tl_set_x:N = \l__stex_notation_variant_str ,
3098     unknown .code:n = \str_set:Nx
3099         \l__stex_notation_variant_str \l_keys_key_str
3100 }
3101
3102 \cs_new_protected:Nn \_stex_setnotation_args:n {
3103     % \str_clear:N \l__stex_notation_lang_str
3104     \str_clear:N \l__stex_notation_variant_str
3105     \keys_set:nn { stex / setnotation } { #1 }
3106 }
3107
3108 \cs_new_protected:Nn \_stex_notation_setnotation:nn {
3109     \seq_if_exist:CT{l_stex_symdecl_#1_notations}{
3110         \seq_remove_all:cn { l_stex_symdecl_#1_notations }{ #2 }
3111         \seq_put_left:cn { l_stex_symdecl_#1_notations }{ #2 }
3112     }
3113 }
3114
3115 \cs_new_protected:Nn \stex_setnotation:n {
3116     \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1_notations }
3117         { \l__stex_notation_variant_str }{
3118         \stex_execute_in_module:x{ \_stex_notation_setnotation:nn {#1}{\l__stex_notation_vari

```

```

3119 \stex_debug:nn {notations}{
3120   Setting~default~notation~
3121   {\l_stex_notation_variant_str }~for~
3122   #1 \
3123   \expandafter\meaning\csname
3124   l_stex_symdecl_#1 _notations\endcsname
3125 }
3126 }{
3127   \msg_error:nnxx{stex}{unknownnotation}{\l_stex_notation_variant_str}{#1}
3128 }
3129 }
3130
3131 \NewDocumentCommand \setnotation {m m} {
3132   \stex_get_symbol:n { #1 }
3133   \_stex_setnotation_args:n { #2 }
3134   \stex_setnotation:n{\l_stex_get_symbol_uri_str}
3135   \stex_smsmode_do:\ignorespacesandpars
3136 }
3137
3138 \cs_new_protected:Nn \stex_copy_notations:nn {
3139   \stex_debug:nn {notations}{
3140     Copying~notations~from~#2~to~#1\
3141     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
3142   }
3143   \tl_clear:N \l_tmpa_tl
3144   \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
3145     \tl_put_right:Nn \l_tmpa_tl { {##### #1} }
3146   }
3147   \seq_map_inline:cn {l_stex_symdecl_#2_notations}{\begingroup
3148     \stex_debug:nn{Here}{Here:~##1}
3149     \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
3150     \edef \l_tmpa_tl {
3151       \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
3152       \exp_after:wN\exp_after:wN\exp_after:wN {
3153         \exp_after:wN \l_tmpa_cs \l_tmpa_tl
3154       }
3155     }
3156
3157     \exp_after:wN \def \exp_after:wN \l_tmpa_tl
3158     \exp_after:wN ####\exp_after:wN 1 \exp_after:wN ####\exp_after:wN 2
3159     \exp_after:wN { \l_tmpa_tl }
3160
3161     \edef \l_tmpa_tl {
3162       \exp_after:wN \exp_not:n \exp_after:wN {
3163         \l_tmpa_tl {##### 1}{##### 2}
3164       }
3165     }
3166
3167     \stex_debug:nn{Here}{Here:~\expandafter\detokenize\expandafter{\l_tmpa_tl}}
3168
3169   \stex_execute_in_module:x {
3170     \_stex_notation_restore_notation:nnnnn
3171     {#1}{##1}
3172     { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }

```

```

3173     { \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl} }
3174     {
3175         \cs_if_exist:cT{stex_op_notation_ #2\c_hash_str ##1 _cs}{
3176             \exp_args:NNo\exp_args:No\exp_not:n{\csname stex_op_notation_ #2\c_hash_str ##1
3177             }
3178         }
3179     }\endgroup
3180 }
3181 }
3182
3183 \NewDocumentCommand \copynotation {m m} {
3184     \stex_get_symbol:n { #1 }
3185     \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
3186     \stex_get_symbol:n { #2 }
3187     \exp_args:Noo
3188     \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
3189     \stex_smsmode_do:\ignorespacesandpars
3190 }
3191

```

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

**\symdef**

```

3192 \keys_define:nn { stex / symdef } {
3193     name .str_set_x:N = \l_stex_symdecl_name_str ,
3194     local .bool_set:N = \l_stex_symdecl_local_bool ,
3195     args .str_set_x:N = \l_stex_symdecl_args_str ,
3196     type .tl_set:N = \l_stex_symdecl_type_tl ,
3197     def .tl_set:N = \l_stex_symdecl_definiens_tl ,
3198     reorder .str_set_x:N = \l_stex_symdecl_reorder_str ,
3199     op .tl_set:N = \l__stex_notation_op_tl ,
3200 % lang .str_set_x:N = \l__stex_notation_lang_str ,
3201     variant .str_set_x:N = \l__stex_notation_variant_str ,
3202     prec .str_set_x:N = \l__stex_notation_prec_str ,
3203     argnames .clist_set:N = \l_stex_symdecl_argnames_clist ,
3204     assoc .choices:nn =
3205         {bin,binl,binr,pre,conj,pwconj}
3206         {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
3207     unknown .code:n = \str_set:Nx
3208         \l__stex_notation_variant_str \l_keys_key_str
3209 }
3210
3211 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
3212     \str_clear:N \l_stex_symdecl_name_str
3213     \str_clear:N \l_stex_symdecl_args_str
3214     \str_clear:N \l_stex_symdecl_assoctype_str
3215     \str_clear:N \l_stex_symdecl_reorder_str
3216     \bool_set_false:N \l_stex_symdecl_local_bool
3217     \tl_clear:N \l_stex_symdecl_type_tl
3218     \tl_clear:N \l_stex_symdecl_definiens_tl
3219     \clist_clear:N \l_stex_symdecl_argnames_clist
3220 % \str_clear:N \l__stex_notation_lang_str
3221     \str_clear:N \l__stex_notation_variant_str
3222     \str_clear:N \l__stex_notation_prec_str

```

```

3223 \tl_clear:N \l__stex_notation_op_tl
3224
3225 \keys_set:nn { stex / symdef } { #1 }
3226 }
3227
3228 \NewDocumentCommand \symdef { m O{} } {
3229   \__stex_notation_symdef_args:n { #2 }
3230   \bool_set_true:N \l_stex_symdecl_make_macro_bool
3231   \stex_symdecl_do:n { #1 }
3232   \tl_set:Nn \l_stex_notation_after_do_tl {
3233     \__stex_notation_final:
3234     \stex_smsmode_do:\ignorespacesandpars
3235   }
3236   \str_set:Nx \l_stex_get_symbol_uri_str {
3237     \l_stex_current_module_str ? \l_stex_symdecl_name_str
3238   }
3239   \exp_args:Nx \stex_notation_do:nnnnn
3240     { \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str_prop} { args } }
3241     { \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str_prop} { arity } }
3242     { \l__stex_notation_variant_str }
3243     { \l__stex_notation_prec_str }
3244   }
3245   \stex_deactivate_macro:Nn \symdef {module~environments}

```

(End definition for `\symdef`. This function is documented on page 86.)

## 29.3 Variables

```

3246 <@@=stex_variables>
3247
3248 \keys_define:nn { stex / vardef } {
3249   name .str_set_x:N = \l__stex_variables_name_str ,
3250   args .str_set_x:N = \l__stex_variables_args_str ,
3251   type .tl_set:N = \l__stex_variables_type_tl ,
3252   def .tl_set:N = \l__stex_variables_def_tl ,
3253   op .tl_set:N = \l__stex_variables_op_tl ,
3254   prec .str_set_x:N = \l__stex_variables_prec_str ,
3255   reorder .str_set_x:N = \l__stex_variables_reorder_str ,
3256   argnames .clist_set:N = \l__stex_variables_argnames_clist ,
3257   assoc .choices:nn =
3258     {bin,binl,binr,pre,conj,pwconj}
3259     {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
3260   bind .choices:nn =
3261     {forall,exists}
3262     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
3263 }
3264
3265 \cs_new_protected:Nn \__stex_variables_args:n {
3266   \str_clear:N \l__stex_variables_name_str
3267   \str_clear:N \l__stex_variables_args_str
3268   \str_clear:N \l__stex_variables_prec_str
3269   \str_clear:N \l__stex_variables_assoctype_str
3270   \str_clear:N \l__stex_variables_reorder_str
3271   \str_clear:N \l__stex_variables_bind_str

```

```

3272 \tl_clear:N \l__stex_variables_type_tl
3273 \tl_clear:N \l__stex_variables_def_tl
3274 \tl_clear:N \l__stex_variables_op_tl
3275 \clist_clear:N \l__stex_variables_argnames_clist
3276
3277 \keys_set:nn { stex / vardef } { #1 }
3278 }
3279
3280 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{}} {
3281   \__stex_variables_args:n {#2}
3282   \str_if_empty:NT \l__stex_variables_name_str {
3283     \str_set:Nx \l__stex_variables_name_str { #1 }
3284   }
3285   \prop_clear:N \l_tmpa_prop
3286   \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
3287
3288   \int_zero:N \l_tmpb_int
3289   \bool_set_true:N \l_tmpa_bool
3290   \str_map_inline:Nn \l__stex_variables_args_str {
3291     \token_case_meaning:NnF ##1 {
3292       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
3293       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
3294       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
3295       {\tl_to_str:n a} {
3296         \bool_set_false:N \l_tmpa_bool
3297         \int_incr:N \l_tmpb_int
3298       }
3299       {\tl_to_str:n B} {
3300         \bool_set_false:N \l_tmpa_bool
3301         \int_incr:N \l_tmpb_int
3302       }
3303     }{
3304       \msg_error:nnxx{stex}{error/wrongargs}{
3305         variable~\l__stex_variables_name_str
3306       }{##1}
3307     }
3308   }
3309   \bool_if:NTF \l_tmpa_bool {
3310     % possibly numeric
3311     \str_if_empty:NTF \l__stex_variables_args_str {
3312       \prop_put:Nnn \l_tmpa_prop { args } {}
3313       \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
3314     }{
3315       \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
3316       \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
3317       \str_clear:N \l_tmpa_str
3318       \int_step_inline:nn \l_tmpa_int {
3319         \str_put_right:Nn \l_tmpa_str i
3320       }
3321       \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
3322       \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3323     }
3324   } {
3325     \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }

```

```

3326     \prop_put:Nnx \l_tmpa_prop { arity }
3327     { \str_count:N \l__stex_variables_args_str }
3328 }
3329 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
3330 \tl_set:cx { #1 }{ \stex_invoke_variable:n { \l__stex_variables_name_str } }
3331
3332 % argnames
3333
3334 \clist_clear:N \l_tmpa_clist
3335 \int_step_inline:nn {\prop_item:Nn \l_tmpa_prop {arity}} {
3336     \clist_if_empty:NTF \l__stex_variables_argnames_clist {
3337         \clist_put_right:Nn \l_tmpa_clist {##1}
3338     }{
3339         \clist_pop:NN \l__stex_variables_argnames_clist \l_tmpa_tl
3340         \exp_args:NNx \clist_put_right:Nn \l_tmpa_clist {\c_dollar_str\l_tmpa_tl}
3341     }
3342 }
3343 \prop_put:Nnx \l_tmpa_prop {argnames} {\clist_use:Nn \l_tmpa_clist ,}
3344
3345
3346 \prop_set_eq:cN { l_stex_symdecl_var://\l__stex_variables_name_str _prop} \l_tmpa_prop
3347
3348 \tl_if_empty:NF \l__stex_variables_op_tl {
3349     \cs_set:cpx {
3350         stex_var_op_notation_ \l__stex_variables_name_str _cs
3351     } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
3352 }
3353
3354 \tl_set:Nn \l_stex_notation_after_do_tl {
3355     \exp_args:Nne \use:nn {
3356         \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str _cs }
3357         \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
3358     } {{
3359         \exp_after:wN \exp_after:wN \exp_after:wN
3360         \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
3361         { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \STEXInter
3362     }}
3363     \stex_if_do_html:T {
3364         \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
3365             \stex_annotate_invisible:nnn { precedence }
3366             { \l__stex_variables_prec_str }{}
3367         }
3368         \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{\l__stex_variables_type_str}{
3369             \stex_annotate_invisible:nnn{args}{\l__stex_variables_args_str }{}
3370             \stex_annotate_invisible:nnn{macroname}{#1}{
3371         }
3372         \tl_if_empty:NF \l__stex_variables_def_tl {
3373             \stex_annotate_invisible:nnn{definiens}{
3374                 {\l__stex_variables_def_tl$}
3375             }
3376         }
3377         \str_if_empty:NF \l__stex_variables_assoctype_str {
3378             \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{
3379         }
3380         \str_if_empty:NF \l__stex_variables_reorder_str {
3381             \stex_annotate_invisible:nnn{reorderargs}{\l__stex_variables_reorder_str}{
3382         }
3383     }

```

```

3380 \int_zero:N \l_tmpa_int
3381 \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
3382 \tl_clear:N \l_tmpa_tl
3383 \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
3384   \int_incr:N \l_tmpa_int
3385   \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
3386   \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables
3387   \str_if_eq:VnTF \l_tmpb_str a {
3388     \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3389       \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3390       \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3391     } }
3392   }{
3393     \str_if_eq:VnTF \l_tmpb_str B {
3394       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3395         \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3396         \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3397       } }
3398     }{
3399       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3400         \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
3401       } }
3402     }
3403   }
3404 }
3405 \stex_annotate_invisible:nnn { notationcomp }{}{
3406   \str_set:Nx \STEXInternalCurrentSymbolStr {var://\l__stex_variables_name_str }
3407   $ \exp_args:Nno \use:nn { \use:c {
3408     stex_var_notation_\l__stex_variables_name_str _cs
3409   } } { \l_tmpa_tl } $
3410 }
3411 \tl_if_empty:NF \l__stex_variables_op_tl {
3412   \stex_annotate_invisible:nnn { notationopcomp }{}{
3413     $\l__stex_variables_op_tl$
3414   }
3415 }
3416 }
3417 \str_if_empty:NF \l__stex_variables_bind_str {
3418   \stex_annotate_invisible:nnn {bindtype}{\l__stex_variables_bind_str,\l__stex_variab
3419 }
3420 }\ignorespacesandpars
3421 }
3422
3423 \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
3424 }
3425
3426 \cs_new:Nn \_stex_reset:N {
3427   \tl_if_exist:NTF #1 {
3428     \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
3429   }{
3430     \let \exp_not:N #1 \exp_not:N \undefined
3431   }
3432 }
3433

```

```

3434 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
3435   \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
3436   \exp_args:Nnx \use:nn {
3437     % TODO
3438     \stex_annotate_invisible:nnn {vardecl}{\clist_use:Nn\l__stex_variables_names,}{
3439       #2
3440     }
3441   }{
3442     \stex_reset:N \varnot
3443     \stex_reset:N \vartype
3444     \stex_reset:N \vardefi
3445   }
3446 }
3447
3448 \NewDocumentCommand \vardef { s } {
3449   \IfBooleanTF#1 {
3450     \__stex_variables_do_complex:nn
3451   }{
3452     \__stex_variables_do_simple:nnn
3453   }
3454 }
3455
3456 \NewDocumentCommand \svar { 0{} m }{
3457   \tl_if_empty:nTF {#1}{
3458     \str_set:Nn \l_tmpa_str { #2 }
3459   }{
3460     \str_set:Nn \l_tmpa_str { #1 }
3461   }
3462   \stex_term_omv:nn {
3463     var://\l_tmpa_str
3464   }{
3465     \exp_args:Nnx \use:nn {
3466       \def\comp{\_varcomp}
3467       \str_set:Nx \STEXInternalCurrentSymbolStr { var://\l_tmpa_str }
3468       \comp{ #2 }
3469     }{
3470       \stex_reset:N \comp
3471       \stex_reset:N \STEXInternalCurrentSymbolStr
3472     }
3473   }
3474 }
3475
3476
3477
3478 \keys_define:nn { stex / varseq } {
3479   name .str_set_x:N = \l__stex_variables_name_str ,
3480   args .int_set:N = \l__stex_variables_args_int ,
3481   type .tl_set:N = \l__stex_variables_type_tl ,
3482   mid .tl_set:N = \l__stex_variables_mid_tl ,
3483   bind .choices:nn =
3484     {forall,exists}
3485     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
3486 }
3487

```



```

3488 \cs_new_protected:Nn \__stex_variables_seq_args:n {
3489   \str_clear:N \l__stex_variables_name_str
3490   \int_set:Nn \l__stex_variables_args_int 1
3491   \tl_clear:N \l__stex_variables_type_tl
3492   \str_clear:N \l__stex_variables_bind_str
3493
3494   \keys_set:nn { stex / varseq } { #1 }
3495 }
3496
3497 \NewDocumentCommand \varseq {m O{ } m m m}{
3498   \__stex_variables_seq_args:n { #2 }
3499   \str_if_empty:NT \l__stex_variables_name_str {
3500     \str_set:Nx \l__stex_variables_name_str { #1 }
3501   }
3502   \prop_clear:N \l_tmpa_prop
3503   \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
3504
3505   \seq_set_from_clist:Nn \l_tmpa_seq {#3}
3506   \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
3507     \msg_error:nnxx{stex}{error/seqlength}
3508     {\int_use:N \l__stex_variables_args_int}
3509     {\seq_count:N \l_tmpa_seq}
3510   }
3511   \seq_set_from_clist:Nn \l_tmpb_seq {#4}
3512   \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
3513     \msg_error:nnxx{stex}{error/seqlength}
3514     {\int_use:N \l__stex_variables_args_int}
3515     {\seq_count:N \l_tmpb_seq}
3516   }
3517   \prop_put:Nnn \l_tmpa_prop {starts} {#3}
3518   \prop_put:Nnn \l_tmpa_prop {ends} {#4}
3519
3520   \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
3521   \cs_set:Npn {\int_use:N \l__stex_variables_args_int} { #5 }
3522
3523   % argnames
3524
3525   \clist_clear:N \l_tmpa_clist
3526   \int_step_inline:nn {\l__stex_variables_args_int} {
3527     \clist_put_right:Nn \l_tmpa_clist {##1}
3528   }
3529   \prop_put:Nnx \l_tmpa_prop {argnames} {\clist_use:Nn \l_tmpa_clist ,}
3530
3531
3532
3533
3534   \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3535   \int_step_inline:nn \l__stex_variables_args_int {
3536     \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
3537   }
3538   \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
3539   \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
3540   \tl_if_empty:NF \l__stex_variables_mid_tl {
3541     \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl

```

```

3542 \tl_put_right:Nn \l_tmpa_tl {\,ellipses,}
3543 }
3544 \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq\l__stex_variables_name_str_cs}}
3545 \int_step_inline:nn \l__stex_variables_args_int {
3546 \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
3547 }
3548 \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
3549 \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
3550
3551
3552 \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl
3553
3554 \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}
3555
3556 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq\l__stex_variables_name_str_cs}}
3557
3558 \int_step_inline:nn \l__stex_variables_args_int {
3559 \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
3560 \STEXInternalTermMathArgiii{i##1}{0}{\exp_not:n{####}##1}
3561 }}
3562 }
3563
3564 \tl_set:Nx \l_tmpa_tl {
3565 \STEXInternalTermMathOMaiiii { varseq://\l__stex_variables_name_str}{\l__stex_variables_name_str}{0}{
3566 \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
3567 }
3568 }
3569
3570 \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \STEXInternalSymbolAfterInvokationTL} }
3571
3572 \exp_args:Nno \use:nn {
3573 \cs_generate_from_arg_count:cNnn {stex_varseq\l__stex_variables_name_str_cs}
3574 \cs_set:Npn {\int_use:N \l__stex_variables_args_int}{\l_tmpa_tl}
3575
3576 \stex_debug:nn{sequences}{New~Sequence:~
3577 \expandafter\meaning\csname stex_varseq\l__stex_variables_name_str_cs\endcsname\\~\\
3578 \prop_to_keyval:N \l_tmpa_prop
3579 }
3580 \prop_set_eq:cN {\l_stex_symdecl_varseq://\l__stex_variables_name_str_prop}\l_tmpa_prop
3581
3582 \stex_if_do_html:T{\stex_annotate_invisible:nnn{varseq}{\l__stex_variables_name_str}{
3583 \tl_if_empty:NF \l__stex_variables_type_tl {
3584 \stex_annotate:nnn {type}{\l__stex_variables_type_tl}
3585 }
3586 \stex_annotate:nnn {args}{\int_use:N \l__stex_variables_args_int}{
3587 \str_if_empty:NF \l__stex_variables_bind_str {
3588 \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{
3589 }
3590 \stex_annotate:nnn{startindex}{\l__stex_variables_startindex}{
3591 \stex_annotate:nnn{endindex}{\l__stex_variables_endindex}{
3592
3593 \tl_clear:N \l_tmpa_tl
3594 \int_step_inline:nn \l__stex_variables_args_int {
3595 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {

```

```

3596     \stex_annotate:nnn{argmarker}{##1}{-}
3597   } }
3598 }
3599 \stex_annotate_invisible:nnn { notationcomp }{}{
3600   \str_set:Nx \STEXInternalCurrentSymbolStr {varseq://\l__stex_variables_name_str }
3601   $ \exp_args:Nno \use:nn { \use:c {
3602     stex_varseq_\l__stex_variables_name_str _cs
3603   } } { \l_tmpa_tl } $
3604 }
3605 \stex_annotate_invisible:nnn { notationopcomp }{}{
3606   $ \prop_item:Nn \l_tmpa_prop { notation } $
3607 }
3608
3609 }}
3610
3611 \ignorespacesandpars
3612 }
3613
3614 </package>

```

## Chapter 30

# STEX -Terms Implementation

```
3615 <*package>
3616
3617 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
3618
3619 <@@=stex_terms>
3620
3621   Warnings and error messages
3622 \msg_new:nnn{stex}{error/nonotation}{
3623   Symbol~#1~invoked,~but~has~no~notation#2!
3624 }
3625 \msg_new:nnn{stex}{error/notationarg}{
3626   Error~in~parsing~notation~#1
3627 }
3628 \msg_new:nnn{stex}{error/noop}{
3629   Symbol~#1~has~no~operator~notation~for~notation~#2
3630 }
3631 \msg_new:nnn{stex}{error/notallowed}{
3632   Symbol~invocation~#1~not~allowed~in~notation~component~of~#2
3633 }
3634 \msg_new:nnn{stex}{error/doubleargument}{
3635   Argument~#1~of~symbol~#2~already~assigned
3636 }
3637 \msg_new:nnn{stex}{error/overarity}{
3638   Argument~#1~invalid~for~symbol~#2~with~arity~#3
3639 }
```

### 30.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
3639
3640
3641 \bool_new:N \l_stex_allow_semantic_bool
3642 \bool_set_true:N \l_stex_allow_semantic_bool
3643
```

```

3644 \cs_new_protected:Nn \stex_invoke_symbol:n {
3645   \ifvmode\indent\fi
3646   \bool_if:NTF \l_stex_allow_semantic_bool {
3647     \str_if_eq:eeF {
3648       \prop_item:cn {
3649         l_stex_symdecl_#1_prop
3650       }{ deprecate }
3651     }{}{
3652       \msg_warning:nxxx{stex}{warning/deprecated}{
3653         Symbol~#1
3654       }{
3655         \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
3656       }
3657     }
3658     \if_mode_math:
3659       \exp_after:wN \__stex_terms_invoke_math:n
3660     \else:
3661       \exp_after:wN \__stex_terms_invoke_text:n
3662     \fi: { #1 }
3663   }{
3664     \msg_error:nxxx{stex}{error/notallowed}{#1}{\STEXInternalCurrentSymbolStr}
3665   }
3666 }
3667
3668 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
3669   \peek_charcode_remove:NTF ! {
3670     \__stex_terms_invoke_op_custom:nn {#1}
3671   }{
3672     \__stex_terms_invoke_custom:nn {#1}
3673   }
3674 }
3675
3676 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
3677   \peek_charcode_remove:NTF ! {
3678     % operator
3679     \peek_charcode_remove:NTF * {
3680       % custom op
3681       \__stex_terms_invoke_op_custom:nn {#1}
3682     }{
3683       % op notation
3684       \peek_charcode:NTF [ {
3685         \__stex_terms_invoke_op_notation:nw {#1}
3686       }{
3687         \__stex_terms_invoke_op_notation:nw {#1}[]
3688       }
3689     }
3690   }{
3691     \peek_charcode_remove:NTF * {
3692       \__stex_terms_invoke_custom:nn {#1}
3693       % custom
3694     }{
3695       % normal
3696       \peek_charcode:NTF [ {
3697         \__stex_terms_invoke_notation:nw {#1}

```

```

3698     }{
3699         \_stex_terms_invoke_notation:nw {#1}[]
3700     }
3701 }
3702 }
3703 }
3704
3705
3706 \cs_new_protected:Nn \_stex_terms_invoke_op_custom:nn {
3707     \exp_args:Nnx \use:nn {
3708         \def\comp{\_comp}
3709         \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3710         \bool_set_false:N \l_stex_allow_semantic_bool
3711         \stex_mathml_intent:nn{#1}{
3712             \_stex_term_oms:nnn {#1}{#1 \c_hash_str CUSTOM-}{
3713                 \comp{ #2 }
3714             }
3715         }
3716     }{
3717         \_stex_reset:N \comp
3718         \_stex_reset:N \STEXInternalCurrentSymbolStr
3719         \bool_set_true:N \l_stex_allow_semantic_bool
3720     }
3721 }
3722
3723 \keys_define:nn { stex / terms } {
3724     % lang      .tl_set_x:N = \l_stex_notation_lang_str ,
3725     variant .tl_set_x:N = \l_stex_notation_variant_str ,
3726     unknown .code:n      = \str_set:Nx
3727         \l_stex_notation_variant_str \l_keys_key_str
3728 }
3729
3730 \cs_new_protected:Nn \_stex_terms_args:n {
3731     % \str_clear:N \l_stex_notation_lang_str
3732     \str_clear:N \l_stex_notation_variant_str
3733
3734     \keys_set:nn { stex / terms } { #1 }
3735 }
3736
3737 \cs_new_protected:Nn \stex_find_notation:nn {
3738     \_stex_terms_args:n { #2 }
3739     \seq_if_empty:cTF {
3740         l_stex_symdecl_ #1 _notations
3741     } {
3742         \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
3743     } {
3744         \str_if_empty:NTF \l_stex_notation_variant_str {
3745             \seq_get_left:cN {l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
3746         }{
3747             \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
3748                 \l_stex_notation_variant_str
3749             }{
3750                 % \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3751             }{

```

```

3752         \msg_error:nxxx{stex}{error/nonotation}{#1}{
3753         ~\l_stex_notation_variant_str
3754         }
3755     }
3756 }
3757 }
3758 }
3759
3760 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3761     \exp_args:Nnx \use:nn {
3762         \def\comp{\_comp}
3763         \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3764         \stex_find_notation:nn { #1 }{ #2 }
3765         \bool_set_false:N \l_stex_allow_semantic_bool
3766         \cs_if_exist:cTF {
3767             stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3768         }{
3769             \_stex_term_oms:nnn { #1 }{
3770                 #1 \c_hash_str \l_stex_notation_variant_str
3771             }{
3772                 \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3773             }
3774         }{
3775             \int_compare:nNnTF {\prop_item:cn {l_stex_symdecl_#1_prop}{arity}} = 0{
3776                 \cs_if_exist:cTF {
3777                     stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3778                 }{
3779                     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
3780                         \_stex_reset:N \comp
3781                         \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
3782                         \_stex_reset:N \STEXInternalCurrentSymbolStr
3783                         \bool_set_true:N \l_stex_allow_semantic_bool
3784                     }
3785                     \def\comp{\_comp}
3786                     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3787                     \bool_set_false:N \l_stex_allow_semantic_bool
3788                     \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3789                 }{
3790                     \msg_error:nxxx{stex}{error/nonotation}{#1}{
3791                     ~\l_stex_notation_variant_str
3792                     }
3793                 }
3794             }{
3795                 \msg_error:nxxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3796             }
3797         }
3798     }{
3799         \_stex_reset:N \comp
3800         \_stex_reset:N \STEXInternalCurrentSymbolStr
3801         \bool_set_true:N \l_stex_allow_semantic_bool
3802     }
3803 }
3804
3805 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {

```

```

3806 \stex_find_notation:nn { #1 }{ #2 }
3807 \cs_if_exist:cTF {
3808   stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3809 }{
3810   \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
3811     \_stex_reset:N \comp
3812     \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
3813     \_stex_reset:N \STEXInternalCurrentSymbolStr
3814     \bool_set_true:N \l_stex_allow_semantic_bool
3815   }
3816   \def\comp{\_comp}
3817   \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3818   \bool_set_false:N \l_stex_allow_semantic_bool
3819   \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3820 }{
3821   \msg_error:nnxx{stex}{error/nonotation}{#1}{
3822     ~\l_stex_notation_variant_str
3823   }
3824 }
3825 }
3826
3827 \prop_new:N \l__stex_terms_custom_args_prop
3828 \clist_new:N \l_stex_argnames_seq
3829 \seq_new:N \l__stex_terms_tmp_seq
3830
3831 \cs_new_protected:Nn \__stex_terms_custom_comp:n{ \bool_set_false:N \l_stex_allow_semantic_bo
3832
3833 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3834   \exp_args:Nnx \use:nn {
3835     \def\comp{\__stex_terms_custom_comp:n}
3836     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3837     \prop_clear:N \l__stex_terms_custom_args_prop
3838     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3839     \prop_get:cnN {
3840       l_stex_symdecl_#1 _prop
3841     }{ args } \l_tmpa_str
3842     \exp_args:NNx \seq_set_from_clist:Nn \l_stex_argnames_seq {
3843       \prop_item:cn {l_stex_symdecl_#1 _prop}{argnames}
3844     }
3845     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3846     \tl_set:Nn \arg { \__stex_terms_arg: }
3847     \str_if_empty:NTF \l_tmpa_str {
3848       \stex_mathml_intent:nn{#1}{
3849         \_stex_term_oms:nnn {#1}{#1\c_hash_str CUSTOM-}{\ignorespaces#2}
3850       }
3851     }{
3852       \seq_clear:N \l__stex_terms_tmp_seq
3853       \exp_args:Nx\int_step_inline:nn{\prop_item:cn{l_stex_symdecl_#1 _prop}{arity}}{
3854         \tl_set:Nx \l__stex_terms_tmp_tl {\seq_item:Nn \l_stex_argnames_seq {##1}}
3855         \bool_lazy_or:nnT{
3856           \str_if_eq_p:nn{a}{\str_item:Nn \l_tmpa_str{##1}}
3857         }{
3858           \str_if_eq_p:nn{B}{\str_item:Nn \l_tmpa_str{##1}}
3859         }{

```



```

3860         \tl_put_right:Nn \l__stex_terms_tmp_tl +
3861     }
3862     \seq_put_right:No \l__stex_terms_tmp_seq \l__stex_terms_tmp_tl
3863 }
3864 \stex_mathml_intent:nn{
3865     #1[\prop_item:cn {l_stex_symdecl_#1 _prop}]{ args }(
3866         \seq_use:Nn \l__stex_terms_tmp_seq ,
3867     )
3868 }{
3869     \str_if_in:NnTF \l_tmpa_str b {
3870         \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
3871     }{
3872         \str_if_in:NnTF \l_tmpa_str B {
3873             \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
3874         }{
3875             \_stex_term_oma:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
3876         }
3877     }
3878 }
3879 }
3880 % TODO check that all arguments exist
3881 }{
3882     \_stex_reset:N \l_stex_argnames_seq
3883     \_stex_reset:N \STEXInternalCurrentSymbolStr
3884     \_stex_reset:N \arg
3885     \_stex_reset:N \comp
3886     \_stex_reset:N \l__stex_terms_custom_args_prop
3887     %\bool_set_true:N \l_stex_allow_semantic_bool
3888 }
3889 }
3890
3891 \NewDocumentCommand \__stex_terms_arg: { s O{} m}{
3892     \tl_if_empty:nTF {#2}{
3893         \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
3894         \bool_set_true:N \l_tmpa_bool
3895         \bool_do_while:Nn \l_tmpa_bool {
3896             \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int}
3897             \int_incr:N \l_tmpa_int
3898         }{
3899             \bool_set_false:N \l_tmpa_bool
3900         }
3901     }
3902     }{
3903         \int_set:Nn \l_tmpa_int { #2 }
3904     }
3905     \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
3906     \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3907         \msg_error:nnxxx{stex}{error/overarity}
3908         {\int_use:N \l_tmpa_int}
3909         {\STEXInternalCurrentSymbolStr}
3910         {\str_count:N \l_tmpa_str}
3911     }
3912     \str_set:Nx \l_tmpa_str {\str_item:Nn \l_tmpa_str \l_tmpa_int}
3913     \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {

```

```

3914 \bool_lazy_any:nF {
3915   {\str_if_eq_p:Vn \l_tmpa_str {a}}
3916   {\str_if_eq_p:Vn \l_tmpa_str {B}}
3917 }{
3918   \msg_error:nnxx{stex}{error/doubleargument}
3919   {\int_use:N \l_tmpa_int}
3920   {\STEXInternalCurrentSymbolStr}
3921 }
3922 }
3923 \exp_args:NNx \prop_put:Nnn \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {\ign
3924 \bool_if:NTF \l_stex_allow_semantic_bool \use_i:nn {
3925   \bool_set_true:N \l_stex_allow_semantic_bool
3926   \use:nn
3927 }
3928 {
3929 \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq \l_tmpa_int}{
3930   \IfBooleanTF#1{
3931     \stex_annotate_invisible:n { %TODO
3932       \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
3933     }
3934   }{ %TODO
3935     \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
3936   }
3937 }}
3938 {\bool_set_false:N \l_stex_allow_semantic_bool}
3939 }
3940
3941
3942 \cs_new_protected:Nn \_stex_term_arg:nn {
3943   \bool_set_true:N \l_stex_allow_semantic_bool
3944   \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3945   \bool_set_false:N \l_stex_allow_semantic_bool
3946 }
3947
3948 \cs_new_protected:Npn \STEXInternalTermMathArgiii #1#2#3 {
3949   \exp_args:Nnx \use:nn
3950   { \int_set:Nn \l__stex_terms_downprec { #2 }
3951     \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq \l_tmpa_int}{
3952       \_stex_term_arg:nn { #1 }{ #3 }
3953     }
3954   }
3955   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3956 }

```

(End definition for `\stex_invoke_symbol:n`. This function is documented on page 87.)

`\STEXInternalTermMathAssocArgiiii`

```

3957 \cs_new_protected:Npn \STEXInternalTermMathAssocArgiiii #1#2#3#4#5 {
3958   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3959   \tl_set:Nn \l_tmpb_tl {\STEXInternalTermMathArgiii{#5#1}{#2}}
3960   \tl_if_empty:nTF { #3 }{
3961     \STEXInternalTermMathArgiii{#5#1}{#2}{}
3962   }{
3963     \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 } }{

```

```

3964 \expandafter\if\expandafter\relax\noexpand#3
3965 \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_maybe_sequence:Nnn#3{#1}{#5}}
3966 \else
3967 \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_simple:nnn{#1}{#3}{#5}}
3968 \fi
3969 \l_tmpa_tl
3970 }{
3971 \__stex_terms_math_assoc_arg_simple:nnn{#1}{#3}{#5}
3972 }
3973 }
3974 }
3975
3976 \cs_new_protected:Nn \__stex_terms_math_assoc_arg_maybe_sequence:Nnn {
3977 \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
3978 \str_if_empty:NTF \l_tmpa_str {
3979 \exp_args:Nx \cs_if_eq:NNTF {
3980 \tl_head:N #1
3981 } \stex_invoke_sequence:n {
3982 \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
3983 \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
3984 \tl_set:Nx \l_tmpa_tl {\prop_item:cn {l_stex_symdecl_varseq://\l_tmpa_str _prop}{notat
3985 \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
3986 \tl_set:Nx \l_tmpa_tl {\exp_not:N \exp_not:n{
3987 \exp_not:n{\exp_args:Nnx \use:nn} {
3988 \exp_not:n {
3989 \def\comp{\_varcomp}
3990 \str_set:Nn \STEXInternalCurrentSymbolStr
3991 } {varseq://\l_tmpa_str}
3992 \exp_not:n{ ##1 }
3993 }{
3994 \exp_not:n {
3995 \_stex_reset:N \comp
3996 \_stex_reset:N \STEXInternalCurrentSymbolStr
3997 }
3998 }
3999 }}}
4000 \exp_args:Nno \use:n {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
4001 \seq_reverse:N \l_tmpa_seq
4002 \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
4003 \seq_map_inline:Nn \l_tmpa_seq {
4004 \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
4005 \exp_args:Nno
4006 \l_tmpa_cs { ##1 } \l_tmpa_tl
4007 }
4008 }
4009 \tl_set:Nx \l_tmpa_tl {
4010 \_stex_term_omv:nn {varseq://\l_tmpa_str}{
4011 \exp_args:No \exp_not:n \l_tmpa_tl
4012 }
4013 }
4014 \exp_args:No\l_tmpb_tl\l_tmpa_tl
4015 }{
4016 \__stex_terms_math_assoc_arg_simple:nnn{#2} { #1 }{#3}
4017 }

```

```

4018 } {
4019   \_stex_terms_math_assoc_arg_simple:nnn{#2} { #1 }{#3}
4020 }
4021
4022 }
4023
4024 \cs_new_protected:Nn \_stex_terms_math_assoc_arg_simple:nnn {
4025   \clist_set:Nn \l_tmpa_clist{ #2 }
4026   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
4027     \tl_set:Nn \l_tmpa_tl {
4028       \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq #1}{
4029         \_stex_term_arg:nn{A#3#1}{ #2 } }
4030     }
4031   }{
4032     \clist_reverse:N \l_tmpa_clist
4033     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
4034     \tl_set:Nx \l_tmpa_tl {
4035       \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq #1}{
4036         \_stex_term_arg:nn{A#3#1}{
4037           \exp_args:No \exp_not:n \l_tmpa_tl
4038         }
4039       }
4040     }
4041     \clist_map_inline:Nn \l_tmpa_clist {
4042       \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
4043         \exp_args:Nno
4044           \l_tmpa_cs {
4045             \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq #1}{
4046               \_stex_term_arg:nn{A#3#1}{##1}
4047             }
4048           } \l_tmpa_tl
4049         }
4050       }
4051     }
4052     \exp_args:No\l_tmpb_tl\l_tmpa_tl
4053   }

```

(End definition for `\STEXInternalTermMathAssocArgiiii`. This function is documented on page 88.)

## 30.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
4053 \tl_const:Nx \infprec {\int_use:N \c_max_int}
4054 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
4055 \int_new:N \l__stex_terms_downprec
4056 \int_set_eq:NN \l__stex_terms_downprec \infprec

```

(End definition for `\infprec`, `\neginfprec`, and `\l__stex_terms_downprec`. These variables are documented on page 88.)

Bracketing:

```

\l__stex_terms_left_bracket_str
\l__stex_terms_right_bracket_str
4057 \tl_set:Nn \l__stex_terms_left_bracket_str (
4058 \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

```

4059 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
4060   \bool_if:NTF \l__stex_terms_brackets_done_bool {
4061     \bool_set_false:N \l__stex_terms_brackets_done_bool
4062     #2
4063   } {
4064     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
4065       \bool_if:NTF \l__stex_inarray_bool { #2 }{
4066         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
4067         \dobrackets { #2 }
4068       }
4069     }{ #2 }
4070   }
4071 }

```

(End definition for \\_stex\_terms\_maybe\_brackets:nn.)

**\dobrackets**

```

4072 \bool_new:N \l__stex_terms_brackets_done_bool
4073 %\RequirePackage{scalereel}
4074 \cs_new_protected:Npn \dobrackets #1 {
4075   %\ThisStyle{\if D\m@switch
4076   %   \exp_args:Nnx \use:nn
4077   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
4078   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
4079   %   \else
4080   %   \exp_args:Nnx \use:nn
4081   %   {
4082   %     \bool_set_true:N \l__stex_terms_brackets_done_bool
4083   %     \int_set:Nn \l__stex_terms_downprec \infpref
4084   %     \l__stex_terms_left_bracket_str
4085   %     #1
4086   %   }
4087   %   {
4088   %     \bool_set_false:N \l__stex_terms_brackets_done_bool
4089   %     \l__stex_terms_right_bracket_str
4090   %     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
4091   %   }
4092   %\fi}
4093 }

```

(End definition for \dobrackets. This function is documented on page 88.)

**\withbrackets**

```

4094 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
4095   \exp_args:Nnx \use:nn
4096   {
4097     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
4098     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
4099     #3
4100   }
4101 }

```

```

4102     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
4103     {\l__stex_terms_left_bracket_str}
4104     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
4105     {\l__stex_terms_right_bracket_str}
4106   }
4107 }

```

(End definition for `\withbrackets`. This function is documented on page 88.)

## `\STEXinvisible`

```

4108 \cs_new_protected:Npn \STEXinvisible #1 {
4109   \stex_annotate_invisible:n { #1 }
4110 }

```

(End definition for `\STEXinvisible`. This function is documented on page 88.)

OMDoc terms:

## `\STEXInternalTermMathOMSiiii`

```

4111 \cs_new_protected:Nn \_stex_term_oms:nnn {
4112   \stex_annotate:nnn{ OMID }{ #2 }{
4113     #3
4114   }
4115 }
4116
4117 \cs_new_protected:Npn \STEXInternalTermMathOMSiiii #1#2#3#4 {
4118   \__stex_terms_maybe_brackets:nn { #3 }{
4119     \stex_mathml_intent:nn{#1} {
4120       \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
4121     }
4122   }
4123 }

```

(End definition for `\STEXInternalTermMathOMSiiii`. This function is documented on page 87.)

## `\_stex_term_math_omv:nn`

```

4124 \cs_new_protected:Nn \_stex_term_omv:nn {
4125   \stex_annotate:nnn{ OMV }{ #1 }{
4126     #2
4127   }
4128 }

```

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

## `\STEXInternalTermMathOMAiiai`

```

4129 \cs_new_protected:Nn \_stex_term_oma:nnn {
4130   \stex_annotate:nnn{ OMA }{ #2 }{
4131     #3
4132   }
4133 }
4134
4135 \cs_new_protected:Npn \STEXInternalTermMathOMAiiai #1#2#3#4 {
4136   \exp_args:Nnx \use:nn {
4137     \seq_clear:N \l__stex_terms_tmp_seq
4138     \prop_if_exist:cT{l_stex_symdecl_#1 _prop}{
4139       \exp_args:NNx \seq_set_from_clist:Nn \l_stex_argnames_seq {

```

```

4140     \prop_item:cn {l_stex_symdecl_#1 _prop}{argnames}
4141   }
4142   \exp_args:Nx\int_step_inline:nn{\prop_item:cn{l_stex_symdecl_#1 _prop}{arity}}{
4143     \tl_set:Nx \l__stex_terms_tmp_tl {\seq_item:Nn \l_stex_argnames_seq {##1}}
4144     \bool_lazy_or:nnT{
4145       \str_if_eq_p:nn{a}{\str_item:Nn\l_tmpa_str{##1}}
4146     }{
4147       \str_if_eq_p:nn{B}{\str_item:Nn\l_tmpa_str{##1}}
4148     }{
4149       \tl_put_right:Nn \l__stex_terms_tmp_tl +
4150     }
4151     \seq_put_right:No \l__stex_terms_tmp_seq \l__stex_terms_tmp_tl
4152   }
4153 }
4154 \__stex_terms_maybe_brackets:nn { #3 }{
4155   \stex_mathml_intent:nn{
4156     #1[\prop_item:cn {l_stex_symdecl_#1 _prop}{ args }](
4157       \seq_use:Nn \l__stex_terms_tmp_seq ,
4158     )
4159   }{
4160     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
4161   }
4162 }
4163 }{
4164   \_stex_reset:N \l_stex_argnames_seq
4165 }
4166 }

```

(End definition for `\STEXInternalTermMathOMAi`. This function is documented on page 87.)

`\STEXInternalTermMathOMBiiii`

```

4167 \cs_new_protected:Nn \_stex_term_ombind:nnn {
4168   \stex_annotate:nnn{ OMBIND }{ #2 }{
4169     #3
4170   }
4171 }
4172
4173 \cs_new_protected:Npn \STEXInternalTermMathOMBiiii #1#2#3#4 {
4174   \exp_args:Nnx \use:nn {
4175     \seq_clear:N \l__stex_terms_tmp_seq
4176     \prop_if_exist:cT{l_stex_symdecl_#1 _prop}{
4177       \exp_args:NNx \seq_set_from_clist:Nn \l_stex_argnames_seq {
4178         \prop_item:cn {l_stex_symdecl_#1 _prop}{argnames}
4179       }
4180       \exp_args:Nx\int_step_inline:nn{\prop_item:cn{l_stex_symdecl_#1 _prop}{arity}}{
4181         \tl_set:Nx \l__stex_terms_tmp_tl {\seq_item:Nn \l_stex_argnames_seq {##1}}
4182         \bool_lazy_or:nnT{
4183           \str_if_eq_p:nn{a}{\str_item:Nn\l_tmpa_str{##1}}
4184         }{
4185           \str_if_eq_p:nn{B}{\str_item:Nn\l_tmpa_str{##1}}
4186         }{
4187           \tl_put_right:Nn \l__stex_terms_tmp_tl +
4188         }
4189         \seq_put_right:No \l__stex_terms_tmp_seq \l__stex_terms_tmp_tl

```

```

4190     }
4191   }
4192   \__stex_terms_maybe_brackets:nn { #3 }{
4193     \stex_mathml_intent:nn{
4194       #1[\prop_item:cn {l_stex_symdecl_#1 _prop}{ args }](
4195         \seq_use:Nn \l__stex_terms_tmp_seq ,
4196       )
4197     }{
4198       \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
4199     }
4200   }
4201   }{
4202     \stex_reset:N \l_stex_argnames_seq
4203   }
4204 }

```

(End definition for `\STEXInternalTermMathOMBiiii`. This function is documented on page 87.)

`\symref`  
`\symname`

```

4205 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
4206
4207 \keys_define:nn { stex / symname } {
4208   pre      .tl_set_x:N      = \l__stex_terms_pre_tl ,
4209   post     .tl_set_x:N      = \l__stex_terms_post_tl ,
4210   root     .tl_set_x:N      = \l__stex_terms_root_tl
4211 }
4212
4213 \cs_new_protected:Nn \stex_symname_args:n {
4214   \tl_clear:N \l__stex_terms_post_tl
4215   \tl_clear:N \l__stex_terms_pre_tl
4216   \tl_clear:N \l__stex_terms_root_str
4217   \keys_set:nn { stex / symname } { #1 }
4218 }
4219
4220 \NewDocumentCommand \symref { m m }{
4221   \let\compemph_uri_prev:\compemph@uri
4222   \let\compemph@uri\symrefemph@uri
4223   \STEXsymbol{#1}!\{ #2 }
4224   \let\compemph@uri\compemph_uri_prev:
4225 }
4226
4227 \NewDocumentCommand \synonym { 0{} m m }{
4228   \stex_symname_args:n { #1 }
4229   \let\compemph_uri_prev:\compemph@uri
4230   \let\compemph@uri\symrefemph@uri
4231   % TODO
4232   \STEXsymbol{#2}!\{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
4233   \let\compemph@uri\compemph_uri_prev:
4234 }
4235
4236 \NewDocumentCommand \symname { 0{} m }{
4237   \stex_symname_args:n { #1 }
4238   \stex_get_symbol:n { #2 }
4239   \str_set:Nx \l_tmpa_str {

```



```

4240 \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4241 }
4242 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4243
4244 \let\compemph_uri_prev:\compemph@uri
4245 \let\compemph@uri\symrefemph@uri
4246 \exp_args:Nnx \use:nn
4247 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmode*\fi{
4248 \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
4249 } }
4250 \let\compemph@uri\compemph_uri_prev:
4251 }
4252
4253 \NewDocumentCommand \Symname { 0{} m }{
4254 \stex_symname_args:n { #1 }
4255 \stex_get_symbol:n { #2 }
4256 \str_set:Nx \l_tmpa_str {
4257 \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4258 }
4259 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4260 \let\compemph_uri_prev:\compemph@uri
4261 \let\compemph@uri\symrefemph@uri
4262 \exp_args:Nnx \use:nn
4263 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmode*\fi{
4264 \exp_after:wN \stex_capitalize:n \l_tmpa_str
4265 \l__stex_terms_post_tl
4266 } }
4267 \let\compemph@uri\compemph_uri_prev:
4268 }

```

(End definition for `\symref` and `\symname`. These functions are documented on page 87.)

### 30.3 Notation Components

```

4269 <@@=stex_notationcomps>

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

4270 \cs_new_protected:Npn \_comp #1 {
4271 \str_if_empty:NF \STEXInternalCurrentSymbolStr {
4272 \stex_html_backend:TF {
4273 \stex_annotate:nnn { comp }{ \STEXInternalCurrentSymbolStr }{ #1 }
4274 }{
4275 \exp_args:Nnx \compemph@uri { #1 } { \STEXInternalCurrentSymbolStr }
4276 }
4277 }
4278 }
4279
4280 \cs_new_protected:Npn \_varcomp #1 {
4281 \str_if_empty:NF \STEXInternalCurrentSymbolStr {
4282 \stex_html_backend:TF {
4283 \stex_annotate:nnn { varcomp }{ \STEXInternalCurrentSymbolStr }{ #1 }
4284 }{
4285 \exp_args:Nnx \varemp@uri { #1 } { \STEXInternalCurrentSymbolStr }
4286 }

```

```

4287 }
4288 }
4289
4290 \def\comp{\_comp}
4291
4292 \cs_new_protected:Npn \compemph@uri #1 #2 {
4293   \compemph{ #1 }
4294 }
4295
4296
4297 \cs_new_protected:Npn \compemph #1 {
4298   #1
4299 }
4300
4301 \cs_new_protected:Npn \defemph@uri #1 #2 {
4302   \defemph{#1}
4303 }
4304
4305 \cs_new_protected:Npn \defemph #1 {
4306   \textbf{#1}
4307 }
4308
4309 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
4310   \symrefemph{#1}
4311 }
4312
4313 \cs_new_protected:Npn \symrefemph #1 {
4314   \emph{#1}
4315 }
4316
4317 \cs_new_protected:Npn \varemp@uri #1 #2 {
4318   \varemp{#1}
4319 }
4320
4321 \cs_new_protected:Npn \varemp #1 {
4322   #1
4323 }

```

(End definition for `\comp` and others. These functions are documented on page 88.)

## `\ellipses`

```

4324 \NewDocumentCommand \ellipses {} { \ldots }

```

(End definition for `\ellipses`. This function is documented on page 88.)

```

\parray
\prmatrix 4325 \bool_new:N \l_stex_inarray_bool
\parrayline 4326 \bool_set_false:N \l_stex_inarray_bool
\parraylineh 4327 \NewDocumentCommand \parray { m m } {
\parraycell 4328   \begingroup
4329   \bool_set_true:N \l_stex_inarray_bool
4330   \begin{array}{#1}
4331     #2
4332   \end{array}
4333   \endgroup

```

```

4334 }
4335
4336 \NewDocumentCommand \prmatrix { m } {
4337   \begingroup
4338   \bool_set_true:N \l_stex_inarray_bool
4339   \begin{matrix}
4340     #1
4341   \end{matrix}
4342   \endgroup
4343 }
4344
4345 \def \maybepline {
4346   \bool_if:NT \l_stex_inarray_bool {\hline}
4347 }
4348
4349 \def \parrayline #1 #2 {
4350   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
4351 }
4352
4353 \def \pmrow #1 { \parrayline{}{ #1 } }
4354
4355 \def \parraylineh #1 #2 {
4356   #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
4357 }
4358
4359 \def \parraycell #1 {
4360   #1 \bool_if:NT \l_stex_inarray_bool {&}
4361 }

```

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

## 30.4 Variables

```

4362 <@@=stex_variables>

```

\stex\_invoke\_variable:n Invokes a variable

```

4363 \cs_new_protected:Nn \stex_invoke_variable:n {
4364   \if_mode_math:
4365     \exp_after:wN \__stex_variables_invoke_math:n
4366   \else:
4367     \exp_after:wN \__stex_variables_invoke_text:n
4368   \fi: {#1}
4369 }
4370
4371 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
4372   \peek_charcode_remove:NTF ! {
4373     \__stex_variables_invoke_op_custom:nn {#1}
4374   }{
4375     \__stex_variables_invoke_custom:nn {#1}
4376   }
4377 }
4378
4379
4380 \cs_new_protected:Nn \__stex_variables_invoke_math:n {

```

```

4381 \peek_charcode_remove:NTF ! {
4382   \peek_charcode_remove:NTF ! {
4383     \peek_charcode:NTF [ {
4384       % TODO throw error
4385     }{
4386       \__stex_variables_invoke_op_custom:nn
4387     }
4388   }{
4389     \__stex_variables_invoke_op:n { #1 }
4390   }
4391 }{
4392   \peek_charcode_remove:NTF * {
4393     \__stex_variables_invoke_custom:nn { #1 }
4394   }{
4395     \__stex_variables_invoke_math_ii:n { #1 }
4396   }
4397 }
4398 }
4399
4400 \cs_new_protected:Nn \__stex_variables_invoke_op_custom:nn {
4401   \exp_args:Nnx \use:nn {
4402     \def\comp{\_varcomp}
4403     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4404     \bool_set_false:N \l_stex_allow_semantic_bool
4405     \stex_term_omv:nn {var://#1}{
4406       \comp{ #2 }
4407     }
4408   }{
4409     \stex_reset:N \comp
4410     \stex_reset:N \STEXInternalCurrentSymbolStr
4411     \bool_set_true:N \l_stex_allow_semantic_bool
4412   }
4413 }
4414
4415 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
4416   \cs_if_exist:cTF {
4417     stex_var_op_notation_ #1 _cs
4418   }{
4419     \exp_args:Nnx \use:nn {
4420       \def\comp{\_varcomp}
4421       \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4422       \stex_term_omv:nn { var://#1 }{
4423         \use:c{stex_var_op_notation_ #1 _cs }
4424       }
4425     }{
4426       \stex_reset:N \comp
4427       \stex_reset:N \STEXInternalCurrentSymbolStr
4428     }
4429   }{
4430     \int_compare:nNnTF {\prop_item:cn {l_stex_symdecl_var://#1_prop}{arity}} = 0{
4431       \__stex_variables_invoke_math_ii:n {#1}
4432     }{
4433       \msg_error:nxxx{stex}{error/noop}{variable~#1}{}
4434     }

```

```

4435 }
4436 }
4437
4438 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n #1 {
4439   \cs_if_exist:cTF {
4440     stex_var_notation_#1_cs
4441   }{
4442     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4443       \_stex_reset:N \comp
4444       \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
4445       \_stex_reset:N \STEXInternalCurrentSymbolStr
4446       \bool_set_true:N \l_stex_allow_semantic_bool
4447     }
4448     \def\comp{\_varcomp}
4449     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4450     \bool_set_false:N \l_stex_allow_semantic_bool
4451     \use:c{stex_var_notation_#1_cs}
4452   }{
4453     \msg_error:nnxx{stex}{error/nonotation}{variable-#1}{s}
4454   }
4455 }
4456
4457 \cs_new_protected:Nn \__stex_variables_invoke_custom:nn {
4458   \exp_args:Nnx \use:nn {
4459     \def\comp{\_varcomp}
4460     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4461     \prop_clear:N \l__stex_terms_custom_args_prop
4462     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
4463     \prop_get:cnN {
4464       l_stex_symdecl_var://#1 _prop
4465     }{ args } \l_tmpa_str
4466     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
4467     \tl_set:Nn \arg { \_stex_terms_arg: }
4468     \str_if_empty:NTF \l_tmpa_str {
4469       \_stex_term_omv:nn {var://#1}{\ignorespaces#2}
4470     }{
4471       \str_if_in:NnTF \l_tmpa_str b {
4472         \_stex_term_ombind:nnn {var://#1}{\ignorespaces#2}
4473       }{
4474         \str_if_in:NnTF \l_tmpa_str B {
4475           \_stex_term_ombind:nnn {var://#1}{\ignorespaces#2}
4476         }{
4477           \_stex_term_oma:nnn {var://#1}{\ignorespaces#2}
4478         }
4479       }
4480     }
4481     % TODO check that all arguments exist
4482   }{
4483     \_stex_reset:N \STEXInternalCurrentSymbolStr
4484     \_stex_reset:N \arg
4485     \_stex_reset:N \comp
4486     \_stex_reset:N \l__stex_terms_custom_args_prop
4487     %\bool_set_true:N \l_stex_allow_semantic_bool
4488   }

```

```
4489 }
```

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

## 30.5 Sequences

```
4490 <@@=stex_sequences>
4491
4492 \cs_new_protected:Nn \stex_invoke_sequence:n {
4493   \peek_charcode_remove:NTF ! {
4494     \stex_term_omv:nn {varseq://#1}{
4495       \exp_args:Nnx \use:nn {
4496         \def\comp{\_varcomp}
4497         \str_set:Nn \STEXInternalCurrentSymbolStr {varseq://#1}
4498         \prop_item:cn{l_stex_symdecl_varseq://#1_prop}{notation}
4499       }{
4500         \_stex_reset:N \comp
4501         \_stex_reset:N \STEXInternalCurrentSymbolStr
4502       }
4503     }
4504   }{
4505     \bool_set_false:N \l_stex_allow_semantic_bool
4506     \def\comp{\_varcomp}
4507     \str_set:Nn \STEXInternalCurrentSymbolStr {varseq://#1}
4508     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4509       \_stex_reset:N \comp
4510       \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
4511       \_stex_reset:N \STEXInternalCurrentSymbolStr
4512       \bool_set_true:N \l_stex_allow_semantic_bool
4513     }
4514     \use:c { stex_varseq_#1_cs }
4515   }
4516 }
4517 </package>
```

## Chapter 31

# STEX -Structural Features Implementation

```
4518 ⟨*package⟩
4519
4520 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4521
4522     Warnings and error messages
4523 \msg_new:nnn{stex}{error/copymodule/notallowed}{
4524     Symbol~#1~can~not~be~assigned~in~copymodule~#2
4525 }
4526 \msg_new:nnn{stex}{error/interpretmodule/nodfiniens}{
4527     Symbol~#1~not~assigned~in~interpretmodule~#2
4528 }
4529 \msg_new:nnn{stex}{error/unknownstructure}{
4530     No~structure~#1~found!
4531 }
4532
4533 \msg_new:nnn{stex}{error/unknownfield}{
4534     No~field~#1~in~instance~#2~found!\\#3
4535 }
4536
4537 \msg_new:nnn{stex}{error/keyval}{
4538     Invalid~key=value~pair:#1
4539 }
4540 \msg_new:nnn{stex}{error/instantiate/missing}{
4541     Assignments~missing~in~instantiate:~#1
4542 }
4543 \msg_new:nnn{stex}{error/incompatible}{
4544     Incompatible~signature:~#1~(#2)~and~#3~(#4)
4545 }
4546
```

## 31.1 Imports with modification

```

4547 <@@=stex_copymodule>
4548 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
4549   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4550     \tl_set:Nn \l_tmpa_tl { #1 }
4551     \__stex_copymodule_get_symbol_from_cs:
4552   }{
4553     % argument is a string
4554     % is it a command name?
4555     \cs_if_exist:cTF { #1 }{
4556       \cs_set_eq:Nc \l_tmpa_tl { #1 }
4557       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4558       \str_if_empty:NTF \l_tmpa_str {
4559         \exp_args:Nx \cs_if_eq:NNTF {
4560           \tl_head:N \l_tmpa_tl
4561         } \stex_invoke_symbol:n {
4562           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
4563         }{
4564           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4565         }
4566       } {
4567         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4568       }
4569     }{
4570       % argument is not a command name
4571       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4572       % \l_stex_all_symbols_seq
4573     }
4574   }
4575 }
4576
4577 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
4578   \str_set:Nn \l_tmpa_str { #1 }
4579   \bool_set_false:N \l_tmpa_bool
4580   \bool_if:NF \l_tmpa_bool {
4581     \tl_set:Nn \l_tmpa_tl {
4582       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
4583     }
4584     \str_set:Nn \l_tmpa_str { #1 }
4585     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4586     \seq_map_inline:Nn #2 {
4587       \str_set:Nn \l_tmpb_str { ##1 }
4588       \str_if_eq:eeT { \l_tmpa_str } {
4589         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4590       } {
4591         \seq_map_break:n {
4592           \tl_set:Nn \l_tmpa_tl {
4593             \str_set:Nn \l_stex_get_symbol_uri_str {
4594               ##1
4595             }
4596           }
4597         }
4598       }

```



```

4599     }
4600     \l_tmpa_tl
4601   }
4602 }
4603
4604 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
4605   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4606     { \tl_tail:N \l_tmpa_tl }
4607   \tl_if_single:NTF \l_tmpa_tl {
4608     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
4609       \exp_after:wN \str_set:Nn \exp_after:wN
4610         \l_stex_get_symbol_uri_str \l_tmpa_tl
4611       \__stex_copymodule_get_symbol_check:n { #1 }
4612     }{
4613       % TODO
4614       % tail is not a single group
4615     }
4616   }{
4617     % TODO
4618     % tail is not a single group
4619   }
4620 }
4621
4622 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
4623   \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
4624     \msg_error:nxxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
4625       :~\seq_use:Nn #1 {,~}
4626     }
4627   }
4628 }
4629
4630 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
4631   % import module
4632   \stex_import_module_uri:nn { #1 } { #2 }
4633   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
4634   \stex_import_require_module:nnnn
4635     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4636     { \l_stex_import_path_str } { \l_stex_import_name_str }
4637
4638   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
4639   \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
4640
4641   % fields
4642   \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
4643   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4644     \seq_map_inline:cn {c_stex_module_##1_constants}{
4645       \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
4646         ##1 ? #####1
4647       }
4648     }
4649   }
4650
4651   % setup prop
4652   \seq_clear:N \l_tmpa_seq

```

```

4653 \exp_args:Nn \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
4654   name      = \l_stex_current_copymodule_name_str ,
4655   module    = \l_stex_current_module_str ,
4656   from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
4657   includes  = \l_tmpa_seq %,
4658 % fields    = \l_tmpa_seq
4659 }
4660 \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
4661   as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
4662 \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_seq {,
4663 \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
4664
4665 \stex_if_do_html:T {
4666   \begin{stex_annotate_env} {#4} {
4667     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4668   }
4669   \stex_annotate_invisible:nnn{domain}{\l_stex_import_ns_str ?\l_stex_import_name_str}{\}
4670 }
4671 }
4672
4673 \cs_new_protected:Nn \stex_copymodule_end:n {
4674   % apply to every field
4675   \def \l_tmpa_cs ##1 ##2 {#1}
4676
4677   \tl_clear:N \__stex_copymodule_module_tl
4678   \tl_clear:N \__stex_copymodule_exec_tl
4679
4680   %\prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
4681   \seq_clear:N \__stex_copymodule_fields_seq
4682
4683   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4684     \seq_map_inline:cn {c_stex_module_##1_constants}{
4685
4686       \tl_clear:N \__stex_copymodule_curr_symbol_tl % <- wrap in current symbol html
4687       \l_tmpa_cs{##1}{####1}
4688
4689       \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
4690         \str_set_eq:Nc \__stex_copymodule_curr_name_str {l__stex_copymodule_copymodule_##1?####1_name_str}
4691         \stex_if_do_html:T {
4692           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4693             \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?####1_name_str}}
4694           }
4695         }
4696       }{
4697         \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str /
4698       }
4699
4700       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
4701       \prop_put:Nnx \l_tmpa_prop { name } \__stex_copymodule_curr_name_str
4702       \prop_put:Nnx \l_tmpa_prop { module } \l_stex_current_module_str
4703
4704       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4705         \stex_if_do_html:T {
4706           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {

```

```

4707         $\stex_annotate_invisible:nnn{definiens}{\exp_after:wN \exp_not:N\csname l__st
4708     }
4709 }
4710 \prop_put:Nnn \l_tmpa_prop { defined } { true }
4711 }
4712
4713 \stex_add_constant_to_current_module:n \__stex_copymodule_curr_name_str
4714 \tl_put_right:Nx \__stex_copymodule_module_tl {
4715     \seq_clear:c {l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_n
4716     \prop_set_from_keyval:cn {
4717         l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_name_str _prop
4718     }{
4719         \prop_to_keyval:N \l_tmpa_prop
4720     }
4721 }
4722
4723 \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
4724     \stex_if_do_html:T {
4725         \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4726             \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1
4727         }
4728     }
4729     \tl_put_right:Nx \__stex_copymodule_module_tl {
4730         \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
4731             \stex_invoke_symbol:n {
4732                 \l_stex_current_module_str ? \__stex_copymodule_curr_name_str
4733             }
4734         }
4735     }
4736 }
4737
4738 \seq_put_right:Nx \__stex_copymodule_fields_seq {\l_stex_current_module_str ? \__stex_
4739
4740 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4741     \stex_copy_notations:nn {\l_stex_current_module_str ? \__stex_copymodule_curr_name_s
4742 }
4743
4744 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4745     \stex_if_do_html:TF{
4746         \stex_annotate_invisible:nnn{assignment} {##1?####1} { \exp_after:wN \exp_not:n \e
4747     }{
4748         \exp_after:wN \exp_not:n \exp_after:wN {\__stex_copymodule_curr_symbol_tl}
4749     }
4750 }
4751 }
4752 }
4753
4754
4755 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \__stex_copymodule_fields_seq
4756 \tl_put_left:Nx \__stex_copymodule_module_tl {
4757     \prop_set_from_keyval:cn {
4758         l_stex_copymodule_ \l_stex_current_module_str?l_stex_current_copymodule_name_str _pro
4759 }{
4760     \prop_to_keyval:N \l_stex_current_copymodule_prop

```

```

4761     }
4762   }
4763
4764   \seq_gput_right:cx{c_stex_module_\l_stex_current_module_str _copymodules}{
4765     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4766   }
4767
4768   \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4769   \stex_debug:nn{copymodule}{result:\meaning \__stex_copymodule_module_tl}
4770   \stex_debug:nn{copymodule}{output:\meaning \__stex_copymodule_exec_tl}
4771
4772   \__stex_copymodule_exec_tl
4773   \stex_if_do_html:T {
4774     \end{stex_annotate_env}
4775   }
4776 }
4777
4778 \NewDocumentEnvironment {copymodule} { 0{} m m}{
4779   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ copymodule }
4780   \stex_deactivate_macro:Nn \symdecl {module~environments}
4781   \stex_deactivate_macro:Nn \symdef {module~environments}
4782   \stex_deactivate_macro:Nn \notation {module~environments}
4783   \stex_reactivate_macro:N \assign
4784   \stex_reactivate_macro:N \renamedekl
4785   \stex_reactivate_macro:N \donotcopy
4786   \stex_smsmode_do:
4787 }{
4788   \stex_copymodule_end:n {}
4789 }
4790
4791 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
4792   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ interpretmodule }
4793   \stex_deactivate_macro:Nn \symdecl {module~environments}
4794   \stex_deactivate_macro:Nn \symdef {module~environments}
4795   \stex_deactivate_macro:Nn \notation {module~environments}
4796   \stex_reactivate_macro:N \assign
4797   \stex_reactivate_macro:N \renamedekl
4798   \stex_reactivate_macro:N \donotcopy
4799   \stex_smsmode_do:
4800 }{
4801   \stex_copymodule_end:n {
4802     \tl_if_exist:cF {
4803       l__stex_copymodule_copymodule_##1?##2_def_tl
4804     }{
4805       \str_if_eq:eeF {
4806         \prop_item:cn{
4807           l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
4808         }{ true }{
4809           \msg_error:nxxx{stex}{error/interpretmodule/nodéfiniens}{
4810             ##1?##2
4811           }{\l_stex_current_copymodule_name_str}
4812         }
4813       }
4814     }

```

```

4815 }
4816
4817 \iffalse \begin{stex_annotate_env} \fi
4818 \NewDocumentEnvironment {realization} { 0 } { m } {
4819   \stex_copymodule_start:nnnn { #1 } { #2 } { #2 } { realize }
4820   \stex_deactivate_macro:Nn \symdecl {module~environments}
4821   \stex_deactivate_macro:Nn \symdef {module~environments}
4822   \stex_deactivate_macro:Nn \notation {module~environments}
4823   \stex_reactivate_macro:N \donotcopy
4824   \stex_reactivate_macro:N \assign
4825   \stex_smsmode_do:
4826 } {
4827   \stex_import_module_uri:nn { #1 } { #2 }
4828   \tl_clear:N \__stex_copymodule_exec_tl
4829   \tl_set:Nx \__stex_copymodule_module_tl {
4830     \stex_import_require_module:nnnn
4831     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4832     { \l_stex_import_path_str } { \l_stex_import_name_str }
4833   }
4834
4835   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4836     \seq_map_inline:cn {c_stex_module_##1_constants}{
4837       \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str / #1 }
4838       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?###1_def_tl}{
4839         \stex_if_do_html:T {
4840           \tl_put_right:Nx \__stex_copymodule_exec_tl {
4841             \stex_annotate_invisible:nnn{assignment} {##1?###1} {
4842               $\stex_annotate_invisible:nnn{definien}{}{\exp_after:wN \exp_not:N\csname l__
4843             }
4844           }
4845         }
4846         \tl_put_right:Nx \__stex_copymodule_module_tl {
4847           \prop_put:cnn {l_stex_symdecl_##1?###1_prop}{ defined }{ true }
4848         }
4849       }
4850     }
4851   }
4852   \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4853
4854   \__stex_copymodule_exec_tl
4855   \stex_if_do_html:T {\end{stex_annotate_env}}
4856 }
4857
4858 \NewDocumentCommand \donotcopy { m } {
4859   \str_clear:N \l_stex_import_name_str
4860   \str_set:Nn \l_tmpa_str { #1 }
4861   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4862   \seq_map_inline:Nn \l_stex_all_modules_seq {
4863     \str_set:Nn \l_tmpb_str { ##1 }
4864     \str_if_eq:eeT { \l_tmpa_str } {
4865       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4866     } {
4867       \seq_map_break:n {
4868         \stex_if_do_html:T {

```

```

4869         \stex_if_smsmode:F {
4870             \stex_annotate_invisible:nnn{donotcopy}{##1}{
4871                 \stex_annotate:nnn{domain}{##1}{}}
4872         }
4873     }
4874 }
4875 \str_set_eq:NN \l_stex_import_name_str \l_tmpb_str
4876 }
4877 }
4878 \seq_map_inline:cn {c_stex_module_###1_copymodules}{
4879     \str_set:Nn \l_tmpb_str { #####1 }
4880     \str_if_eq:eeT { \l_tmpa_str } {
4881         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4882     } {
4883         \seq_map_break:n {\seq_map_break:n {
4884             \stex_if_do_html:T {
4885                 \stex_if_smsmode:F {
4886                     \stex_annotate_invisible:nnn{donotcopy}{#####1}{
4887                         \stex_annotate:nnn{domain}{
4888                             \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4889                         }{}
4890                     }
4891                 }
4892             }
4893             \str_set:Nx \l_stex_import_name_str {
4894                 \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4895             }
4896         }}
4897     }
4898 }
4899 }
4900 \str_if_empty:NTF \l_stex_import_name_str {
4901     % TODO throw error
4902 }{
4903     \stex_collect_imports:n {\l_stex_import_name_str }
4904     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4905         \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
4906         \seq_map_inline:cn {c_stex_module_###1_constants}{
4907             \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? #####1 }
4908             \bool_lazy_any:nT {
4909                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_name_str}}
4910                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_macroname_str}}
4911                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_def_tl}}
4912             }{
4913                 % TODO throw error
4914             }
4915         }
4916     }
4917     \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
4918     \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_name_str }
4919     \prop_put:Nno \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
4920 }
4921 \stex_smsmode_do:
4922 }

```

```

4923
4924 \NewDocumentCommand \assign { m m }{
4925   \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
4926   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
4927   \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_def_tl}{#2}
4928   \stex_smsmode_do:
4929 }
4930
4931 \keys_define:nn { stex / renamedecl } {
4932   name          .str_set_x:N = \l_stex_renamedecl_name_str
4933 }
4934 \cs_new_protected:Nn \__stex_copymodule_renamedecl_args:n {
4935   \str_clear:N \l_stex_renamedecl_name_str
4936   \keys_set:nn { stex / renamedecl } { #1 }
4937 }
4938
4939 \NewDocumentCommand \renamedecl { O{} m m }{
4940   \__stex_copymodule_renamedecl_args:n { #1 }
4941   \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
4942   \stex_debug:nn{renamedecl}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
4943   \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_macroname_str}{#3}
4944   \str_if_empty:NTF \l_stex_renamedecl_name_str {
4945     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4946       \l_stex_get_symbol_uri_str
4947     } }
4948   } {
4949     \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_name_str}{\l_stex
4950       \stex_debug:nn{renamedecl}{@~\l_stex_current_module_str ? \l_stex_renamedecl_name_str}
4951       \prop_set_eq:cc {l_stex_symdecl_
4952         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4953         _prop
4954       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str_prop}
4955       \seq_set_eq:cc {l_stex_symdecl_
4956         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4957         _notations
4958       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str_notations}
4959       \prop_put:cnx {l_stex_symdecl_
4960         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4961         _prop
4962       }{ name }{ \l_stex_renamedecl_name_str }
4963       \prop_put:cnx {l_stex_symdecl_
4964         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4965         _prop
4966       }{ module }{ \l_stex_current_module_str }
4967       \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
4968         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4969       }
4970       \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4971         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4972       } }
4973     }
4974   \stex_smsmode_do:
4975 }
4976

```

```

4977 \stex_deactivate_macro:Nn \assign {copymodules}
4978 \stex_deactivate_macro:Nn \renamedekl {copymodules}
4979 \stex_deactivate_macro:Nn \donotcopy {copymodules}
4980
4981

```

## 31.2 The feature environment

`structural@feature (env.)`

```

4982 <@@=stex_features>
4983
4984 \NewDocumentEnvironment{structural_feature_module}{ m m m }{
4985   \stex_if_in_module:F {
4986     \msg_set:nnn{stex}{error/nomodule}{
4987       Structural~Feature~has~to~occur~in~a~module:\\
4988       Feature~#2~of~type~#1\\
4989       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
4990     }
4991     \msg_error:nn{stex}{error/nomodule}
4992   }
4993
4994   \str_set_eq:NN \l_stex_feature_parent_str \l_stex_current_module_str
4995
4996   \stex_module_setup:nn{meta=NONE}{#2 - #1}
4997
4998   \stex_if_do_html:T {
4999     \begin{stex_annotate_env}{ feature:#1 }{\l_stex_feature_parent_str ? #2 - #1}
5000     \stex_annotate_invisible:nnn{header}{}{ #3 }
5001   }
5002   }{
5003     \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
5004     \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
5005     \stex_debug:nn{features}{
5006       Feature: \l_stex_last_feature_str
5007     }
5008     \stex_if_do_html:T {
5009       \end{stex_annotate_env}
5010     }
5011   }

```

## 31.3 Structure

`structure (env.)`

```

5012 <@@=stex_structures>
5013 \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
5014   \prop_if_exist:cF {c_stex_module_ \l_stex_current_module_str _structures}{
5015     \prop_new:c {c_stex_module_ \l_stex_current_module_str _structures}
5016   }
5017   \prop_gput:cxn{c_stex_module_ \l_stex_current_module_str _structures}
5018   {#1}{#2}
5019 }
5020

```



```

5021 \keys_define:nn { stex / features / structure } {
5022   name          .str_set_x:N = \l__stex_structures_name_str ,
5023 }
5024
5025 \cs_new_protected:Nn \__stex_structures_structure_args:n {
5026   \str_clear:N \l__stex_structures_name_str
5027   \keys_set:nn { stex / features / structure } { #1 }
5028 }
5029
5030 \NewDocumentEnvironment{mathstructure}{m O{}}{
5031   \__stex_structures_structure_args:n { #2 }
5032   \str_if_empty:NT \l__stex_structures_name_str {
5033     \str_set:Nx \l__stex_structures_name_str { #1 }
5034   }
5035   \stex_suppress_html:n {
5036     \bool_set_true:N \l_stex_symdecl_make_macro_bool
5037     \exp_args:Nx \stex_symdecl_do:nn {
5038       name = \l__stex_structures_name_str ,
5039       def = {\STEXsymbol{module-type}}{
5040         \STEXInternalTermMathOMSiiii {
5041           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
5042             { ns } ?
5043           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
5044             { name } / \l__stex_structures_name_str - structure
5045         }{}{0}{}
5046       }}
5047     }{ #1 }
5048   }
5049   \exp_args:Nnnx
5050   \begin{structural_feature_module}{ structure }
5051     { \l__stex_structures_name_str }{}
5052   \stex_smsmode_do:
5053 }{
5054   \end{structural_feature_module}
5055   \_stex_reset_up_to_module:n \l_stex_last_feature_str
5056   \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
5057   \seq_clear:N \l_tmpa_seq
5058   \seq_map_inline:Nn \l_stex_collect_imports_seq {
5059     \seq_map_inline:cn{c_stex_module_##1_constants}{
5060       \seq_put_right:Nn \l_tmpa_seq { ##1 ? ####1 }
5061     }
5062   }
5063   \exp_args:Nnno
5064   \prop_gput:cnn {c_stex_module_ \l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
5065   \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
5066   \stex_add_structure_to_current_module:nn
5067     \l__stex_structures_name_str
5068     \l_stex_last_feature_str
5069
5070   \stex_execute_in_module:x {
5071     \tl_set:cn { #1 }{
5072       \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures
5073     }
5074   }

```

```

5075 }
5076
5077 \cs_new:Nn \stex_invoke_structure:nn {
5078   \stex_invoke_symbol:n { #1?#2 }
5079 }
5080
5081 \cs_new_protected:Nn \stex_get_structure:n {
5082   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
5083     \tl_set:Nn \l_tmpa_tl { #1 }
5084     \__stex_structures_get_from_cs:
5085   }{
5086     \cs_if_exist:cTF { #1 }{
5087       \cs_set_eq:Nc \l_tmpa_cs { #1 }
5088       \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
5089       \str_if_empty:NTF \l_tmpa_str {
5090         \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs } \stex_invoke_structure:nn {
5091           \__stex_structures_get_from_cs:
5092         }{
5093           \__stex_structures_get_from_string:n { #1 }
5094         }
5095       }{
5096         \__stex_structures_get_from_string:n { #1 }
5097       }
5098     }{
5099       \__stex_structures_get_from_string:n { #1 }
5100     }
5101   }
5102 }
5103
5104 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
5105   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
5106     { \tl_tail:N \l_tmpa_tl }
5107   \str_set:Nx \l_tmpa_str {
5108     \exp_after:wN \use_i:nn \l_tmpa_tl
5109   }
5110   \str_set:Nx \l_tmpb_str {
5111     \exp_after:wN \use_ii:nn \l_tmpa_tl
5112   }
5113   \str_set:Nx \l_stex_get_structure_str {
5114     \l_tmpa_str ? \l_tmpb_str
5115   }
5116   \str_set:Nx \l_stex_get_structure_module_str {
5117     \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
5118   }
5119 }
5120
5121 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
5122   \tl_set:Nn \l_tmpa_tl {
5123     \msg_error:nnn{stex}{error/unknownstructure}{#1}
5124   }
5125   \str_set:Nn \l_tmpa_str { #1 }
5126   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
5127
5128   \seq_map_inline:Nn \l_stex_all_modules_seq {

```

```

5129 \prop_if_exist:cT {c_stex_module_##1_structures} {
5130 \prop_map_inline:cn {c_stex_module_##1_structures} {
5131 \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
5132 \prop_map_break:n{\seq_map_break:n{
5133 \tl_set:Nn \l_tmpa_tl {
5134 \str_set:Nn \l_stex_get_structure_str {##1?####1}
5135 \str_set:Nn \l_stex_get_structure_module_str {####2}
5136 }
5137 }}
5138 }
5139 }
5140 }
5141 }
5142 \l_tmpa_tl
5143 }

```

**\instantiate**

```

5144
5145 \keys_define:nn { stex / instantiate } {
5146 name .str_set_x:N = \l__stex_structures_name_str
5147 }
5148 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
5149 \str_clear:N \l__stex_structures_name_str
5150 \keys_set:nn { stex / instantiate } { #1 }
5151 }
5152
5153 \NewDocumentCommand \instantiate {m O{} m m O{}}{
5154 \beginingroup
5155 \stex_get_structure:n {#3}
5156 \__stex_structures_instantiate_args:n { #2 }
5157 \str_if_empty:NT \l__stex_structures_name_str {
5158 \str_set:Nn \l__stex_structures_name_str { #1 }
5159 }
5160 \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5161 \seq_clear:N \l__stex_structures_fields_seq
5162 \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
5163 \seq_map_inline:Nn \l_stex_collect_imports_seq {
5164 \seq_map_inline:cn {c_stex_module_##1_constants}{
5165 \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
5166 }
5167 }
5168
5169 \tl_if_empty:nF{#5}{
5170 \seq_set_split:Nnn \l_tmpa_seq , {#5}
5171 \prop_clear:N \l_tmpa_prop
5172 \seq_map_inline:Nn \l_tmpa_seq {
5173 \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
5174 \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
5175 \msg_error:nnn{stex}{error/keyval}{##1}
5176 }
5177 \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
5178 \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
5179 \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
5180 \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}

```

```

5181     \exp_args:Nxx \str_if_eq:nnF
5182     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5183     {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
5184     \msg_error:nnxxxx{stex}{error/incompatible}
5185     {\l__stex_structures_dom_str}
5186     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5187     {\l_stex_get_symbol_uri_str}
5188     {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
5189   }
5190   \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
5191 }
5192 }
5193
5194 \seq_map_inline:Nn \l__stex_structures_fields_seq {
5195   \str_set:Nx \l_tmpa_str {field:\l__stex_structures_name_str . \prop_item:cn {l_stex_sy
5196   \stex_debug:nn{instantiate}{Field~\l_tmpa_str :~##1}
5197
5198   \stex_add_constant_to_current_module:n {\l_tmpa_str}
5199   \stex_execute_in_module:x {
5200     \prop_set_from_keyval:cn { l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _p
5201     name = \l_tmpa_str ,
5202     args = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
5203     arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
5204     assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs} ,
5205     argnames = {\prop_item:cn {l_stex_symdecl_##1_prop}{argnames}}
5206   }
5207   \seq_clear:c {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notations}
5208 }
5209
5210 \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
5211   \stex_find_notation:nn{##1}{}
5212   \stex_execute_in_module:x {
5213     \seq_put_right:cn {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notation
5214   }
5215
5216   \stex_copy_control_sequence_ii:ccN
5217   {stex_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_notation_
5218   {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5219   \l_tmpa_tl
5220   \exp_args:No \stex_execute_in_module:n \l_tmpa_tl
5221
5222
5223   \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
5224     \tl_set_eq:Nc \l_tmpa_cs {stex_op_notation_##1\c_hash_str \l_stex_notation_variant
5225     \stex_execute_in_module:x {
5226       \tl_set:cn
5227       {stex_op_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_not
5228       { \exp_args:No \exp_not:n \l_tmpa_cs}
5229     }
5230   }
5231
5232 }
5233
5234 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\l_stex_cur

```

```

5235 }
5236
5237 \stex_execute_in_module:x {
5238   \prop_set_from_keyval:cn {l_stex_instance_ \l_stex_current_module_str? \l__stex_structur
5239     domain = \l_stex_get_structure_module_str ,
5240     \prop_to_keyval:N \l_tmpa_prop
5241   }
5242   \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str? \l__stex_structur
5243 }
5244 \stex_debug:nn{instantiate}{
5245   Instance~ \l_stex_current_module_str? \l__stex_structures_name_str \
5246   \prop_to_keyval:N \l_tmpa_prop
5247 }
5248 \exp_args:Nxx \stex_symdecl_do:nn {
5249   type={\STEXsymbol{module-type}{
5250     \STEXInternalTermMathOMSiiii {
5251       \l_stex_get_structure_module_str
5252     }{}{0}{}}
5253   }}
5254 }{\l__stex_structures_name_str}
5255 % {
5256   \str_set:Nx \l_stex_get_symbol_uri_str {\l_stex_current_module_str? \l__stex_structures
5257   \tl_set:Nn \l_stex_notation_after_do_tl {\_stex_notation_final:}
5258   \stex_notation_do:nnnnn{}{}{}{}{\comp{#4}}
5259 % }
5260 %\exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}
5261 \endgroup
5262 \stex_smsmode_do:\ignorespacesandpars
5263 }
5264
5265 \cs_new_protected:Nn \stex_symbol_or_var:n {
5266   \cs_if_exist:cTF{#1}{
5267     \cs_set_eq:Nc \l_tmpa_tl { #1 }
5268     \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
5269     \str_if_empty:NTF \l_tmpa_str {
5270       \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
5271       \stex_invoke_variable:n {
5272         \bool_set_true:N \l_stex_symbol_or_var_bool
5273         \bool_set_false:N \l_stex_instance_or_symbol_bool
5274         \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
5275         \tl_set:Nx \l_tmpa_tl {\exp_after:wN \use:n \l_tmpa_tl}
5276         \str_set:Nx \l_stex_get_symbol_uri_str {
5277           \exp_after:wN \use:n \l_tmpa_tl
5278         }
5279       }{ % TODO \stex_invoke_varinstance:n
5280         \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl } \stex_invoke_varinstance:n {
5281           \bool_set_true:N \l_stex_symbol_or_var_bool
5282           \bool_set_true:N \l_stex_instance_or_symbol_bool
5283           \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
5284           \tl_set:Nx \l_tmpa_tl {\exp_after:wN \use:n \l_tmpa_tl}
5285           \str_set:Nx \l_stex_get_symbol_uri_str {
5286             \exp_after:wN \use:n \l_tmpa_tl
5287           }
5288         }{

```

```

5289         \bool_set_false:N \l_stex_symbol_or_var_bool
5290         \stex_get_symbol:n{#1}
5291     }
5292 }
5293 }{
5294     \__stex_structures_symbolorvar_from_string:n{ #1 }
5295 }
5296 }{
5297     \__stex_structures_symbolorvar_from_string:n{ #1 }
5298 }
5299 }
5300
5301 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
5302     \prop_if_exist:cTF {l_stex_symdecl_var://#1 _prop}{
5303         \bool_set_true:N \l_stex_symbol_or_var_bool
5304         \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
5305     }{
5306         \bool_set_false:N \l_stex_symbol_or_var_bool
5307         \stex_get_symbol:n{#1}
5308     }
5309 }
5310
5311 \keys_define:nn { stex / varinstantiate } {
5312     name .str_set_x:N = \l__stex_structures_name_str,
5313     bind .choices:nn =
5314         {forall,exists}
5315         {\str_set:Nx \l__stex_structures_bind_str {\l_keys_choice_tl}}
5316 }
5317 }
5318 \cs_new_protected:Nn \__stex_structures_varinstantiate_args:n {
5319     \str_clear:N \l__stex_structures_name_str
5320     \str_clear:N \l__stex_structures_bind_str
5321     \keys_set:nn { stex / varinstantiate } { #1 }
5322 }
5323
5324 \NewDocumentCommand \varinstantiate {m O{} m m O{}}{
5325     \beginingroup
5326         \stex_get_structure:n {#3}
5327         \__stex_structures_varinstantiate_args:n { #2 }
5328         \str_if_empty:NT \l__stex_structures_name_str {
5329             \str_set:Nn \l__stex_structures_name_str { #1 }
5330         }
5331         \stex_if_do_html:TF{
5332             \stex_annotate:nnn{varinstance}{\l__stex_structures_name_str}
5333         }{\use:n}
5334         {
5335             \stex_if_do_html:T{
5336                 \stex_annotate_invisible:nnn{domain}{\l_stex_get_structure_module_str}{}
5337             }
5338             \seq_clear:N \l__stex_structures_fields_seq
5339             \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
5340             \seq_map_inline:Nn \l_stex_collect_imports_seq {
5341                 \seq_map_inline:cn {c_stex_module_##1_constants}{
5342                     \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? #####1 }

```

```

5343     }
5344 }
5345 \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5346 \prop_clear:N \l_tmpa_prop
5347 \tl_if_empty:nF {#5} {
5348   \seq_set_split:Nnn \l_tmpa_seq , {#5}
5349   \seq_map_inline:Nn \l_tmpa_seq {
5350     \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
5351     \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
5352       \msg_error:nnn{stex}{error/keyval}{##1}
5353     }
5354     \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_structures_dom_str
5355     \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
5356     \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_in_seq:nn
5357     \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
5358     \stex_if_do_html:T{
5359       \stex_annotate:nnn{assign}{\l__stex_structures_dom_str,
5360         \bool_if:NTF\l_stex_symbol_or_var_bool{var://}{\l_stex_get_symbol_uri_str}{}}
5361     }
5362     \bool_if:NTF \l_stex_symbol_or_var_bool {
5363       \exp_args:Nxx \str_if_eq:nnF
5364         {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
5365         {\prop_item:cn{l_stex_symdecl_var://\l_stex_get_symbol_uri_str _prop}{args}}{
5366         \msg_error:nnxxxx{stex}{error/incompatible}
5367         {\l__stex_structures_dom_str}
5368         {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
5369         {\l_stex_get_symbol_uri_str}
5370         {\prop_item:cn{l_stex_symdecl_var://\l_stex_get_symbol_uri_str _prop}{args}}
5371       }
5372       \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n {
5373     }}{
5374       \exp_args:Nxx \str_if_eq:nnF
5375         {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
5376         {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}{
5377         \msg_error:nnxxxx{stex}{error/incompatible}
5378         {\l__stex_structures_dom_str}
5379         {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
5380         {\l_stex_get_symbol_uri_str}
5381         {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}
5382       }
5383       \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n {
5384     }}
5385     }
5386   }
5387   \tl_gclear:N \g__stex_structures_aftergroup_tl
5388   \seq_map_inline:Nn \l__stex_structures_fields_seq {
5389     \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdecl\l__stex_structures_fields_seq #1} \l_tmpa_str :~##1}
5390     \stex_debug:nn{varinstantiate}{Field~\l_tmpa_str :~##1}
5391     \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
5392       \stex_find_notation:nn{##1}{
5393         \cs_gset_eq:cc{g__stex_structures_tmpa_\l_tmpa_str _cs}
5394           {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5395         \stex_debug:nn{varinstantiate}{Notation:~\cs_meaning:c{g__stex_structures_tmpa_\l_tmpa_str _cs}}
5396         \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{

```

```

5397         \cs_gset_eq:cc {g__stex_structures_tmpa_op_\l_tmpa_str_cs}
5398         {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str_cs}
5399         \stex_debug:nn{varinstantiate}{Operator~Notation:~\cs_meaning:c{g__stex_struct
5400     }
5401 }
5402
5403 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
5404     \prop_set_from_keyval:cn { \l_stex_symdecl_ var://\l_tmpa_str_prop}{
5405         name = \l_tmpa_str ,
5406         args = \prop_item:cn { \l_stex_symdecl_##1_prop}{args} ,
5407         arity = \prop_item:cn { \l_stex_symdecl_##1_prop}{arity} ,
5408         assocs = \prop_item:cn { \l_stex_symdecl_##1_prop}{assocs} ,
5409         argnames = {\prop_item:cn { \l_stex_symdecl_##1_prop}{argnames}} ,
5410     }
5411     \cs_set_eq:cc {stex_var_notation_\l_tmpa_str_cs}
5412     {g__stex_structures_tmpa_\l_tmpa_str_cs}
5413     \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str_cs}
5414     {g__stex_structures_tmpa_op_\l_tmpa_str_cs}
5415 }
5416 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn { \l_stex_symdecl_##1_prop}{name}}{\stex_inv
5417 }
5418 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
5419     \prop_set_from_keyval:cn { \l_stex_varinstance_\l__stex_structures_name_str_prop }{
5420         domain = \l_stex_get_structure_module_str ,
5421         \prop_to_keyval:N \l_tmpa_prop
5422     }
5423     \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
5424     \tl_set:cn { \l_stex_varinstance_\l__stex_structures_name_str_op_tl}{
5425         \exp_args:Nnx \exp_not:N \use:nn {
5426             \str_set:Nn \exp_not:N \STEXInternalCurrentSymbolStr {var://\l__stex_structures_
5427             \_stex_term_omv:nn {var://\l__stex_structures_name_str}{
5428                 \exp_not:n{
5429                     \_varcomp{#4}
5430                 }
5431             }
5432         }}{
5433             \exp_not:n{\_stex_reset:N \STEXInternalCurrentSymbolStr}
5434         }
5435     }
5436 }
5437 }
5438 \stex_debug:nn{varinstantiate}{\expandafter\detokenize\expandafter{g__stex_structures_a
5439 \aftergroup\g__stex_structures_aftergroup_tl
5440 \endgroup
5441 \stex_smsmode_do:\ignorespacesandpars
5442 }
5443
5444 \cs_new_protected:Nn \stex_invoke_instance:n {
5445     \peek_charcode_remove:NTF ! {
5446         \stex_invoke_symbol:n{#1}
5447     }{
5448         \_stex_invoke_instance:nn {#1}
5449     }
5450 }

```



```

5451
5452
5453 \cs_new_protected:Nn \stex_invoke_varinstance:n {
5454   \peek_charcode_remove:NTF ! {
5455     \exp_args:Nnx \use:nn {
5456       \def\comp{\_varcomp}
5457       \use:c{l_stex_varinstance_#1_op_tl}
5458     }{
5459       \_stex_reset:N \comp
5460     }
5461   }{
5462     \_stex_invoke_varinstance:nn {#1}
5463   }
5464 }
5465
5466 \cs_new_protected:Nn \stex_invoke_instance:nn {
5467   \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
5468     \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
5469   }{
5470     \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
5471     \msg_error:nnxxx{stex}{error/unknownfield}{#2}{#1}{
5472       \prop_to_keyval:N \l_tmpa_prop
5473     }
5474   }
5475 }
5476
5477 \cs_new_protected:Nn \stex_invoke_varinstance:nn {
5478   \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
5479     \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
5480     \l_tmpa_tl
5481   }{
5482     \msg_error:nnnnn{stex}{error/unknownfield}{#2}{#1}{}
5483   }
5484 }

```

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

\stex\_invoke\_structure:nnn

```

5485 % #1: URI of the instance
5486 % #2: URI of the instantiated module
5487 \cs_new_protected:Nn \stex_invoke_structure:nnn {
5488   \tl_if_empty:nTF{ #3 }{
5489     \prop_set_eq:Nc \l__stex_structures_structure_prop {
5490       c_stex_feature_ #2 _prop
5491     }
5492     \tl_clear:N \l_tmpa_tl
5493     \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
5494     \seq_map_inline:Nn \l_tmpa_seq {
5495       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
5496       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
5497       \cs_if_exist:cT {
5498         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str_cs
5499       }{
5500         \tl_if_empty:NF \l_tmpa_tl {

```

```

5501         \tl_put_right:Nn \l_tmpa_tl {,}
5502     }
5503     \tl_put_right:Nx \l_tmpa_tl {
5504         \stex_invoke_symbol:n {#1/\l_tmpa_str}!
5505     }
5506 }
5507 }
5508 \exp_args:No \mathstruct \l_tmpa_tl
5509 }{
5510     \stex_invoke_symbol:n{#1/#3}
5511 }
5512 }

```

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

```

5513 </package>

```

## Chapter 32

# STEX -Statements Implementation

```
5514 <*package>
5515
5516 %%%%%%%%%%% features.dtx %%%%%%%%%%%
5517
5518 <@@=stex_statements>
    Warnings and error messages
5519

\titleemph
5520 \def\titleemph#1{\textbf{#1}}

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

### 32.1 Definitions

#### definiendum

```
5521 \keys_define:nn {stex / definiendum }{
5522   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
5523   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
5524   root     .str_set_x:N    = \l__stex_statements_definiendum_root_str,
5525   gfa      .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
5526 }
5527 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
5528   \str_clear:N \l__stex_statements_definiendum_root_str
5529   \tl_clear:N \l__stex_statements_definiendum_post_tl
5530   \str_clear:N \l__stex_statements_definiendum_gfa_str
5531   \keys_set:nn { stex / definiendum }{ #1 }
5532 }
5533 \NewDocumentCommand \definiendum { O{} m m } {
5534   \__stex_statements_definiendum_args:n { #1 }
5535   \stex_get_symbol:n { #2 }
5536   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5537   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
5538     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```

5539     \tl_set:Nn \l_tmpa_tl { #3 }
5540   } {
5541     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
5542     \tl_set:Nn \l_tmpa_tl {
5543       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
5544     }
5545   }
5546 } {
5547   \tl_set:Nn \l_tmpa_tl { #3 }
5548 }
5549
5550 % TODO root
5551 \stex_html_backend:TF {
5552   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
5553 } {
5554   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
5555 }
5556 }
5557 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

#### definame

```

5558
5559 \NewDocumentCommand \definame { 0{ } m } {
5560   \__stex_statements_definiendum_args:n { #1 }
5561   % TODO: root
5562   \stex_get_symbol:n { #2 }
5563   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5564   \str_set:Nx \l_tmpa_str {
5565     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5566   }
5567   \str_replace_all:Nnn \l_tmpa_str {-} {~}
5568   \stex_html_backend:TF {
5569     \stex_if_do_html:T {
5570       \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5571         \l_tmpa_str\l__stex_statements_definiendum_post_tl
5572       }
5573     }
5574   } {
5575     \exp_args:Nnx \defemph@uri {
5576       \l_tmpa_str\l__stex_statements_definiendum_post_tl
5577     } { \l_stex_get_symbol_uri_str }
5578   }
5579 }
5580 \stex_deactivate_macro:Nn \definame {definition~environments}
5581
5582 \NewDocumentCommand \Definame { 0{ } m } {
5583   \__stex_statements_definiendum_args:n { #1 }
5584   \stex_get_symbol:n { #2 }
5585   \str_set:Nx \l_tmpa_str {
5586     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5587   }
5588   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}

```

```

5589 \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5590 \stex_html_backend:TF {
5591   \stex_if_do_html:T {
5592     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5593       \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5594     }
5595   }
5596 } {
5597   \exp_args:Nnx \defemph@uri {
5598     \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5599   } { \l_stex_get_symbol_uri_str }
5600 }
5601 }
5602 \stex_deactivate_macro:Nn \Definame {definition-environments}
5603
5604 \NewDocumentCommand \premise { m }{
5605   \noindent\stex_annotate:nnn{ premise }{}{\ignorespaces #1 }
5606 }
5607 \NewDocumentCommand \conclusion { m }{
5608   \noindent\stex_annotate:nnn{ conclusion }{}{\ignorespaces #1 }
5609 }
5610 \NewDocumentCommand \definiens { 0{} m }{
5611   \str_clear:N \l_stex_get_symbol_uri_str
5612   \tl_if_empty:nF {#1} {
5613     \stex_get_symbol:n { #1 }
5614   }
5615   \str_if_empty:NT \l_stex_get_symbol_uri_str {
5616     \int_compare:nNnTF {\clist_count:N \l__stex_statements_sdefinition_for_clist} = 1 {
5617       \str_set:Nx \l_stex_get_symbol_uri_str {\clist_item:Nn \l__stex_statements_sdefinition
5618     }{
5619       % TODO throw error
5620     }
5621   }
5622   \str_if_eq:eeT {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{module}}
5623   {\l_stex_current_module_str}{
5624     \str_if_eq:eeF {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defin
5625   }{true}{
5626     \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5627     \exp_args:Nx \stex_add_to_current_module:n {
5628       \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5629     }
5630   }
5631 }
5632 \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
5633 }
5634
5635 \NewDocumentCommand \varbindforall {m}{
5636   \stex_symbol_or_var:n {#1}
5637   \bool_if:NTF\l_stex_symbol_or_var_bool{
5638     \stex_if_do_html:T {
5639       \stex_annotate_invisible:nnn {bindtype}{forall,\l_stex_get_symbol_uri_str}{}
5640     }
5641   }{
5642     % todo throw error

```

```

5643 }
5644 }
5645
5646 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
5647 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
5648 \stex_deactivate_macro:Nn \definiens {definition~environments}
5649 \stex_deactivate_macro:Nn \varbindforall {definition~or~assertion~environments}
5650

```

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

sdefinition (env.)

```

5651
5652 \keys_define:nn {stex / sdefinition }{
5653   type      .str_set_x:N = \sdefinitiontype,
5654   id        .str_set_x:N = \sdefinitionid,
5655   name      .str_set_x:N = \sdefinitionname,
5656   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5657   title     .tl_set:N     = \sdefinitiontitle
5658 }
5659 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
5660   \str_clear:N \sdefinitiontype
5661   \str_clear:N \sdefinitionid
5662   \str_clear:N \sdefinitionname
5663   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5664   \tl_clear:N \sdefinitiontitle
5665   \keys_set:nn { stex / sdefinition }{ #1 }
5666 }
5667
5668 \NewDocumentEnvironment{sdefinition}{0{}}{
5669   \__stex_statements_sdefinition_args:n{ #1 }
5670   \stex_reactivate_macro:N \definiendum
5671   \stex_reactivate_macro:N \definame
5672   \stex_reactivate_macro:N \Definame
5673   \stex_reactivate_macro:N \premise
5674   \stex_reactivate_macro:N \definiens
5675   \stex_reactivate_macro:N \varbindforall
5676   \stex_if_smsmode:F{
5677     \seq_clear:N \l_tmpb_seq
5678     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5679       \tl_if_empty:nF{ ##1 }{
5680         \stex_get_symbol:n { ##1 }
5681         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5682           \l_stex_get_symbol_uri_str
5683         }
5684       }
5685     }
5686     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5687     \exp_args:Nnnx
5688     \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
5689     \str_if_empty:NF \sdefinitiontype {
5690       \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5691     }
5692     \str_if_empty:NF \sdefinitionname {

```

```

5693     \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5694   }
5695   \clist_set:No \l_tmpa_clist \sdefinitiontype
5696   \tl_clear:N \l_tmpa_tl
5697   \clist_map_inline:Nn \l_tmpa_clist {
5698     \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
5699       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
5700     }
5701   }
5702   \tl_if_empty:NTF \l_tmpa_tl {
5703     \__stex_statements_sdefinition_start:
5704   }{
5705     \l_tmpa_tl
5706   }
5707 }
5708 \stex_ref_new_doc_target:n \sdefinitionid
5709 \stex_smsmode_do:
5710 }{
5711   \stex_suppress_html:n {
5712     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5713   }
5714   \stex_if_smsmode:F {
5715     \clist_set:No \l_tmpa_clist \sdefinitiontype
5716     \tl_clear:N \l_tmpa_tl
5717     \clist_map_inline:Nn \l_tmpa_clist {
5718       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
5719         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
5720       }
5721     }
5722     \tl_if_empty:NTF \l_tmpa_tl {
5723       \__stex_statements_sdefinition_end:
5724     }{
5725       \l_tmpa_tl
5726     }
5727     \end{stex_annotate_env}
5728   }
5729 }

```

## **\stexpatchdefinition**

```

5730 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
5731   \stex_par:\noindent\titllemph{Definition\tl_if_empty:NF \sdefinitiontitle {
5732     ~(\sdefinitiontitle)
5733   }~}
5734 }
5735 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\stex_par:\medskip}
5736
5737 \newcommand\stexpatchdefinition[3] [] {
5738   \str_set:Nx \l_tmpa_str{ #1 }
5739   \str_if_empty:NTF \l_tmpa_str {
5740     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
5741     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
5742   }{
5743     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
5744     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }

```

```

5745     }
5746 }

```

(End definition for `\stexpatchdefinition`. This function is documented on page 51.)

`\inlinedef` inline:

```

5747 \keys_define:nn {stex / inlinedef }{
5748   type      .str_set_x:N = \sdefinitiontype,
5749   id        .str_set_x:N = \sdefinitionid,
5750   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5751   name      .str_set_x:N = \sdefinitionname
5752 }
5753 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
5754   \str_clear:N \sdefinitiontype
5755   \str_clear:N \sdefinitionid
5756   \str_clear:N \sdefinitionname
5757   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5758   \keys_set:nn { stex / inlinedef }{ #1 }
5759 }
5760 \NewDocumentCommand \inlinedef { 0{} m } {
5761   \begingroup
5762   \__stex_statements_inlinedef_args:n{ #1 }
5763   \stex_reactivate_macro:N \definiendum
5764   \stex_reactivate_macro:N \definame
5765   \stex_reactivate_macro:N \Definame
5766   \stex_reactivate_macro:N \premise
5767   \stex_reactivate_macro:N \definiens
5768   \stex_reactivate_macro:N \varbindforall
5769   \stex_ref_new_doc_target:n \sdefinitionid
5770   \stex_if_smsmode:TF{\stex_suppress_html:n {
5771     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5772   }}{
5773     \seq_clear:N \l_tmpb_seq
5774     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5775       \tl_if_empty:nF{ ##1 }{
5776         \stex_get_symbol:n { ##1 }
5777         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5778           \l_stex_get_symbol_uri_str
5779         }
5780       }
5781     }
5782     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5783     \exp_args:Nnx
5784     \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpb_seq {,}}{
5785       \str_if_empty:NF \sdefinitiontype {
5786         \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5787       }
5788       #2
5789       \str_if_empty:NF \sdefinitionname {
5790         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sdefinitionname}}
5791         \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5792       }
5793     }
5794   }

```





```

5842 \clist_map_inline:Nn \l_tmpa_clist {
5843   \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
5844     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
5845   }
5846 }
5847 \tl_if_empty:NTF \l_tmpa_tl {
5848   \__stex_statements_sassertion_start:
5849 }{
5850   \l_tmpa_tl
5851 }
5852 }
5853 \str_if_empty:NTF \sassertionid {
5854   \str_if_empty:NF \sassertionname {
5855     \stex_ref_new_doc_target:n {}
5856   }
5857 } {
5858   \stex_ref_new_doc_target:n \sassertionid
5859 }
5860 \stex_smsmode_do:
5861 ){
5862   \str_if_empty:NF \sassertionname {
5863     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5864     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5865   }
5866   \stex_if_smsmode:F {
5867     \clist_set:No \l_tmpa_clist \sassertiontype
5868     \tl_clear:N \l_tmpa_tl
5869     \clist_map_inline:Nn \l_tmpa_clist {
5870       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
5871         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
5872       }
5873     }
5874     \tl_if_empty:NTF \l_tmpa_tl {
5875       \__stex_statements_sassertion_end:
5876     }{
5877       \l_tmpa_tl
5878     }
5879     \end{stex_annotate_env}
5880   }
5881 }

```

### **\stexpatchassertion**

```

5882
5883 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
5884   \stex_par:\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
5885     (\sassertiontitle)
5886   }~}
5887 }
5888 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\stex_par:\medskip}
5889
5890 \newcommand\stexpatchassertion[3] [] {
5891   \str_set:Nx \l_tmpa_str{ #1 }
5892   \str_if_empty:NTF \l_tmpa_str {
5893     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }

```

```

5894     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
5895   }{
5896     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
5897     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
5898   }
5899 }

```

(End definition for `\stexpatchassertion`. This function is documented on page 51.)

`\inlineass` inline:

```

5900 \keys_define:nn {stex / inlineass }{
5901   type      .str_set_x:N = \sassertiontype,
5902   id        .str_set_x:N = \sassertionid,
5903   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
5904   name      .str_set_x:N = \sassertionname
5905 }
5906 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
5907   \str_clear:N \sassertiontype
5908   \str_clear:N \sassertionid
5909   \str_clear:N \sassertionname
5910   \clist_clear:N \l__stex_statements_sassertion_for_clist
5911   \keys_set:nn { stex / inlineass }{ #1 }
5912 }
5913 \NewDocumentCommand \inlineass { 0{} m } {
5914   \beginngroup
5915   \stex_reactivate_macro:N \premise
5916   \stex_reactivate_macro:N \conclusion
5917   \stex_reactivate_macro:N \varbindforall
5918   \__stex_statements_inlineass_args:n{ #1 }
5919   \str_if_empty:NTF \sassertionid {
5920     \str_if_empty:NF \sassertionname {
5921       \stex_ref_new_doc_target:n {}
5922     }
5923   } {
5924     \stex_ref_new_doc_target:n \sassertionid
5925   }
5926
5927   \stex_if_smsmode:TF{
5928     \str_if_empty:NF \sassertionname {
5929       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sassertionname}}
5930     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5931   }
5932 }{
5933   \seq_clear:N \l_tmpb_seq
5934   \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5935     \tl_if_empty:nF{ ##1 }{
5936       \stex_get_symbol:n { ##1 }
5937       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5938         \l_stex_get_symbol_uri_str
5939       }
5940     }
5941   }
5942   \exp_args:Nnx
5943   \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpb_seq {},,}{

```

```

5944 \str_if_empty:NF \sassassertiontype {
5945   \stex_annotate_invisible:nnn{typestrings}{\sassassertiontype}{ }
5946 }
5947 #2
5948 \str_if_empty:NF \sassassertionname {
5949   \stex_suppress_html:n{\stex_symdecl_do:nn}{\sassassertionname}}
5950   \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassassertionname}
5951   \stex_annotate_invisible:nnn{statementname}{\sassassertionname}{ }
5952 }
5953 }
5954 }
5955 \endgroup
5956 \stex_smsmode_do:
5957 }

```

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

## 32.3 Examples

`sexample (env.)`

```

5958
5959 \keys_define:nn {stex / sexample }{
5960   type      .str_set_x:N = \exampletype,
5961   id        .str_set_x:N = \sexampleid,
5962   title     .tl_set:N    = \sexampletitle,
5963   name      .str_set_x:N = \sexamplename ,
5964   for       .clist_set:N = \l__stex_statements_sexample_for_clist,
5965 }
5966 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
5967   \str_clear:N \sexampletype
5968   \str_clear:N \sexampleid
5969   \str_clear:N \sexamplename
5970   \tl_clear:N \sexampletitle
5971   \clist_clear:N \l__stex_statements_sexample_for_clist
5972   \keys_set:nn { stex / sexample }{ #1 }
5973 }
5974
5975 \NewDocumentEnvironment{sexample}{0{}}{
5976   \__stex_statements_sexample_args:n{ #1 }
5977   \stex_reactivate_macro:N \premise
5978   \stex_reactivate_macro:N \conclusion
5979   \stex_if_smsmode:F {
5980     \seq_clear:N \l_tmpb_seq
5981     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5982       \tl_if_empty:nF{ ##1 }{
5983         \stex_get_symbol:n { ##1 }
5984         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5985           \l_stex_get_symbol_uri_str
5986         }
5987       }
5988     }
5989     \exp_args:Nnnx
5990     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpb_seq {,}}

```

```

5991 \str_if_empty:NF \sexamplotype {
5992   \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{}
5993 }
5994 \str_if_empty:NF \sexamplename {
5995   \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
5996 }
5997 \clist_set:No \l_tmpa_clist \sexamplotype
5998 \tl_clear:N \l_tmpa_tl
5999 \clist_map_inline:Nn \l_tmpa_clist {
6000   \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
6001     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
6002   }
6003 }
6004 \tl_if_empty:NTF \l_tmpa_tl {
6005   \__stex_statements_sexample_start:
6006 }{
6007   \l_tmpa_tl
6008 }
6009 }
6010 \str_if_empty:NF \sexampleid {
6011   \stex_ref_new_doc_target:n \sexampleid
6012 }
6013 \stex_smsmode_do:
6014 }{
6015   \str_if_empty:NF \sexamplename {
6016     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
6017   }
6018   \stex_if_smsmode:F {
6019     \clist_set:No \l_tmpa_clist \sexamplotype
6020     \tl_clear:N \l_tmpa_tl
6021     \clist_map_inline:Nn \l_tmpa_clist {
6022       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
6023         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
6024       }
6025     }
6026     \tl_if_empty:NTF \l_tmpa_tl {
6027       \__stex_statements_sexample_end:
6028     }{
6029       \l_tmpa_tl
6030     }
6031     \end{stex_annotate_env}
6032   }
6033 }

```

**\stexpatchexample**

```

6034
6035 \cs_new_protected:Nn \__stex_statements_sexample_start: {
6036   \stex_par:\noindent\titllemph{Example~\tl_if_empty:NF \sexampltitle {
6037     (\sexampltitle)
6038   }~}
6039 }
6040 \cs_new_protected:Nn \__stex_statements_sexample_end: {\stex_par:\medskip}
6041
6042 \newcommand\stexpatchexample[3]{} {

```

```

6043 \str_set:Nx \l_tmpa_str{ #1 }
6044 \str_if_empty:NTF \l_tmpa_str {
6045   \tl_set:Nn \__stex_statements_sexample_start: { #2 }
6046   \tl_set:Nn \__stex_statements_sexample_end: { #3 }
6047 }{
6048   \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
6049   \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
6050 }
6051 }

```

(End definition for `\stexpatchexample`. This function is documented on page 51.)

`\inlineex` inline:

```

6052 \keys_define:nn {stex / inlineex }{
6053   type      .str_set_x:N = \sexamplotype,
6054   id        .str_set_x:N = \sexampleid,
6055   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
6056   name      .str_set_x:N = \sexamplename
6057 }
6058 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
6059   \str_clear:N \sexamplotype
6060   \str_clear:N \sexampleid
6061   \str_clear:N \sexamplename
6062   \clist_clear:N \l__stex_statements_sexample_for_clist
6063   \keys_set:nn { stex / inlineex }{ #1 }
6064 }
6065 \NewDocumentCommand \inlineex { 0{} m } {
6066   \begingroup
6067   \stex_reactivate_macro:N \premise
6068   \stex_reactivate_macro:N \conclusion
6069   \__stex_statements_inlineex_args:n{ #1 }
6070   \str_if_empty:NF \sexampleid {
6071     \stex_ref_new_doc_target:n \sexampleid
6072   }
6073   \stex_if_smsmode:TF{
6074     \str_if_empty:NF \sexamplename {
6075       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sexamplename}}
6076   }
6077 }{
6078   \seq_clear:N \l_tmpb_seq
6079   \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
6080     \tl_if_empty:nF{ ##1 }{
6081       \stex_get_symbol:n { ##1 }
6082       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6083         \l_stex_get_symbol_uri_str
6084       }
6085     }
6086   }
6087   \exp_args:Nnx
6088   \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpb_seq {,}}{
6089     \str_if_empty:NF \sexamplotype {
6090       \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{ }
6091     }
6092     #2

```

```

6093     \str_if_empty:NF \sexamplename {
6094       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
6095       \stex_annotate_invisible:nnn{statementname}{\sexamplename}{ }
6096     }
6097   }
6098 }
6099 \endgroup
6100 \stex_smsmode_do:
6101 }

```

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

## 32.4 Logical Paragraphs

`sparagraph` (*env.*)

```

6102 \keys_define:nn { stex / sparagraph } {
6103   id      .str_set:x:N = \sparagraphid ,
6104   title   .tl_set:N    = \l_stex_sparagraph_title_tl ,
6105   type    .str_set:x:N = \sparagraphtype ,
6106   for     .clist_set:N  = \l__stex_statements_sparagraph_for_clist ,
6107   from    .tl_set:N     = \sparagraphfrom ,
6108   to      .tl_set:N     = \sparagraphto ,
6109   start   .tl_set:N     = \l_stex_sparagraph_start_tl ,
6110   name    .str_set:N    = \sparagraphname ,
6111   imports .tl_set:N     = \l__stex_statements_sparagraph_imports_tl
6112 }
6113
6114 \cs_new_protected:Nn \stex_sparagraph_args:n {
6115   \tl_clear:N \l_stex_sparagraph_title_tl
6116   \tl_clear:N \sparagraphfrom
6117   \tl_clear:N \sparagraphto
6118   \tl_clear:N \l_stex_sparagraph_start_tl
6119   \tl_clear:N \l__stex_statements_sparagraph_imports_tl
6120   \str_clear:N \sparagraphid
6121   \str_clear:N \sparagraphtype
6122   \clist_clear:N \l__stex_statements_sparagraph_for_clist
6123   \str_clear:N \sparagraphname
6124   \keys_set:nn { stex / sparagraph } { #1 }
6125 }
6126 \newif\if@in@omtext\@in@omtextfalse
6127
6128 \NewDocumentEnvironment {sparagraph} { 0{ } } {
6129   \stex_sparagraph_args:n { #1 }
6130   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
6131     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
6132   }{
6133     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
6134   }
6135   \@in@omtexttrue
6136   \stex_if_smsmode:F {
6137     \seq_clear:N \l_tmpb_seq
6138     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
6139       \tl_if_empty:nF{ ##1 }{

```

```

6140     \stex_get_symbol:n { ##1 }
6141     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6142         \l_stex_get_symbol_uri_str
6143     }
6144 }
6145 }
6146 \exp_args:Nnnx
6147 \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}
6148 \str_if_empty:NF \sparagraphtype {
6149     \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
6150 }
6151 \str_if_empty:NF \sparagraphfrom {
6152     \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
6153 }
6154 \str_if_empty:NF \sparagraphto {
6155     \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
6156 }
6157 \str_if_empty:NF \sparagraphname {
6158     \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
6159 }
6160 \clist_set:No \l_tmpa_clist \sparagraphtype
6161 \tl_clear:N \l_tmpa_tl
6162 \clist_map_inline:Nn \sparagraphtype {
6163     \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
6164         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
6165     }
6166 }
6167 \stex_csl_to_imports:No \usemodule \l__stex_statements_sparagraph_imports_tl
6168 \tl_if_empty:NTF \l_tmpa_tl {
6169     \__stex_statements_sparagraph_start:
6170 }{
6171     \l_tmpa_tl
6172 }
6173 }
6174 \clist_set:No \l_tmpa_clist \sparagraphtype
6175 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
6176     {
6177         \stex_reactivate_macro:N \definiendum
6178         \stex_reactivate_macro:N \definame
6179         \stex_reactivate_macro:N \Definame
6180         \stex_reactivate_macro:N \premise
6181         \stex_reactivate_macro:N \definiens
6182     }
6183 \str_if_empty:NTF \sparagraphid {
6184     \str_if_empty:NTF \sparagraphname {
6185         \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
6186             \stex_ref_new_doc_target:n {}
6187         }
6188     } {
6189         \stex_ref_new_doc_target:n {}
6190     }
6191 } {
6192     \stex_ref_new_doc_target:n \sparagraphid
6193 }

```



```

6194 \exp_args:NNx
6195 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
6196   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
6197     \tl_if_empty:nF{ ##1 }{
6198       \stex_get_symbol:n { ##1 }
6199       \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
6200     }
6201   }
6202 }
6203 \stex_smsmode_do:
6204 \ignorespacesandpars
6205 }{
6206   \str_if_empty:NF \sparagraphname {
6207     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
6208     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
6209   }
6210   \stex_if_smsmode:F {
6211     \clist_set:No \l_tmpa_clist \sparagraphtype
6212     \tl_clear:N \l_tmpa_tl
6213     \clist_map_inline:Nn \l_tmpa_clist {
6214       \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
6215         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
6216       }
6217     }
6218     \tl_if_empty:NTF \l_tmpa_tl {
6219       \__stex_statements_sparagraph_end:
6220     }{
6221       \l_tmpa_tl
6222     }
6223     \end{stex_annotate_env}
6224   }
6225 }

```

## \stexpatchparagraph

```

6226
6227 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
6228   \stex_par:\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
6229     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
6230       \titleemph{\l_stex_sparagraph_title_tl}:~
6231     }
6232   }{
6233     \titleemph{\l_stex_sparagraph_start_tl}~
6234   }
6235 }
6236 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\stex_par:\medskip}
6237
6238 \newcommand\stexpatchparagraph[3] [] {
6239   \str_set:Nx \l_tmpa_str{ #1 }
6240   \str_if_empty:NTF \l_tmpa_str {
6241     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
6242     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
6243   }{
6244     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2 }
6245     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }

```

```

6246     }
6247 }
6248
6249 \keys_define:nn { stex / inlinepara } {
6250   id      .str_set:N = \sparagraphid ,
6251   type    .str_set:N = \sparagraphtype ,
6252   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
6253   from    .tl_set:N   = \sparagraphfrom ,
6254   to      .tl_set:N   = \sparagraphto ,
6255   name    .str_set:N   = \sparagraphname
6256 }
6257 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
6258   \tl_clear:N \sparagraphfrom
6259   \tl_clear:N \sparagraphto
6260   \str_clear:N \sparagraphid
6261   \str_clear:N \sparagraphtype
6262   \clist_clear:N \l__stex_statements_sparagraph_for_clist
6263   \str_clear:N \sparagraphname
6264   \keys_set:nn { stex / inlinepara }{ #1 }
6265 }
6266 \NewDocumentCommand \inlinepara { 0{} m } {
6267   \begingroup
6268   \__stex_statements_inlinepara_args:n{ #1 }
6269   \clist_set:Nn \l_tmpa_clist \sparagraphtype
6270   \str_if_empty:NTF \sparagraphid {
6271     \str_if_empty:NTF \sparagraphname {
6272       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
6273         \stex_ref_new_doc_target:n {}
6274       }
6275     } {
6276       \stex_ref_new_doc_target:n {}
6277     }
6278   } {
6279     \stex_ref_new_doc_target:n \sparagraphid
6280   }
6281   \stex_if_smsmode:TF{
6282     \str_if_empty:NF \sparagraphname {
6283       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
6284     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
6285   }
6286 }{
6287   \seq_clear:N \l_tmpb_seq
6288   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
6289     \tl_if_empty:nF{ ##1 }{
6290       \stex_get_symbol:n { ##1 }
6291       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6292         \l_stex_get_symbol_uri_str
6293       }
6294     }
6295   }
6296   \exp_args:Nnx
6297   \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}{
6298     \str_if_empty:NF \sparagraphtype {
6299       \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{

```

```

6300     }
6301     \str_if_empty:NF \sparagraphfrom {
6302         \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
6303     }
6304     \str_if_empty:NF \sparagraphto {
6305         \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
6306     }
6307     \str_if_empty:NF \sparagraphname {
6308         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
6309         \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
6310         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
6311     }
6312     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
6313         \clist_map_inline:Nn \l_tmpb_seq {
6314             \stex_ref_new_sym_target:n {##1}
6315         }
6316     }
6317     #2
6318 }
6319 }
6320 \endgroup
6321 \stex_smsmode_do:
6322 }
6323

```

(End definition for `\stexpatchparagraph`. This function is documented on page 51.)

```

6324 </package>

```

## Chapter 33

# The Implementation

```
6325 <*package>
6326 <@@=stex_sproof>
6327
6328 %%%%%%%%%%    sproof.dtx    %%%%%%%%%%
6329
```

### 33.1 Proofs

We first define some keys for the proof environment.

```
6330 \keys_define:nn { stex / spf } {
6331   id          .str_set_x:N = \spfid,
6332   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,
6333   from        .tl_set:N    = \l__stex_sproof_spf_from_tl ,
6334   proofend    .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
6335   type        .str_set_x:N = \spftype,
6336   title       .tl_set:N    = \spftitle,
6337   continues   .tl_set:N    = \l__stex_sproof_spf_continues_tl,
6338   functions   .tl_set:N    = \l__stex_sproof_spf_functions_tl,
6339   term        .tl_set:N    = \l__stex_sproof_spf_term_tl,
6340   method      .tl_set:N    = \l__stex_sproof_spf_method_tl,
6341   hide        .bool_set:N  = \l__stex_sproof_spf_hide_bool
6342 }
6343 \cs_new_protected:Nn \l__stex_sproof_spf_args:n {
6344   \str_clear:N \spfid
6345   \tl_clear:N \l__stex_sproof_spf_for_tl
6346   \tl_clear:N \l__stex_sproof_spf_from_tl
6347   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
6348   \str_clear:N \spftype
6349   \tl_clear:N \spftitle
6350   \tl_clear:N \l__stex_sproof_spf_continues_tl
6351   \tl_clear:N \l__stex_sproof_spf_term_tl
6352   \tl_clear:N \l__stex_sproof_spf_functions_tl
6353   \tl_clear:N \l__stex_sproof_spf_method_tl
6354   \bool_set_false:N \l__stex_sproof_spf_hide_bool
6355   \keys_set:nn { stex / spf }{ #1 }
6356 }
6357 \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`

```
6358 \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.

```
6359 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
6360 \cs_new_protected:Npn \sproofnumber {
6361   \int_set:Nn \l_tmpa_int {1}
6362   \bool_while_do:nn {
6363     \int_compare_p:nNn {
6364       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6365     } > 0
6366   }{
6367     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
6368     \int_incr:N \l_tmpa_int
6369   }
6370 }
6371 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
6372   \int_set:Nn \l_tmpa_int {1}
6373   \bool_while_do:nn {
6374     \int_compare_p:nNn {
6375       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6376     } > 0
6377   }{
6378     \int_incr:N \l_tmpa_int
6379   }
6380   \int_compare:nNnF \l_tmpa_int = 1 {
6381     \int_decr:N \l_tmpa_int
6382   }
6383   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
6384     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1
6385   }
6386 }
6387
6388 \cs_new_protected:Npn \__stex_sproof_add_counter: {
6389   \int_set:Nn \l_tmpa_int {1}
6390   \bool_while_do:nn {
6391     \int_compare_p:nNn {
6392       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6393     } > 0
6394   }{
6395     \int_incr:N \l_tmpa_int
6396   }
6397   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 1 }
6398 }
6399
```

```

6400 \cs_new_protected:Npn \__stex_sproof_remove_counter: {
6401   \int_set:Nn \l_tmpa_int {1}
6402   \bool_while_do:nn {
6403     \int_compare_p:nNn {
6404       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6405     } > 0
6406   }{
6407     \int_incr:N \l_tmpa_int
6408   }
6409   \int_decr:N \l_tmpa_int
6410   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 0 }
6411 }

```

**\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

```

6412 \def\sproof@box{
6413   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
6414 }
6415 \def\sproofend{
6416   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
6417     \hfil\hfill\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
6418   }
6419 }

```

(End definition for \sproofend. This function is documented on page 51.)

**spf@\*kw**

```

6420 \def\spf@proofsketch@kw{Proof~Sketch}
6421 \def\spf@proof@kw{Proof}
6422 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

6423 \AddToHook{begindocument}{
6424   \ltx@ifpackageloaded{babel}{
6425     \makeatletter
6426     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6427     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6428       \input{sproof-ngerman.ldf}
6429     }
6430     \clist_if_in:NnT \l_tmpa_clist {finnish}{
6431       \input{sproof-finnish.ldf}
6432     }
6433     \clist_if_in:NnT \l_tmpa_clist {french}{
6434       \input{sproof-french.ldf}
6435     }
6436     \clist_if_in:NnT \l_tmpa_clist {russian}{
6437       \input{sproof-russian.ldf}
6438     }
6439     \makeatother
6440   }{}
6441 }

```

spfsketch

```

6442 \newcommand\spfsketch[2] [] {
6443   \begin{group}
6444   \let \premise \stex_proof_premise:
6445   \_stex_sproof_spf_args:n{#1}
6446   \stex_if_smsmode:TF {
6447     \str_if_empty:NF \spfid {
6448       \stex_ref_new_doc_target:n \spfid
6449     }
6450   }{
6451     \seq_clear:N \l_tmpa_seq
6452     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6453       \tl_if_empty:nF{ ##1 }{
6454         \stex_get_symbol:n { ##1 }
6455         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6456           \l_stex_get_symbol_uri_str
6457         }
6458       }
6459     }
6460     \exp_args:Nnx
6461     \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {},,}{
6462       \str_if_empty:NF \spftype {
6463         \stex_annotate_invisible:nnn{type}{\spftype}{
6464         }
6465       }
6466       \clist_set:Nn \l_tmpa_clist \spftype
6467       \tl_set:Nn \l_tmpa_tl {
6468         \titleemph{
6469           \tl_if_empty:NTF \spftitle {
6470             \spf@proofsketch@kw
6471           }{
6472             \spftitle
6473           }
6474         }
6475       }
6476       \clist_map_inline:Nn \l_tmpa_clist {
6477         \exp_args:Nn \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6478           \tl_clear:N \l_tmpa_tl
6479         }
6480       }
6481       \str_if_empty:NF \spfid {
6482         \stex_ref_new_doc_target:n \spfid
6483       }
6484       \l_tmpa_tl #2 \sproofend
6485     }
6486   }
6487   \endgroup
6488 }
6489

```

(End definition for *spfsketch*. This function is documented on page 50.)

```

\_stex_sproof_maybe_comment:
\_stex_sproof_maybe_comment_end:
\_stex_sproof_start_comment:
6490 \bool_set_false:N \l__stex_sproof_in_spfblock_bool

```

```

6491
6492 \cs_new_protected:Nn \__stex_sproof_maybe_comment: {
6493   \bool_if:NF \l__stex_sproof_in_spfblock_bool {
6494     \par \setbox \l_tmpa_box \vbox \bgroup \everypar{\__stex_sproof_start_comment:}
6495   }
6496 }
6497 \cs_new_protected:Nn \__stex_sproof_maybe_comment_end: {
6498   \bool_if:NF \l__stex_sproof_in_spfblock_bool { \egroup }
6499 }
6500 \cs_new_protected:Nn \__stex_sproof_start_comment: {
6501   \csname @ @ par\endcsname\egroup\item[]\bgroup\stexcommentfont
6502 }
6503

```

(End definition for `\__stex_sproof_maybe_comment:`, `\__stex_sproof_maybe_comment_end:`, and `\__stex_sproof_start_comment:`.)

`\stexcommentfont`

```

6504 \cs_new_protected:Npn \stexcommentfont {
6505   \small\itshape
6506 }

```

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

**sproof (env.)** 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.

```

6507 \cs_new_protected:Nn \__stex_sproof_start_env:nnn {
6508   \seq_clear:N \l_tmpa_seq
6509   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6510     \tl_if_empty:NF{ ##1 }{
6511       \stex_get_symbol:n { ##1 }
6512       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6513         \l_stex_get_symbol_uri_str
6514       }
6515     }
6516   }
6517   \exp_args:Nnnx
6518   \begin{stex_annotate_env}{#1}{\seq_use:Nn \l_tmpa_seq {,}}
6519   \str_if_empty:NF \spftype {
6520     \stex_annotate_invisible:nnn{type}{\spftype}{}
6521   }
6522   #3 {\~\stex_annotate:nnn{spftitle}{}{#2}}
6523   \str_if_empty:NF \spfid {
6524     \stex_ref_new_doc_target:n \spfid
6525   }
6526   \begin{stex_annotate_env}{spfbody}{\bool_if:NTF \l__stex_sproof_spf_hide_bool {false}{true}
6527   \bool_if:NT \l__stex_sproof_spf_hide_bool{
6528     \stex_html_backend:F{\setbox\l_tmpa_box\vbox\bgroup}
6529   }
6530   \begin{list}{}{
6531     \setlength\topsep{0pt}
6532     \setlength\parsep{0pt}
6533     \setlength\rightmargin{0pt}

```



```

6534 } \_stex_sproof_maybe_comment:
6535 }
6536 }
6537 \cs_new_protected:Nn \_stex_sproof_end_env:n {
6538   \stex_if_smsmode:F{
6539     \_stex_sproof_maybe_comment_end:
6540     \end{list}
6541     \bool_if:NT \l__stex_sproof_spf_hide_bool{
6542       \stex_html_backend:F{\egroup}
6543     }
6544     \clist_set:No \l_tmpa_clist \spftype
6545     #1
6546     \end{stex_annotate_env}
6547     \end{stex_annotate_env}
6548   }
6549 }
6550 \NewDocumentEnvironment{sproof}{s O{} m}{
6551   \intarray_gzero:N \l__stex_sproof_counter_intarray
6552   \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
6553   \stex_reactivate_macro:N \yield
6554   \stex_reactivate_macro:N \eqstep
6555   \stex_reactivate_macro:N \assumption
6556   \stex_reactivate_macro:N \conclude
6557   \stex_reactivate_macro:N \spfstep
6558   \_stex_sproof_spf_args:n{#2}
6559   \stex_if_smsmode:TF {
6560     \str_if_empty:NF \spfid {
6561       \stex_ref_new_doc_target:n \spfid
6562     }
6563   }{
6564     \_stex_sproof_start_env:nnn{sproof}{#3}{
6565       \clist_set:No \l_tmpa_clist \spftype
6566       \tl_clear:N \l_tmpa_tl
6567       \clist_map_inline:Nn \l_tmpa_clist {
6568         \tl_if_exist:cT {\_stex_sproof_sproof_##1_start:}{
6569           \tl_set:Nn \l_tmpa_tl {\use:c{\_stex_sproof_sproof_##1_start:}}
6570         }
6571         \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6572           \tl_set:Nn \l_tmpa_tl {\use:n{}}
6573         }
6574       }
6575       \tl_if_empty:NTF \l_tmpa_tl {
6576         \_stex_sproof_sproof_start:
6577       }{
6578         \l_tmpa_tl
6579       }
6580     }
6581   }
6582   \stex_smsmode_do:
6583 }{\_stex_sproof_end_env:n{
6584   \tl_clear:N \l_tmpa_tl
6585   \clist_map_inline:Nn \l_tmpa_clist {
6586     \tl_if_exist:cT {\_stex_sproof_sproof_##1_end:}{
6587       \tl_set:Nn \l_tmpa_tl {\use:c{\_stex_sproof_sproof_##1_end:}}

```

```

6588     }
6589   }
6590   \tl_if_empty:NTF \l_tmpa_tl {
6591     \__stex_sproof_sproof_end:
6592   }{
6593     \l_tmpa_tl
6594   }
6595 }}
6596 \NewDocumentEnvironment{subproof}{s O{} m}{
6597   \__stex_sproof_spf_args:n{#2}
6598   \stex_if_smsmode:TF {
6599     \str_if_empty:NF \spfid {
6600       \stex_ref_new_doc_target:n \spfid
6601     }
6602   }{
6603     \__stex_sproof_start_env:nnn{subproof}{\item[\sproofnumber]\ignorespacesandpars #3}{}
6604   }
6605   \__stex_sproof_add_counter:
6606   \stex_smsmode_do:
6607 }{\__stex_sproof_remove_counter:\__stex_sproof_end_env:n{
6608   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6609     \__stex_sproof_inc_counter:
6610   }
6611   \aftergroup\__stex_sproof_maybe_comment:
6612 }
6613 \AddToHook{env/subproof/before}{\__stex_sproof_maybe_comment_end:}
6614
6615 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
6616   \par\noindent\titleemph{
6617     \tl_if_empty:NTF \spftype {
6618       \spf@proof@kw
6619     }{
6620       \spftype
6621     }
6622   }:
6623 }
6624 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
6625
6626 \newcommand\stexpatchproof[3] [] {
6627   \str_set:Nx \l_tmpa_str{ #1 }
6628   \str_if_empty:NTF \l_tmpa_str {
6629     \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
6630     \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
6631   }{
6632     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
6633     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
6634   }
6635 }

```

\pstep  
 \conclude  
 \assumption  
 \have  
 \eqstep

```

6636 \keys_define:nn { stex / spfsteps } {
6637   id          .str_set_x:N = \spfstepid,
6638   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,

```

```

6640 type .str_set_x:N = \spftype,
6641 title .tl_set:N = \spftitle,
6642 method .tl_set:N = \l__stex_sproof_spf_method_tl,
6643 term .tl_set:N = \l__stex_sproof_spf_term_tl
6644 }
6645 \cs_new_protected:Nn \__stex_sproof_spfstep_args:n {
6646 \str_clear:N \spfstepid
6647 \clist_clear:N \l__stex_sproof_spf_for_clist
6648 \str_clear:N \spftype
6649 \tl_clear:N \l__stex_sproof_spf_method_tl
6650 \tl_clear:N \l__stex_sproof_spf_term_tl
6651 %\bool_set_false:N \l__stex_sproof_inc_counter_bool
6652 \keys_set:nn { stex / spfsteps }{ #1 }
6653 }
6654
6655 \cs_new_protected:Nn \__stex_sproof_make_step_macro:Nnnnn {
6656 \NewDocumentCommand #1 {s O{} +m} {
6657 \__stex_sproof_maybe_comment_end:
6658
6659 \__stex_sproof_spfstep_args:n{##2}
6660 \stex_annotate:nnn{spfstep}{#2}{
6661 \tl_if_empty:NF \l__stex_sproof_spf_term_tl {
6662 \stex_annotate_invisible:nnn{spfyield}{}{\l__stex_sproof_spf_term_tl$}
6663 }
6664 \bool_if:NTF \l__stex_sproof_in_spfblock_bool {
6665 #4
6666 }{
6667 \item[\IfBooleanTF ##1 {}{#3}]
6668 }
6669 \ignorespacesandpars ##3
6670 }
6671 \bool_if:NF \l__stex_sproof_in_spfblock_bool { \IfBooleanTF ##1 {}{ #5 } }
6672 \__stex_sproof_maybe_comment:
6673 }
6674 \stex_deactivate_macro:Nn #1 {sproof~environments}
6675 }
6676
6677 \__stex_sproof_make_step_macro:Nnnnn \assumption {assumption} \sproofnumber {} \__stex_sproof
6678 \__stex_sproof_make_step_macro:Nnnnn \conclude {conclusion} {\$ \Rightarrow$} {} {}
6679 \__stex_sproof_make_step_macro:Nnnnn \spfstep {} \sproofnumber {} \__stex_sproof_inc_counter
6680
6681 \NewDocumentCommand \eqstep {s m}{
6682 \__stex_sproof_maybe_comment_end:
6683 \bool_if:NTF \l__stex_sproof_in_spfblock_bool {
6684 $=$
6685 }{
6686 \item[$=$]
6687 }
6688 $\stex_annotate:nnn{spfstep}{eq}{ #2 }$
6689 \__stex_sproof_maybe_comment:
6690 }
6691 \stex_deactivate_macro:Nn \eqstep {sproof~environments}
6692
6693 \NewDocumentCommand \yield {+m}{

```

```

6694 \stex_annotate:nnn{spfyield}{\}{ #1 }
6695 }
6696 \stex_deactivate_macro:Nn \yield {sproof~environments}
6697
6698 \NewDocumentEnvironment{spfblock}{\}{
6699   \item[]
6700   \bool_set_true:N \l__stex_sproof_in_spfblock_bool
6701 }{
6702   \aftergroup\__stex_sproof_maybe_comment:
6703 }
6704 \AddToHook{env/spfblock/before}{\__stex_sproof_maybe_comment_end:}
6705

```

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

### **\spfidea**

```

6706 \NewDocumentCommand\spfidea{0{} +m}{
6707   \__stex_sproof_spf_args:n{#1}
6708   \titleemph{
6709     \tl_if_empty:NTF \spftype {Proof~Idea}{
6710       \spftype
6711     }:
6712   }~#2
6713   \sproofend
6714 }

```

*(End definition for \spfidea. This function is documented on page 50.)*

```

6715 \newcommand\spfjust[1]{
6716   #1
6717 }
6718 \</package>

```

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

## Chapter 34

# STEX -Others Implementation

```
6719 <*package>
6720
6721 %%%%%%%%%% others.dtx %%%%%%%%%%
6722
6723 <@@=stex_others>
        Warnings and error messages
6724 % None

\MSC Math subject classifier

6725 \NewDocumentCommand \MSC {m} {
6726 % TODO
6727 }

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

6728 \@ifpackageloaded{tikzinput}{
6729 \RequirePackage{stex-tikzinput}
6730 }{}
6731
6732 \bool_if:NT \c_stex_persist_mode_bool {
6733 \let__stex_notation_restore_notation_old:nnnnn
6734 \__stex_notation_restore_notation:nnnnn
6735 \def__stex_notation_restore_notation_new:nnnnn#1#2#3#4#5{
6736 \__stex_notation_restore_notation_old:nnnnn{#1}{#2}{#3}{#4}{#5}
6737 \ExplSyntaxOn
6738 }
6739 \def__stex_notation_restore_notation:nnnnn{
6740 \ExplSyntaxOff
6741 \catcode'\sim10
6742 \__stex_notation_restore_notation_new:nnnnn
6743 }
6744 \input{\jobname.sms}
6745 \let__stex_notation_restore_notation:nnnnn
6746 \__stex_notation_restore_notation_old:nnnnn
6747 \prop_if_exist:NT\c_stex_mathhub_main_manifest_prop{
```

```

6748 \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
6749 \l_tmpa_str
6750 \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str _manifest_prop }
6751 \c_stex_mathhub_main_manifest_prop
6752 \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
6753 }
6754 }
6755
6756 \stex_get_document_uri:
6757 </package>

```

## Chapter 35

# STEX -Metatheory Implementation

```
6758 <*package>
6759 <@@=stex_modules>
6760
6761 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
6762
6763 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX/meta}
6764 \begingroup
6765 \stex_module_setup:nn{
6766   ns=\c_stex_metatheory_ns_str,
6767   meta=NONE
6768 }{Metatheory}
6769 \stex_reactivate_macro:N \symdecl
6770 \stex_reactivate_macro:N \notation
6771 \stex_reactivate_macro:N \symdef
6772 \ExplSyntaxOff
6773 \csname stex_suppress_html:n\endcsname{
6774   % is-a (a:A, a \in A, a is an A, etc.)
6775   \symdecl{isa}[args=ai]
6776   \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
6777   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
6778   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
6779
6780   % bind (\forall, \Pi, \lambda etc.)
6781   \symdecl{bind}[args=Bi,assoc=pre]
6782   \notation{bind}[depfun,prec=nobrackets,op={(\cdot)\;\to\;\cdot}]{\comp( #1 \comp{}\;\to\;)}
6783   \notation{bind}[forall]{\comp\forall #1.\;#2}{##1 \comp, ##2}
6784   \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6785
6786   % implicit bind
6787   \symdecl{implicitbind}[args=Bi,assoc=pre]
6788   \notation{implicitbind}[braces,prec=nobrackets,op={(\cdot\_I\;\cdot)}]{\comp\{ #1 \comp{
6789   \notation{implicitbind}[depfun,prec=nobrackets]{\comp( #1 \comp{}\;\to\_I\; } #2}{##1 \comp,
6790   \notation{implicitbind}[Pi]{\comp\prod^I_{#1}#2}{##1\comp,##2}
6791
6792   % dummy variable
```

```

6793 \symdecl{dummyvar}
6794 \notation{dummyvar}[underscore]{\comp\_}
6795 \notation{dummyvar}[dot]{\comp\cdot}
6796 \notation{dummyvar}[dash]{\comp{\rm --}}
6797
6798 %fromto (function space, Hom-set, implication etc.)
6799 \symdecl{fromto}[args=ai]
6800 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
6801 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
6802
6803 % mapto (lambda etc.)
6804 \symdecl{mapto}[args=Bi]
6805 %\notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
6806 %\notation{mapto}[lambda]{\comp\lambda #1 \comp.\; #2}{#1 \comp, #2}
6807 %\notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.\; #2}{#1 \comp, #2}
6808
6809 % function/operator application
6810 \symdecl{apply}[args=ia]
6811 \notation{apply}[prec=0;0x\infpres,parens,op=\cdot(\cdot)]{#1 \comp( #2 \comp)}{##1 \comp,
6812 \notation{apply}[prec=0;0x\infpres,lambda]{#1 \; #2 }{##1 \; ; ##2}
6813
6814 % collection of propositions/booleans/truth values
6815 \symdecl{prop}[name=proposition]
6816 \notation{prop}[prop]{\comp{\rm prop}}
6817 \notation{prop}[BOOL]{\comp{\rm BOOL}}
6818
6819 \symdecl{judgmentholds}[args=1]
6820 \notation{judgmentholds}[vdash,op=\vdash]{\comp\vdash\; #1}
6821
6822 % sequences
6823 \symdecl{seqtype}[args=1]
6824 \notation{seqtype}[kleene]{#1^{\comp\ast}}
6825
6826 \symdecl{seqexpr}[args=a]
6827 \notation{seqexpr}[angle,prec=nobrackets]{\comp\langle #1\comp\rangle}{##1\comp,##2}
6828
6829 \symdef{seqmap}[args=abi,setlike]{\comp\{#3 \comp| #2\comp\in \dobrackets{#1} \comp\}}{##1\comp,##2}
6830 \symdef{seqprepend}[args=ia]{#1 \comp{::} #2}{##1 \comp, ##2}
6831 \symdef{seqappend}[args=ai]{#1 \comp{::} #2}{##1 \comp, ##2}
6832 \symdef{seqfoldleft}[args=iabbi]{ \comp{foldl}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\comp}
6833 \symdef{seqfoldright}[args=iabbi,op=foldr]{ \comp{foldr}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\comp}
6834 \symdef{seqhead}[args=a]{\comp{head}\dobrackets{#1}}{##1 \comp, ##2}
6835 \symdef{seqtail}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6836 \symdef{seqlast}[args=a]{\comp{last}\dobrackets{#1}}{##1 \comp, ##2}
6837 \symdef{seqinit}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6838
6839 \symdef{sequence-index}[args=2,li,prec=nobrackets]{\comp{#1}_{#2}}
6840 \notation{sequence-index}[ui,prec=nobrackets]{\comp{#1}^{\comp{#2}}}
6841
6842 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{\,\ellipses\,}}{##1\comp,##2}
6843 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{\,\ellipses\,}#2}{##1\comp,##2}
6844 \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{\,\ellipses\,}#2\comp{\,\ellipses\,}#3}{##1\comp,##2,##3}
6845
6846 % nat literals

```



```

6847 \symdef{natliteral}{\comp{\mathtt{Ord}}}
6848
6849 % letin (''let'', local definitions, variable substitution)
6850 \symdecl{letin}[args=bii]
6851 \notation{letin}{let}{\comp{\rm let}}\;#1\comp{=}\;#2\;\comp{\rm in}}\;#3}
6852 \notation{letin}{subst}{#3 \comp[ #1 \comp/ #2 \comp]}
6853 \notation{letin}{frac}{#3 \comp[ \frac{#2}{#1} \comp]}
6854
6855 % structures
6856 \symdecl*{module-type}[args=1]
6857 \notation{module-type}{\comp{\mathtt{MOD}}} #1}
6858 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
6859 \notation{mathstruct}[angle,prec=nobrackets]{\comp\angle #1 \comp\angle}{##1 \comp, ##2}
6860
6861 % objects
6862 \symdecl{object}
6863 \notation{object}{\comp{\mathtt{OBJECT}}}
6864
6865 }
6866
6867 % The following are abbreviations in the sTeX corpus that are left over from earlier
6868 % developments. They will eventually be phased out.
6869
6870 \ExplSyntaxOn
6871 \stex_add_to_current_module:n{
6872   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
6873   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
6874   \def\livar{\csname sequence-index\endcsname[li]}
6875   \def\uivar{\csname sequence-index\endcsname[ui]}
6876   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
6877   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
6878 }
6879 \__stex_modules_end_module:
6880 \endgroup
6881 \</package>

```

## Chapter 36

# Tikzinput Implementation

```
6882 <@@=tikzinput>
6883 <*package>
6884
6885 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
6886
6887 \ProvidesExplPackage{tikzinput}{2022/08/08}{3.2.0}{tikzinput package}
6888 \RequirePackage{l3keys2e}
6889
6890 \keys_define:nn { tikzinput } {
6891   image .bool_set:N = \c_tikzinput_image_bool,
6892   image .default:n = false ,
6893   unknown .code:n = {}
6894 }
6895
6896 \ProcessKeysOptions { tikzinput }
6897
6898 \bool_if:NTF \c_tikzinput_image_bool {
6899   \RequirePackage{graphicx}
6900
6901   \providecommand\usetikzlibrary[]{}
6902   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
6903 }{
6904   \RequirePackage{tikz}
6905   \RequirePackage{standalone}
6906
6907   \newcommand \tikzinput [2] [] {
6908     \setkeys{Gin}{#1}
6909     \ifx \Gin@ewidth \Gin@exclamation
6910       \ifx \Gin@eheight \Gin@exclamation
6911         \input { #2 }
6912       \else
6913         \resizebox{!}{ \Gin@eheight }{
6914           \input { #2 }
6915         }
6916       \fi
6917     \else
6918       \ifx \Gin@eheight \Gin@exclamation
6919         \resizebox{ \Gin@ewidth }{!}{
```

```

6920         \input { #2 }
6921     }
6922     \else
6923         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
6924             \input { #2 }
6925         }
6926     \fi
6927 \fi
6928 }
6929 }
6930
6931 \newcommand \ctikzinput [2] [] {
6932     \begin{center}
6933         \tikzinput [#1] {#2}
6934     \end{center}
6935 }
6936
6937 \@ifpackageloaded{stex}{
6938     \RequirePackage{stex-tikzinput}
6939 }{}
6940
6941 </package>
6942 <*stex>
6943 \ProvidesExplPackage{stex-tikzinput}{2022/08/08}{3.2.0}{stex-tikzinput}
6944 \RequirePackage{stex}
6945 \RequirePackage{tikzinput}
6946
6947 \newcommand\mhtikzinput[2] []{%
6948     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
6949     \stex_in_repository:nn\Gin@mhrepos{
6950         \tikzinput[#1]{\mhp@path{##1}{#2}}
6951     }
6952 }
6953 \newcommand\cmhtikzinput[2] []{\begin{center}\mhtikzinput[#1]{#2}\end{center}}
6954
6955 \cs_new_protected:Nn \__tikzinput_usetikzlibrary:nn {
6956     \pgfkeys@spdef\pgf@temp{#1}
6957     \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
6958     \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgf@temp
6959     \expandafter\edef\csname tikz@library@#1@atcode\endcsname{\the\catcode'\@}
6960     \expandafter\edef\csname tikz@library@#1@barcode\endcsname{\the\catcode'\|}
6961     \expandafter\edef\csname tikz@library@#1@dollarcode\endcsname{\the\catcode'\$}
6962     \catcode'\@=11
6963     \catcode'\|=12
6964     \catcode'\$=3
6965     \pgfutil@InputIfFileExists{#2}{-}{-}
6966     \catcode'\@=\csname tikz@library@#1@atcode\endcsname
6967     \catcode'\|=\csname tikz@library@#1@barcode\endcsname
6968     \catcode'\$=\csname tikz@library@#1@dollarcode\endcsname
6969 }
6970
6971
6972 \newcommand\libusetikzlibrary[1]{

```

```

6973 \prop_if_exist:NF \l_stex_current_repository_prop {
6974   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6975 }
6976 \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
6977   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6978 }
6979 \seq_clear:N \l__tikzinput_libinput_files_seq
6980 \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
6981 \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
6982
6983 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
6984   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / tikzlibrary
6985   \IfFileExists{ \l_tmpa_str }{
6986     \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6987   }{}
6988   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
6989   \seq_put_right:No \l_tmpa_seq \l_tmpa_str
6990 }
6991
6992 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / tikzlibrary #1 .code.t
6993 \IfFileExists{ \l_tmpa_str }{
6994   \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6995 }{}
6996
6997 \seq_if_empty:NTF \l__tikzinput_libinput_files_seq {
6998   \msg_error:nxxx{stex}{error/nofile}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .code.t
6999 }{
7000   \int_compare:nNnTF {\seq_count:N \l__tikzinput_libinput_files_seq} = 1 {
7001     \seq_map_inline:Nn \l__tikzinput_libinput_files_seq {
7002       \__tikzinput_usetikzlibrary:nn{#1}{ ##1 }
7003     }
7004   }{
7005     \msg_error:nxxx{stex}{error/twofiles}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .co
7006   }
7007 }
7008 }
7009 </stex>

```

## Chapter 37

# document-structure.sty Implementation

```
7010 <*package>
7011 <@@=document_structure>
7012 \ProvidesExplPackage{document-structure}{2022/08/08}{3.2.0}{Modular Document Structure}
7013 \RequirePackage{13keys2e}
```

### 37.1 Package Options

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

```
7014
7015 \keys_define:nn{ document-structure }{
7016   class      .str_set_x:N = \c_document_structure_class_str,
7017   topsect    .str_set_x:N = \c_document_structure_topsect_str,
7018   unknown    .code:n      = {
7019     \PassOptionsToClass{\CurrentOption}{stex}
7020     \PassOptionsToClass{\CurrentOption}{tikzinput}
7021   }
7022   % showignores .bool_set:N = \c_document_structure_showignores_bool,
7023 }
7024 \ProcessKeysOptions{ document-structure }
7025 \str_if_empty:NT \c_document_structure_class_str {
7026   \str_set:Nn \c_document_structure_class_str {article}
7027 }
7028 \str_if_empty:NT \c_document_structure_topsect_str {
7029   \str_set:Nn \c_document_structure_topsect_str {section}
7030 }
```

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

```
7031 \RequirePackage{xspace}
7032 \RequirePackage{comment}
7033 \RequirePackage{stex}
7034 \AddToHook{begindocument}{
```

```

7035 \ltx@ifpackageloaded{babel}{
7036   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7037   \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
7038     \makeatletter\input{document-structure-ngerman.ldf}\makeatother
7039   }
7040 }{}
7041 }

```

`\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.

```

7042 \int_new:N \l_document_structure_section_level_int
7043 \str_case:NnF \c_document_structure_topsect_str {
7044   {part}}{
7045     \int_set:Nn \l_document_structure_section_level_int {0}
7046   }
7047   {chapter}{
7048     \int_set:Nn \l_document_structure_section_level_int {1}
7049   }
7050 }{
7051   \str_case:NnF \c_document_structure_class_str {
7052     {book}{
7053       \int_set:Nn \l_document_structure_section_level_int {0}
7054     }
7055     {report}{
7056       \int_set:Nn \l_document_structure_section_level_int {0}
7057     }
7058   }{
7059     \int_set:Nn \l_document_structure_section_level_int {2}
7060   }
7061 }

```

## 37.2 Document Structure

The structure of the document is given by the `sfragment` environment. The hierarchy is adjusted automatically according to the  $\text{\LaTeX}$  class in effect.

`\currentsectionlevel` For the `\currentsectionlevel` and `\Currentsectionlevel` macros we use an internal macro `\current@section@level` that only contains the keyword (no markup). We initialize it with “document” as a default. In the generated OMDoc, we only generate a text element of class `omdoc_currentsectionlevel`, which will be instantiated by CSS later.<sup>9</sup>

EdN:9

```

7062 \def\current@section@level{document}%
7063 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
7064 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%

```

(End definition for `\currentsectionlevel`. This function is documented on page 58.)

`\skipfragment`

```

7065 \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.

```

7066 \ifcase\l_document_structure_section_level_int
7067 \or\stepcounter{part}
7068 \or\stepcounter{chapter}
7069 \or\stepcounter{section}
7070 \or\stepcounter{subsection}
7071 \or\stepcounter{subsubsection}
7072 \or\stepcounter{paragraph}
7073 \or\stepcounter{subparagraph}
7074 \fi
7075 }

```

(End definition for `\skipfragment`. This function is documented on page 57.)

`blindfragment (env.)`

```

7076 \newcommand\at@begin@blindsfragment[1]{
7077 \newenvironment{blindfragment}
7078 {
7079 \int_incr:N\l_document_structure_section_level_int
7080 \at@begin@blindsfragment\l_document_structure_section_level_int
7081 }{}

```

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

```

7082 \newcommand\sfragment@nonum[2]{
7083 \ifx\hyper@anchor\@undefined\else\phantomsection\fi
7084 \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
7085 }

```

(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.

```

7086 \newcommand\sfragment@num[2]{
7087 \tl_if_empty:NTF \l__document_structure_sfragment_short_tl {
7088 \@nameuse{#1}{#2}
7089 }{
7090 \cs_if_exist:NTF\rdfmata@sectioning{
7091 \@nameuse{rdfmata@#1@old}[\l__document_structure_sfragment_short_tl]{#2}
7092 }{
7093 \@nameuse{#1}[\l__document_structure_sfragment_short_tl]{#2}
7094 }
7095 }
7096 %\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\sfragment@id
7097 }

```

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

`sfragment (env.)`

```

7098 \keys_define:nn { document-structure / sfragment }{
7099 id .str_set_x:N = \l__document_structure_sfragment_id_str,
7100 date .str_set_x:N = \l__document_structure_sfragment_date_str,

```

```

7101 creators      .clist_set:N = \l__document_structure_sfragment_creators_clist,
7102 contributors  .clist_set:N = \l__document_structure_sfragment_contributors_clist,
7103 srccite        .tl_set:N     = \l__document_structure_sfragment_srccite_tl,
7104 type           .tl_set:N     = \l__document_structure_sfragment_type_tl,
7105 short          .tl_set:N     = \l__document_structure_sfragment_short_tl,
7106 intro          .tl_set:N     = \l__document_structure_sfragment_intro_tl,
7107 imports        .tl_set:N     = \l__document_structure_sfragment_imports_tl,
7108 loadmodules    .bool_set:N   = \l__document_structure_sfragment_loadmodules_bool
7109 }
7110 \cs_new_protected:Nn \__document_structure_sfragment_args:n {
7111   \str_clear:N \l__document_structure_sfragment_id_str
7112   \str_clear:N \l__document_structure_sfragment_date_str
7113   \clist_clear:N \l__document_structure_sfragment_creators_clist
7114   \clist_clear:N \l__document_structure_sfragment_contributors_clist
7115   \tl_clear:N \l__document_structure_sfragment_srccite_tl
7116   \tl_clear:N \l__document_structure_sfragment_type_tl
7117   \tl_clear:N \l__document_structure_sfragment_short_tl
7118   \tl_clear:N \l__document_structure_sfragment_imports_tl
7119   \tl_clear:N \l__document_structure_sfragment_intro_tl
7120   \bool_set_false:N \l__document_structure_sfragment_loadmodules_bool
7121   \keys_set:nn { document-structure / sfragment } { #1 }
7122 }

```

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.

`\at@begin@sfragment`

```

7123 \newif\if@mainmatter\@mainmattertrue
7124 \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.

```

7125 \keys_define:nn { document-structure / sectioning }{
7126   name      .str_set_x:N = \l__document_structure_sect_name_str ,
7127   ref       .str_set_x:N = \l__document_structure_sect_ref_str  ,
7128   clear     .bool_set:N  = \l__document_structure_sect_clear_bool ,
7129   clear     .default:n   = {true} ,
7130   num       .bool_set:N  = \l__document_structure_sect_num_bool  ,
7131   num       .default:n   = {true}
7132 }
7133 \cs_new_protected:Nn \__document_structure_sect_args:n {
7134   \str_clear:N \l__document_structure_sect_name_str
7135   \str_clear:N \l__document_structure_sect_ref_str
7136   \bool_set_false:N \l__document_structure_sect_clear_bool
7137   \bool_set_false:N \l__document_structure_sect_num_bool
7138   \keys_set:nn { document-structure / sectioning } { #1 }
7139 }
7140 \newcommand\omdoc@sectioning[3][]{ }
7141   \__document_structure_sect_args:n {#1 }
7142   \let\omdoc@sect@name\l__document_structure_sect_name_str
7143   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
7144   \if@mainmatter% numbering not overridden by frontmatter, etc.
7145     \bool_if:NTF \l__document_structure_sect_num_bool {
7146       \sfragment@num{#2}{#3}
7147     }{

```



```

7148     \sfragment@nonum{#2}{#3}
7149   }
7150   \def\current@section@level{\omdoc@sect@name}
7151   \else
7152     \sfragment@nonum{#2}{#3}
7153   \fi
7154 }% 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.

```

7155 \newcommand\sfragment@redefine@addtocontents[1]{%
7156 %\edef\__document_structureimport{#1}%
7157 %\@for\@I:=\__document_structureimport\do{%
7158 %\edef\@path{\csname module@\@I @path\endcsname}%
7159 %\@ifundefined{tf@toc}\relax%
7160 %    {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
7161 %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
7162 %\def\addcontentsline##1##2##3{%
7163 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
7164 %\else% hyperref.sty not loaded
7165 %\def\addcontentsline##1##2##3{%
7166 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
7167 %\fi
7168 }% 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.

```

7169 \newenvironment{sfragment}[2][ ]% keys, title
7170 {
7171   \__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.

```

7172   \stex_csl_to_imports:No \usemodule \l__document_structure_sfragment_imports_tl
7173
7174   \bool_if:NT \l__document_structure_sfragment_loadmodules_bool {
7175     \sfragment@redefine@addtocontents{
7176       %\@ifundefined{module@id}\used@modules%
7177       %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}
7178     }
7179   }

```

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

```

7180
7181   \stex_document_title:n { #2 }
7182
7183   \int_incr:N\l__document_structure_section_level_int
7184   \ifcase\l__document_structure_section_level_int
7185     \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
7186     \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
7187     \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
7188     \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}

```

```

7189 \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
7190 \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#1}
7191 \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#1}
7192 \fi
7193 \at@begin@sfragment[#1]\l__document_structure_section_level_int{#2}
7194 \str_if_empty:NF \l__document_structure_sfragment_id_str {
7195   \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
7196 }
7197 }% for customization
7198 {}

```

and finally, we localize the sections

```

7199 \newcommand\omdoc@part@kw{Part}
7200 \newcommand\omdoc@chapter@kw{Chapter}
7201 \newcommand\omdoc@section@kw{Section}
7202 \newcommand\omdoc@subsection@kw{Subsection}
7203 \newcommand\omdoc@subsubsection@kw{Subsubsection}
7204 \newcommand\omdoc@paragraph@kw{paragraph}
7205 \newcommand\omdoc@subparagraph@kw{subparagraph}

```

### 37.3 Front and Backmatter

Index markup is provided by the `omtext` package [Kohlhase:smmtf:git], so in the `document-structure` package we only need to supply the corresponding `\printindex` command, if it is not already defined

`\printindex`

```

7206 \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).

```

7207 \cs_if_exist:NTF\frontmatter{
7208   \let\__document_structure_orig_frontmatter\frontmatter
7209   \let\frontmatter\relax
7210 }{
7211   \tl_set:Nn\__document_structure_orig_frontmatter{
7212     \clearpage
7213     \@mainmatterfalse
7214     \pagenumbering{roman}
7215   }
7216 }
7217 \cs_if_exist:NTF\backmatter{
7218   \let\__document_structure_orig_backmatter\backmatter
7219   \let\backmatter\relax
7220 }{
7221   \tl_set:Nn\__document_structure_orig_backmatter{
7222     \clearpage
7223     \@mainmatterfalse
7224     \pagenumbering{roman}
7225   }

```

7226 }

Using these, we can now define the `frontmatter` and `backmatter` environments

`frontmatter (env.)` we use the `\orig@frontmatter` macro defined above and `\mainmatter` if it exists, otherwise we define it.

```
7227 \newenvironment{frontmatter}{
7228   \_document_structure_orig_frontmatter
7229 }{
7230   \cs_if_exist:NTF\mainmatter{
7231     \mainmatter
7232   }{
7233     \clearpage
7234     \@mainmattertrue
7235     \pagenumbering{arabic}
7236   }
7237 }
```

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

```
7238 \newenvironment{backmatter}{
7239   \_document_structure_orig_backmatter
7240 }{
7241   \cs_if_exist:NTF\mainmatter{
7242     \mainmatter
7243   }{
7244     \clearpage
7245     \@mainmattertrue
7246     \pagenumbering{arabic}
7247   }
7248 }
```

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

```
7249 \@mainmattertrue\pagenumbering{arabic}
```

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

```
7250 \def \c__document_structure_document_str{document}
7251 \newcommand\afterprematurestop{}
7252 \def\prematurestop@endsfragment{
7253   \unless\ifx\@currenvir\c__document_structure_document_str
7254     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
7255       \expandafter\prematurestop@endsfragment
7256     }
7257   }
7258 \providecommand\prematurestop{
7259   \message{Stopping~sTeX~processing~prematurely}
7260   \prematurestop@endsfragment
7261   \afterprematurestop
7262   \end{document}
7263 }
```

(End definition for `\prematurestop`. This function is documented on page 58.)

## 37.4 Global Variables

**\setSGvar** set a global variable

```
7264 \RequirePackage{etoolbox}
7265 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}
```

*(End definition for \setSGvar. This function is documented on page 58.)*

**\useSGvar** use a global variable

```
7266 \newrobustcmd\useSGvar[1]{%
7267   \@ifundefined{sTeX@Gvar@#1}
7268   {\PackageError{document-structure}
7269     {The sTeX Global variable #1 is undefined}
7270     {set it with \protect\setSGvar}}
7271   \@nameuse{sTeX@Gvar@#1}}
```

*(End definition for \useSGvar. This function is documented on page 58.)*

**\ifSGvar** execute something conditionally based on the state of the global variable.

```
7272 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
7273   \@ifundefined{sTeX@Gvar@#1}
7274   {\PackageError{document-structure}
7275     {The sTeX Global variable #1 is undefined}
7276     {set it with \protect\setSGvar}}
7277   {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}
```

*(End definition for \ifSGvar. This function is documented on page 58.)*

## Chapter 38

# NotesSlides – Implementation

### 38.1 Class and Package Options

We define some Package Options and switches for the `notesslides` class and activate them by passing them on to `beamer.cls` and `omdoc.cls` and the `notesslides` package. We pass the `nontheorem` option to the `statements` package when we are not in notes mode, since the `beamer` package has its own (overlay-aware) theorem environments.

```
7278 \*cls)
7279 \@@=notesslides)
7280 \ProvidesExplClass{notesslides}{2022/08/08}{3.2.0}{notesslides Class}
7281 \RequirePackage{13keys2e}
7282
7283 \keys_define:nn{notesslides / cls}{
7284   class .str_set_x:N = \c__notesslides_class_str,
7285   notes .bool_set:N = \c__notesslides_notes_bool ,
7286   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
7287   docopt .str_set_x:N = \c__notesslides_docopt_str,
7288   unknown .code:n = {
7289     \PassOptionsToPackage{\CurrentOption}{document-structure}
7290     \PassOptionsToClass{\CurrentOption}{beamer}
7291     \PassOptionsToPackage{\CurrentOption}{notesslides}
7292     \PassOptionsToPackage{\CurrentOption}{stex}
7293   }
7294 }
7295 \ProcessKeysOptions{ notesslides / cls }
7296
7297 \str_if_empty:NF \c__notesslides_class_str {
7298   \PassOptionsToPackage{class=\c__notesslides_class_str}{document-structure}
7299 }
7300
7301 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{book}{
7302   \PassOptionsToPackage{defaultttopsect=part}{notesslides}
7303 }
7304 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{report}{
7305   \PassOptionsToPackage{defaultttopsect=part}{notesslides}
7306 }
7307
7308 \RequirePackage{stex}
```

```

7309 \stex_html_backend:T {
7310   \bool_set_true:N\c__notesslides_notes_bool
7311 }
7312
7313 \bool_if:NTF \c__notesslides_notes_bool {
7314   \PassOptionsToPackage{notes=true}{notesslides}
7315   \message{notesslides.cls:~Formatting~course~materials~in~notes~mode}
7316 }{
7317   \PassOptionsToPackage{notes=false}{notesslides}
7318   \message{notesslides.cls:~Formatting~course~materials~in~slides~mode}
7319 }
7320 </cls>

```

now we do the same for the notesslides package.

```

7321 <*package>
7322 \ProvidesExplPackage{notesslides}{2022/08/08}{3.2.0}{notesslides Package}
7323 \RequirePackage{l3keys2e}
7324
7325 \keys_define:nn{notesslides / pkg}{
7326   topsect          .str_set_x:N = \c__notesslides_topsect_str,
7327   defaulttopsect   .str_set_x:N = \c__notesslides_defaulttopsec_str,
7328   notes            .bool_set:N = \c__notesslides_notes_bool ,
7329   slides           .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
7330   sectocframes     .bool_set:N = \c__notesslides_sectocframes_bool ,
7331   frameimages      .bool_set:N = \c__notesslides_frameimages_bool ,
7332   fiboxed          .bool_set:N = \c__notesslides_fiboxed_bool ,
7333   nopproblems      .bool_set:N = \c__notesslides_nopproblems_bool,
7334   unknown          .code:n      = {
7335     \PassOptionsToClass{\CurrentOption}{stex}
7336     \PassOptionsToClass{\CurrentOption}{tikzinput}
7337   }
7338 }
7339 \ProcessKeysOptions{ notesslides / pkg }
7340
7341 \RequirePackage{stex}
7342 \stex_html_backend:T {
7343   \bool_set_true:N\c__notesslides_notes_bool
7344 }
7345
7346 \newif\ifnotes
7347 \bool_if:NTF \c__notesslides_notes_bool {
7348   \notesttrue
7349 }{
7350   \notesfalse
7351 }
7352

```

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

```

7353 \str_if_empty:NTF \c__notesslides_topsect_str {
7354   \str_set_eq:NN \__notesslides_topsect \c__notesslides_defaulttopsec_str
7355 }{
7356   \str_set_eq:NN \__notesslides_topsect \c__notesslides_topsect_str
7357 }
7358 \PassOptionsToPackage{topsect=\__notesslides_topsect}{document-structure}

```

```
7359 </package>
```

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

```
7360 <*cls>
7361 \bool_if:NTF \c__notesslides_notes_bool {
7362   \str_if_empty:NT \c__notesslides_class_str {
7363     \str_set:Nn \c__notesslides_class_str {article}
7364   }
7365   \exp_after:wN\LoadClass\exp_after:wN[\c__notesslides_dcopt_str]
7366     {\c__notesslides_class_str}
7367 }{
7368   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
7369   \newcounter{Item}
7370   \newcounter{paragraph}
7371   \newcounter{subparagraph}
7372   \newcounter{Hfootnote}
7373 }
7374 \RequirePackage{document-structure}
```

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

```
7375 \RequirePackage{notesslides}
7376 </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.

```
7377 <*package>
7378 \bool_if:NT \c__notesslides_notes_bool {
7379   \RequirePackage{a4wide}
7380   \RequirePackage{marginnote}
7381   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
7382   \RequirePackage{mdframed}
7383   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
7384   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
7385 }
7386 \RequirePackage{stex-tikzinput}
7387 \RequirePackage{comment}
7388 \RequirePackage{url}
7389 \RequirePackage{graphicx}
7390 \RequirePackage{pgf}
7391 \RequirePackage{bookmark}
```

## 38.2 Notes and Slides

For the lecture notes cases, we also provide the `\usetheme` macro that would otherwise come from the `beamer` class.

```
7392 \bool_if:NT \c__notesslides_notes_bool {
7393   \renewcommand\usetheme[2][\usepackage{#1}{beamertheme#2}]
7394 }
```

```

7395 \NewDocumentCommand \libusetheme {0{} m} {
7396   \libusepackage[#1]{beamertheme#2}
7397 }
7398

```

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

```

7399 \newcounter{slide}
7400 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
7401 \newlength{\slideheight}\setlength{\slideheight}{9cm}

```

**note** (*env.*) 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.

```

7402 \bool_if:NTF \c__notesslides_notes_bool {
7403   \renewenvironment{note}{\ignorespaces}{}
7404 }{
7405   \excludecomment{note}
7406 }

```

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.

```

7407 \bool_if:NT \c__notesslides_notes_bool {
7408   \newlength{\slideframewidth}
7409   \setlength{\slideframewidth}{1.5pt}

```

**frame** (*env.*) We first define the keys.

```

7410 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
7411   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
7412     \bool_set_true:N #1
7413   }{
7414     \bool_set_false:N #1
7415   }
7416 }
7417 \keys_define:nn{notesslides / frame}{
7418   label .str_set_x:N = \l__notesslides_frame_label_str,
7419   allowframebreaks .code:n = {
7420     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
7421   },
7422   allowdisplaybreaks .code:n = {
7423     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
7424   },
7425   fragile .code:n = {
7426     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
7427   },
7428   shrink .code:n = {
7429     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
7430   },
7431   squeeze .code:n = {
7432     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
7433   },
7434   t .code:n = {

```



```

7435     \_notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
7436   },
7437   unknown    .code:n      = {}
7438 }
7439 \cs_new_protected:Nn \_notesslides_frame_args:n {
7440   \str_clear:N \l__notesslides_frame_label_str
7441   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
7442   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
7443   \bool_set_true:N \l__notesslides_frame_fragile_bool
7444   \bool_set_true:N \l__notesslides_frame_shrink_bool
7445   \bool_set_true:N \l__notesslides_frame_squeeze_bool
7446   \bool_set_true:N \l__notesslides_frame_t_bool
7447   \keys_set:nn { notesslides / frame }{ #1 }
7448 }

```

We define the environment, read them, and construct the slide number and label.

```

7449 \renewenvironment{frame}[1][]{
7450   \_notesslides_frame_args:n{#1}
7451   \sffamily
7452   \stepcounter{slide}
7453   \def\@currentlabel{\theslide}
7454   \str_if_empty:NF \l__notesslides_frame_label_str {
7455     \label{\l__notesslides_frame_label_str}
7456   }

```

We redefine the `itemize` environment so that it looks more like the one in `beamer`.

```

7457   \def\itemize@level{outer}
7458   \def\itemize@outer{outer}
7459   \def\itemize@inner{inner}
7460   \renewcommand\newpage{\addtocounter{framenum}{1}}
7461   %\newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
7462   \renewenvironment{itemize}{
7463     \ifx\itemize@level\itemize@outer
7464       \def\itemize@label{$\rhd$}
7465     \fi
7466     \ifx\itemize@level\itemize@inner
7467       \def\itemize@label{$\scriptstyle\rhd$}
7468     \fi
7469     \begin{list}
7470       {\itemize@label}
7471       {\setlength{\labelsep}{.3em}
7472        \setlength{\labelwidth}{.5em}
7473        \setlength{\leftmargin}{1.5em}
7474       }
7475     \edef\itemize@level{\itemize@inner}
7476   }{
7477     \end{list}
7478   }

```

We create the box with the `mdframed` environment from the `equinymous` package.

```

7479   \stex_html_backend:TF {
7480     \begin{stex_annotate_env}{frame}{}\vbox\bgroup
7481     \mdf@patchamsthm
7482   }{
7483     \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwid

```

```

7484     }
7485   }{
7486     \stex_html_backend:TF {
7487       \miko@slidelabel\egroup\end{stex_annotate_env}
7488     }\medskip\miko@slidelabel\end{mdframed}}
7489   }

```

Now, we need to redefine the frametitle (we are still in course notes mode).

`\frametitle`

```

7490   \renewcommand{\frametitle}[1]{
7491     \stex_document_title:n { #1 }
7492     {\Large\bf\sf\color{blue}{#1}}\medskip
7493   }
7494 }

```

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

EdN:10

`\pause` 10

```

7495 \bool_if:NT \c__notesslides_notes_bool {
7496   \newcommand\pause{}
7497 }

```

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

`nparagraph (env.)`

```

7498 \bool_if:NTF \c__notesslides_notes_bool {
7499   \newenvironment{nparagraph}[1] [] {\begin{sparagraph}[#1]}\end{sparagraph}}
7500 }{
7501   \excludecomment{nparagraph}
7502 }

```

`nfragment (env.)`

```

7503 \bool_if:NTF \c__notesslides_notes_bool {
7504   \newenvironment{nfragment}[2] [] {\begin{sfragment}[#1]{#2}}\end{sfragment}}
7505 }{
7506   \excludecomment{nfragment}
7507 }

```

`ndefinition (env.)`

```

7508 \bool_if:NTF \c__notesslides_notes_bool {
7509   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}\end{sdefinition}}
7510 }{
7511   \excludecomment{ndefinition}
7512 }

```

`nassertion (env.)`

```

7513 \bool_if:NTF \c__notesslides_notes_bool {
7514   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}\end{sassertion}}
7515 }{
7516   \excludecomment{nassertion}
7517 }

```

---

<sup>10</sup>EDNOTE: MK: fake it in notes mode for now

`nsproof (env.)`

```
7518 \bool_if:NTF \c__notesslides_notes_bool {
7519   \newenvironment{nproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
7520 }{
7521   \excludecomment{nproof}
7522 }
```

`nexample (env.)`

```
7523 \bool_if:NTF \c__notesslides_notes_bool {
7524   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}{\end{sexample}}
7525 }{
7526   \excludecomment{nexample}
7527 }
```

`\inputref@*skip` We customize the hooks for in `\inputref`.

```
7528 \def\inputref@preskip{\smallskip}
7529 \def\inputref@postskip{\medskip}
```

(End definition for `\inputref@*skip`. This function is documented on page ??.)

**`\inputref*`**

```
7530 \let\orig@inputref\inputref
7531 \def\inputref{@ifstar\ninputref\orig@inputref}
7532 \newcommand\ninputref[2] []{
7533   \bool_if:NT \c__notesslides_notes_bool {
7534     \orig@inputref[#1]{#2}
7535   }
7536 }
```

(End definition for `\inputref*`. This function is documented on page 60.)

## 38.3 Header and Footer Lines

Now, we set up the infrastructure for the footer line of the slides, we use boxes for the logos, so that they are only loaded once, that considerably speeds up processing.

**`\setslidelogo`** The default logo is the `STEX` logo. Customization can be done by `\setslidelogo{<logo name>}`.

```
7537 \newlength{\slidelogoheight}
7538
7539 \RequirePackage{graphicx}
7540
7541 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7542 \providecommand\mhgraphics[2] []{
7543   \def\Gin@mhrepos{}\setkeys{Gin}{#1}
7544   \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}
7545 }
7546
7547 \bool_if:NTF \c__notesslides_notes_bool {
7548   \setlength{\slidelogoheight}{.4cm}
7549 }{
7550   \setlength{\slidelogoheight}{.25cm}
7551 }
```

```

7552 \ifcsname slidelogo\endcsname\else
7553   \newsavebox{\slidelogo}
7554   \sbox{\slidelogo}{\sTeX}
7555 \fi
7556 \newrobustcmd{\setslidelogo}[2][]{
7557   \tl_if_empty:nTF{#1}{
7558     \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#2}}
7559   }{
7560     \sbox{\slidelogo}{\mhgraphics[height=\slidelogoheight,mhrepos=#1]{#2}}
7561   }
7562 }

```

(End definition for `\setslidelogo`. This function is documented on page 61.)

**\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.

```

7563 \bool_if:NT \c__notesslides_notes_bool {
7564   \def\author{\@dblarg\@ns@author}
7565   \long\def\@ns@author[#1]#2{%
7566     \def\c__notesslides_shortauthor{#1}%
7567     \def\@author{#2}
7568   }
7569 }

```

(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.

```

7570 \newrobustcmd{\setsource}[1]{\def\source{#1}}

```

(End definition for `\setsource`. This function is documented on page 61.)

**\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.

```

7571 \def\copyrightnotice{%
7572   \footnotesize\copyright : \hspace{.3ex}%
7573   \ifcsname source\endcsname\source\else%
7574   \ifcsname c__notesslides_shortauthor\endcsname\c__notesslides_shortauthor\else%
7575   \PackageWarning{notesslides}{Author/Source-undefined-in-copyright-notice}%
7576   ?source/author?\fi%
7577 \fi}
7578 \newsavebox{\cclogo}
7579 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
7580 \newif\ifcchref\cchreffalse
7581 \AtBeginDocument{
7582   \@ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
7583 }
7584 \def\licensing{
7585   \ifcchref
7586     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
7587   \else
7588     {\usebox{\cclogo}}

```

```

7589 \fi
7590 }
7591 \newrobustcmd{\setlicensing}[2][]{
7592   \def\@url{#1}
7593   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
7594   \ifx\@url\@empty
7595     \def\licensing{\usebox{\cclogo}}
7596   \else
7597     \def\licensing{
7598       \ifcchref
7599         \href{#1}{\usebox{\cclogo}}
7600       \else
7601         {\usebox{\cclogo}}
7602       \fi
7603     }
7604   \fi
7605 }

```

(End definition for \setlicensing. This function is documented on page 61.)

EdN:11

```

\slidelabel Now, we set up the slide label for the article mode.11
7606 \newrobustcmd\miko@slidelabel{
7607   \vbox to \slidelogoheight{
7608     \vss\hbox to \slidewidth
7609       {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
7610   }
7611 }

```

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

## 38.4 Frame Images

**\frameimage** We have to make sure that the width is overwritten, for that we check the \Gin@ewidth macro from the graphicx package. We also add the label key.

```

7612 \def\Gin@mhrepos{}
7613 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7614 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
7615 \newrobustcmd\frameimage[2][]{
7616   \stepcounter{slide}
7617   \bool_if:NT \c__notesslides_frameimages_bool {
7618     \def\Gin@ewidth{}\setkeys{Gin}{#1}
7619     \bool_if:NF \c__notesslides_notes_bool { \vfill }
7620     \begin{center}
7621       \bool_if:NTF \c__notesslides_fiboxed_bool {
7622         \fbox{
7623           \ifx\Gin@ewidth\@empty
7624             \ifx\Gin@mhrepos\@empty
7625               \mhgraphics[width=\slidewidth,#1]{#2}
7626             \else
7627               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7628             \fi
7629           \else% Gin@ewidth empty

```

<sup>11</sup>EdNOTE: see that we can use the themes for the slides some day. This is all fake.

```

7630         \ifx\Gin@mhrepos\@empty
7631         \mhgraphics[#1]{#2}
7632     \else
7633         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7634     \fi
7635 \fi% Gin@ewidth empty
7636 }
7637 }{
7638     \ifx\Gin@ewidth\@empty
7639     \ifx\Gin@mhrepos\@empty
7640         \mhgraphics[width=\slidewidth,#1]{#2}
7641     \else
7642         \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7643     \fi
7644     \ifx\Gin@mhrepos\@empty
7645         \mhgraphics[#1]{#2}
7646     \else
7647         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7648     \fi
7649     \fi% Gin@ewidth empty
7650 }
7651 \end{center}
7652 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
7653 \bool_if:NF \c__notesslides_notes_bool { \vfill }
7654 }
7655 } % ifmks@sty@frameimages

```

(End definition for `\frameimage`. This function is documented on page 61.)

## 38.5 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`.

```

7656 \stex_html_backend:F {
7657     \bool_if:NT \c__notesslides_sectocframes_bool {
7658         \str_if_eq:VnTF \__notesslidestopsect{part}{
7659             \newcounter{chapter}\counterwithin*{section}{chapter}
7660         }{
7661             \str_if_eq:VnT\__notesslidestopsect{chapter}{
7662                 \newcounter{chapter}\counterwithin*{section}{chapter}
7663             }
7664         }
7665     }
7666 }

```

`\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`

```

7667 \def\part@prefix{}
7668 \@ifpackageloaded{document-structure}{
7669     \str_case:VnF \__notesslidestopsect {

```

```

7670 {part}{
7671   \int_set:Nn \l_document_structure_section_level_int {0}
7672   \def\thesection{\arabic{chapter}.\arabic{section}}
7673   \def\part@prefix{\arabic{chapter}.}
7674 }
7675 {chapter}{
7676   \int_set:Nn \l_document_structure_section_level_int {1}
7677   \def\thesection{\arabic{chapter}.\arabic{section}}
7678   \def\part@prefix{\arabic{chapter}.}
7679 }
7680 }{
7681   \int_set:Nn \l_document_structure_section_level_int {2}
7682   \def\part@prefix{}
7683 }
7684 }
7685
7686 \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 (env.)`

```

7687 \renewenvironment{sfragment}[2][]{
7688   \__document_structure_sfragment_args:n { #1 }
7689   \int_incr:N \l_document_structure_section_level_int
7690   \bool_if:NT \c__notesslides_sectocframes_bool {
7691     \stepcounter{slide}
7692     \begin{frame}[noframenumbering]
7693     \vfill\Large\centering
7694     \red{
7695       \ifcase\l_document_structure_section_level_int\or
7696         \stepcounter{part}
7697         \def\__notesslideslabel{\omdoc@part@kw}~\Roman{part}}
7698         \addcontentsline{toc}{part}{\protect\numberline{\thepart}#2}
7699         \pdfbookmark[0]{\thepart\ #2}{part.\thepart}
7700         \def\currentsectionlevel{\omdoc@part@kw}
7701       \or
7702         \stepcounter{chapter}
7703         \def\__notesslideslabel{\omdoc@chapter@kw}~\arabic{chapter}}
7704         \addcontentsline{toc}{chapter}{\protect\numberline{\thechapter}#2}
7705         \pdfbookmark[1]{\thechapter\ #2}{chapter.\cs_if_exist:cT{\thepart}\thepart.\thechapter}
7706         \def\currentsectionlevel{\omdoc@chapter@kw}
7707       \or
7708         \stepcounter{section}
7709         \def\__notesslideslabel{\part@prefix\arabic{section}}
7710         \addcontentsline{toc}{section}{\protect\numberline{\thesection}#2}
7711         \pdfbookmark[2]{\cs_if_exist:cT{\thechapter}{\thechapter}.\thesection\ #2}
7712         {section.\cs_if_exist:cT{\thepart}{\thepart}.\cs_if_exist:cT{\thechapter}{\thechapter}.\thepart}
7713         \def\currentsectionlevel{\omdoc@section@kw}
7714       \or
7715         \stepcounter{subsection}
7716         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
7717         \addcontentsline{toc}{subsection}{\protect\numberline{\thesubsection}#2}

```

```

7718         \pdfbookmark[3]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection
7719         {subsection.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection}
7720         \def\currentsectionlevel{\omdoc@subsection@kw}
7721     \or
7722         \stepcounter{subsubsection}
7723         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
7724         \addcontentsline{toc}{subsubsection}{\protect\numberline{\thesubsubsection}#2}
7725         \pdfbookmark[4]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsubsection}
7726         {subsubsection.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsubsection}
7727         \def\currentsectionlevel{\omdoc@subsubsection@kw}
7728     \or
7729         \stepcounter{paragraph}
7730         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{paragraph}}
7731         \addcontentsline{toc}{paragraph}{\protect\numberline{\theparagraph}#2}
7732         \pdfbookmark[5]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsubsection}
7733         {paragraph.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsubsection}
7734         \def\currentsectionlevel{\omdoc@paragraph@kw}
7735     \else
7736         \def\__notesslideslabel{}
7737         \def\currentsectionlevel{\omdoc@paragraph@kw}
7738     \fi% end ifcase
7739     \__notesslideslabel\quad #2%
7740 }%
7741 \vfill%
7742 \end{frame}%
7743 }
7744 \str_if_empty:NF \l__document_structure_sfragment_id_str {
7745     \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
7746 }
7747 }{}
7748 }

```

We set up a beamer template for theorems like ams style, but without a block environment.

```

7749 \def\inserttheorembodyfont{\normalfont}
7750 %\bool_if:NF \c__notesslides_notes_bool {
7751 % \defbeamertemplate{theorem begin}{miko}
7752 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
7753 % \ifx\inserttheoremaddition@empty\else\ (\inserttheoremaddition)\fi%
7754 % \inserttheorempunctuation\inserttheorembodyfont\xspace}
7755 % \defbeamertemplate{theorem end}{miko}{\}

```

and we set it as the default one.

```

7756 % \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.

```

7757 % \expandafter\def\csname Parent2\endcsname{}
7758 %}
7759
7760 \AddToHook{begindocument}{ % this does not work for some reason
7761     \setbeamertemplate{theorems}[ams style]
7762 }
7763 \bool_if:NT \c__notesslides_notes_bool {
7764     \renewenvironment{columns}[1][\]{}{}

```



```

7765     \par\noindent%
7766     \begin{minipage}%
7767     \slidewidth\centering\leavevmode%
7768   }{%
7769     \end{minipage}\par\noindent%
7770   }%
7771   \newsavebox\columnbox%
7772   \renewenvironment<>{column}[2][]{%
7773     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
7774   }{%
7775     \end{minipage}\end{lrbox}\usebox\columnbox%
7776   }%
7777 }

7778 \bool_if:NTF \c__notesslides_noproblems_bool {
7779   \newenvironment{problems}{}{}
7780 }{
7781   \excludacomment{problems}
7782 }

```

## 38.6 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.

```

7783 \gdef\printexcursions{}
7784 \newcommand\excursionref[2]{% label, text
7785   \bool_if:NT \c__notesslides_notes_bool {
7786     \begin{sparagraph}[title=Excursion]
7787       #2 \sref[fallback=the appendix]{#1}.
7788     \end{sparagraph}
7789   }
7790 }
7791 \newcommand\activate@excursion[2][{}{
7792   \gappto\printexcursions{\inputref{#1}{#2}}
7793 }
7794 \newcommand\excursion[4][{}{ repos, label, path, text
7795   \bool_if:NT \c__notesslides_notes_bool {
7796     \activate@excursion{#1}{#3}\excursionref{#2}{#4}
7797   }
7798 }

```

(End definition for `\excursion`. This function is documented on page 62.)

**\excursiongroup**

```

7799 \keys_define:nn{notesslides / excursiongroup }{
7800   id          .str_set_x:N = \l__notesslides_excursion_id_str,
7801   intro       .tl_set:N   = \l__notesslides_excursion_intro_tl,
7802   mhrepos     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
7803 }
7804 \cs_new_protected:Nn \__notesslides_excursion_args:n {
7805   \tl_clear:N \l__notesslides_excursion_intro_tl
7806   \str_clear:N \l__notesslides_excursion_id_str

```

```

7807 \str_clear:N \l__notesslides_excursion_mhrepos_str
7808 \keys_set:nn {notesslides / excursionsgroup }{ #1 }
7809 }
7810 \newcommand\excursionsgroup[1][]{
7811   \__notesslides_excursion_args:n{ #1 }
7812   \ifdefempty\printexcursions{}% only if there are excursions
7813   {\begin{note}
7814     \begin{sfragment}[#1]{Excursions}%
7815     \ifdefempty\l__notesslides_excursion_intro_tl}{
7816       \inputref[\l__notesslides_excursion_mhrepos_str]{
7817         \l__notesslides_excursion_intro_tl
7818       }
7819     }
7820     \printexcursions%
7821     \end{sfragment}
7822   \end{note}}
7823 }
7824 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
7825 \end{package}

```

(End definition for \excursionsgroup. This function is documented on page 62.)

## Chapter 39

# The Implementation

### 39.1 Package Options

The first step is to declare (a few) package options that handle whether certain information is printed or not. They all come with their own conditionals that are set by the options.

```
7826 <*package>
7827 <@@=problems>
7828 \ProvidesExplPackage{problem}{2022/08/08}{3.2.0}{Semantic Markup for Problems}
7829 \RequirePackage{13keys2e}
7830 \RequirePackage{amssymb}% for \Box
7831
7832 \keys_define:nn { problem / pkg }{
7833   notes      .default:n    = { true },
7834   notes      .bool_set:N   = \c__problems_notes_bool,
7835   gnotes     .default:n    = { true },
7836   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
7837   hints      .default:n    = { true },
7838   hints      .bool_set:N   = \c__problems_hints_bool,
7839   solutions  .default:n    = { true },
7840   solutions  .bool_set:N   = \c__problems_solutions_bool,
7841   pts        .default:n    = { true },
7842   pts        .bool_set:N   = \c__problems_pts_bool,
7843   min        .default:n    = { true },
7844   min        .bool_set:N   = \c__problems_min_bool,
7845   boxed      .default:n    = { true },
7846   boxed      .bool_set:N   = \c__problems_boxed_bool,
7847   unknown    .code:n       = {
7848     \PassOptionsToPackage{\CurrentOption}{stex}
7849   }
7850 }
7851 \newif\ifsolutions
7852
7853 \ProcessKeysOptions{ problem / pkg }
7854 \bool_if:NTF \c__problems_solutions_bool {
7855   \solutionstrue
7856 }{
7857   \solutionsfalse
```

```

7858 }
7859 \RequirePackage{stex}

```

Then we make sure that the necessary packages are loaded (in the right versions).

```

7860 \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.

```

7861 \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.

```

7862 \def\prob@problem@kw{Problem}
7863 \def\prob@solution@kw{Solution}
7864 \def\prob@hint@kw{Hint}
7865 \def\prob@note@kw{Note}
7866 \def\prob@gnote@kw{Grading}
7867 \def\prob@pt@kw{pt}
7868 \def\prob@min@kw{min}
7869 \def\prob@correct@kw{Correct}
7870 \def\prob@wrong@kw{Wrong}

```

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

For the other languages, we set up triggers

```

7871 \AddToHook{begindocument}{
7872   \ltx@ifpackageloaded{babel}{
7873     \makeatletter
7874     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7875     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
7876       \input{problem-ngerman.ldf}
7877     }
7878     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
7879       \input{problem-finnish.ldf}
7880     }
7881     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
7882       \input{problem-french.ldf}
7883     }
7884     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
7885       \input{problem-russian.ldf}
7886     }
7887     \makeatother
7888   }{}
7889 }

```

## 39.2 Problems and Solutions

We now prepare the KeyVal support for problems. The key macros just set appropriate internal macros.

```

7890 \keys_define:nn{ problem / problem }{
7891   id      .str_set_x:N = \l__problems_prob_id_str,
7892   pts     .tl_set:N    = \l__problems_prob_pts_tl,
7893   min     .tl_set:N    = \l__problems_prob_min_tl,

```

```

7894 title .tl_set:N = \l__problems_prob_title_tl,
7895 type .tl_set:N = \l__problems_prob_type_tl,
7896 imports .tl_set:N = \l__problems_prob_imports_tl,
7897 name .str_set_x:N = \l__problems_prob_name_str,
7898 refnum .int_set:N = \l__problems_prob_refnum_int
7899 }
7900 \cs_new_protected:Nn \l__problems_prob_args:n {
7901 \str_clear:N \l__problems_prob_id_str
7902 \str_clear:N \l__problems_prob_name_str
7903 \tl_clear:N \l__problems_prob_pts_tl
7904 \tl_clear:N \l__problems_prob_min_tl
7905 \tl_clear:N \l__problems_prob_title_tl
7906 \tl_clear:N \l__problems_prob_type_tl
7907 \tl_clear:N \l__problems_prob_imports_tl
7908 \int_zero_new:N \l__problems_prob_refnum_int
7909 \keys_set:nn { problem / problem }{ #1 }
7910 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
7911 \let\l__problems_prob_refnum_int\undefined
7912 }
7913 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

7914 \newcounter{problem}[section]
7915 \newcommand\numberproblemsin[1]{\@addtoreset{problem}{#1}}
7916 \def\theplainsproblem{\arabic{problem}}
7917 \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.

```

7918 \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

```

7919 \newcommand\prob@number{
7920 \int_if_exist:NTF \l__problems_inclprob_refnum_int {
7921 \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
7922 }{
7923 \int_if_exist:NTF \l__problems_prob_refnum_int {
7924 \prob@label{\int_use:N \l__problems_prob_refnum_int }
7925 }{
7926 \prob@label\theplainsproblem
7927 }
7928 }
7929 }
7930 \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.

```

7931 \newcommand\prob@title[3]{%
7932   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
7933     #2 \l__problems_inclprob_title_tl #3
7934   }{
7935     \tl_if_empty:NTF \l__problems_prob_title_tl {
7936       #1
7937     }{
7938       #2 \l__problems_prob_title_tl #3
7939     }
7940   }
7941 }

```

(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.

```

7942 \def\prob@heading{
7943   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
7944   %\sref@label{id{\prob@problem@kw~\prob@number}}{~}
7945 }

```

(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 (env.)**

```

7946 \newenvironment{sproblem}[1][]{
7947   \__problems_prob_args:n{#1}%\sref@target%
7948   \@in@omtexttrue% we are in a statement (for inline definitions)
7949   \refstepcounter{sproblem}\record@problem
7950   \def\current@section@level{\prob@problem@kw}
7951
7952   \str_if_empty:NT \l__problems_prob_name_str {
7953     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
7954     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
7955     \seq_get_left:NN \l_tmpa_seq \l__problems_prob_name_str
7956   }
7957
7958   \stex_if_do_html:T{
7959     \tl_if_empty:NF \l__problems_prob_title_tl {
7960       \exp_args:No \stex_document_title:n \l__problems_prob_title_tl
7961     }
7962   }
7963
7964   \exp_args:Nno\stex_module_setup:nn{type=problem}\l__problems_prob_name_str
7965
7966   \stex_reactivate_macro:N \STEXexport
7967   \stex_reactivate_macro:N \importmodule
7968   \stex_reactivate_macro:N \symdecl
7969   \stex_reactivate_macro:N \notation
7970   \stex_reactivate_macro:N \symdef

```

```

7971 \stex_if_do_html:T{
7972   \begin{stex_annotate_env} {problem} {
7973     \l_stex_module_ns_str ? \l_stex_module_name_str
7974   }
7975
7976   \stex_annotate_invisible:nnn{header}{} {
7977     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
7978     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
7979     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
7980       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
7981     }
7982   }
7983 }
7984 }
7985
7986 \stex_csl_to_imports:No \importmodule \l__problems_prob_imports_tl
7987
7988
7989 \tl_if_exist:NTF \l__problems_inclprob_type_tl {
7990   \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
7991 }{
7992   \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
7993 }
7994 \str_if_exist:NTF \l__problems_inclprob_id_str {
7995   \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
7996 }{
7997   \str_set_eq:NN \sproblemid \l__problems_prob_id_str
7998 }
7999
8000
8001 \stex_if_smsmode:F {
8002   \clist_set:No \l_tmpa_clist \sproblemtype
8003   \tl_clear:N \l_tmpa_tl
8004   \clist_map_inline:Nn \l_tmpa_clist {
8005     \tl_if_exist:cT {__problems_sproblem_##1_start:}{
8006       \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_start:}}
8007     }
8008   }
8009   \tl_if_empty:NTF \l_tmpa_tl {
8010     \__problems_sproblem_start:
8011   }{
8012     \l_tmpa_tl
8013   }
8014 }
8015 \stex_ref_new_doc_target:n \sproblemid
8016 \stex_if_smsmode:TF \stex_smsmode_do: \ignorespacesandpars
8017 }{
8018   \__stex_modules_end_module:
8019   \stex_if_smsmode:F{
8020     \clist_set:No \l_tmpa_clist \sproblemtype
8021     \tl_clear:N \l_tmpa_tl
8022     \clist_map_inline:Nn \l_tmpa_clist {
8023       \tl_if_exist:cT {__problems_sproblem_##1_end:}{
8024         \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_end:}}

```

```

8025     }
8026   }
8027   \tl_if_empty:NTF \l_tmpa_tl {
8028     \__problems_sproblem_end:
8029   }{
8030     \l_tmpa_tl
8031   }
8032 }
8033 \stex_if_do_html:T{
8034   \end{stex_annotate_env}
8035 }
8036
8037 \smallskip
8038 }
8039
8040 \seq_put_right:Nx\g_stex_smsmode_allowedenvs_seq{\tl_to_str:n{sproblem}}
8041
8042
8043
8044 \cs_new_protected:Nn \__problems_sproblem_start: {
8045   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
8046 }
8047 \cs_new_protected:Nn \__problems_sproblem_end: { \par\smallskip}
8048
8049 \newcommand\stexpatchproblem[3][] {
8050   \str_set:Nx \l_tmpa_str{ #1 }
8051   \str_if_empty:NTF \l_tmpa_str {
8052     \tl_set:Nn \__problems_sproblem_start: { #2 }
8053     \tl_set:Nn \__problems_sproblem_end: { #3 }
8054   }{
8055     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
8056     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
8057   }
8058 }
8059
8060
8061 \bool_if:NT \c__problems_boxed_bool {
8062   \surroundwithhmdframed{problem}
8063 }

```

**\record@problem** This macro records information about the problems in the \*.aux file.

```

8064 \def\record@problem{
8065   \protected@write\auxout{}
8066   {
8067     \string\@problem{\prob@number}
8068     {
8069       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
8070         \l__problems_inclprob_pts_tl
8071       }{
8072         \l__problems_prob_pts_tl
8073       }
8074     }%
8075     {
8076       \tl_if_exist:NTF \l__problems_inclprob_min_tl {

```



```

8077         \l__problems_inclprob_min_tl
8078     }{
8079         \l__problems_prob_min_tl
8080     }
8081 }
8082 }
8083 }

```

(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).

```

8084 \def\@problem#1#2#3{

```

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

**solution (env.)** 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.

```

8085 \keys_define:nn { problem / solution }{
8086   id          .str_set_x:N = \l__problems_solution_id_str ,
8087   for         .str_set_x:N = \l__problems_solution_for_str ,
8088   type       .str_set_x:N = \l__problems_solution_type_str ,
8089   title      .tl_set:N     = \l__problems_solution_title_tl
8090 }
8091 \cs_new_protected:Nn \__problems_solution_args:n {
8092   \str_clear:N \l__problems_solution_id_str
8093   \str_clear:N \l__problems_solution_type_str
8094   \str_clear:N \l__problems_solution_for_str
8095   \tl_clear:N \l__problems_solution_title_tl
8096   \keys_set:nn { problem / solution }{ #1 }
8097 }

```

**\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.

```

8098 \box_new:N \l__problems_solution_box
8099 \newenvironment{solution}[1][{}]{
8100   \__problems_solution_args:n{#1}
8101   \stex_html_backend:TF{
8102     \stex_if_do_html:T{
8103       \begin{stex_annotate_env}{solution}{}}
8104       \str_if_empty:NF \l__problems_solution_type_str {
8105         \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{}}
8106     }
8107     \noindent\textbf{Solution}\tl_if_empty:NF\l__problems_solution_title_tl{~(\l__problem
8108   }
8109   }{
8110     \setbox\l__problems_solution_box\vbox\bgroup
8111     \par\smallskip\hrule\smallskip
8112     \noindent\textbf{Solution}\tl_if_empty:NF\l__problems_solution_title_tl{~(\l__problems
8113   }
8114   }{
8115     \stex_html_backend:TF{
8116       \stex_if_do_html:T{
8117         \end{stex_annotate_env}

```

```

8118     }
8119   }{
8120     \smallskip\hrule
8121     \egroup
8122     \bool_if:NT \c__problems_solutions_bool {
8123       \box\l__problems_solution_box
8124     }
8125   }
8126 }
8127
8128 \newcommand\startsolutions{
8129   \bool_set_true:N \c__problems_solutions_bool
8130   \solutionstrue
8131   % \specialcomment{solution}{\@startsolution}{
8132   %   \bool_if:NF \c__problems_boxed_bool {
8133   %     \hrule\medskip
8134   %   }
8135   %   \end{small}%
8136   % }
8137   % \bool_if:NT \c__problems_boxed_bool {
8138   %   \surroundwithmdframed{solution}
8139   % }
8140 }

```

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

## \stopsolutions

```

8141 \newcommand\stopsolutions{\bool_set_false:N \c__problems_solutions_bool \solutionsfalse}%\ex

```

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

## exnote (env.)

```

8142 \bool_if:NTF \c__problems_notes_bool {
8143   \newenvironment{exnote}[1][]{
8144     \par\smallskip\hrule\smallskip
8145     \noindent\textbf{\prob@note@kw :~ }\small
8146   }{
8147     \smallskip\hrule
8148   }
8149 }{
8150   \excludacomment{exnote}
8151 }

```

## hint (env.)

```

8152 \bool_if:NTF \c__problems_notes_bool {
8153   \newenvironment{hint}[1][]{
8154     \par\smallskip\hrule\smallskip
8155     \noindent\textbf{\prob@hint@kw :~ }\small
8156   }{
8157     \smallskip\hrule
8158   }
8159   \newenvironment{exhint}[1][]{
8160     \par\smallskip\hrule\smallskip
8161     \noindent\textbf{\prob@hint@kw :~ }\small

```

```

8162 }{
8163   \smallskip\hrule
8164 }
8165 }{
8166   \excludecomment{hint}
8167   \excludecomment{exhint}
8168 }

```

**gnote (env.)**

```

8169 \bool_if:NTF \c__problems_notes_bool {
8170   \newenvironment{gnote}[1][]{
8171     \par\smallskip\hrule\smallskip
8172     \noindent\textbf{\prob@gnote@kw :~ }\small
8173   }{
8174     \smallskip\hrule
8175   }
8176 }{
8177   \excludecomment{gnote}
8178 }

```

### 39.3 Marup for Added Value Services

### 39.4 Multiple Choice Blocks

EdN:12

**mcb (env.)** <sup>12</sup>

```

8179 \newenvironment{mcb}{
8180   \begin{enumerate}
8181 }{
8182   \end{enumerate}
8183 }

```

we define the keys for the mcb macro

```

8184 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
8185   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
8186     \bool_set_true:N #1
8187   }{
8188     \bool_set_false:N #1
8189   }
8190 }
8191 \keys_define:nn { problem / mcb }{
8192   id          .str_set_x:N = \l__problems_mcc_id_str ,
8193   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
8194   T           .default:n   = { false } ,
8195   T           .bool_set:N  = \l__problems_mcc_t_bool ,
8196   F           .default:n   = { false } ,
8197   F           .bool_set:N  = \l__problems_mcc_f_bool ,
8198   Ttext       .tl_set:N    = \l__problems_mcc_Ttext_tl ,
8199   Ftext       .tl_set:N    = \l__problems_mcc_Ftext_tl
8200 }
8201 \cs_new_protected:Nn \l__problems_mcc_args:n {

```

---

<sup>12</sup>EdNOTE: MK: maybe import something better here from a dedicated MC package

```

8202 \str_clear:N \l__problems_mcc_id_str
8203 \tl_clear:N \l__problems_mcc_feedback_tl
8204 \bool_set_false:N \l__problems_mcc_t_bool
8205 \bool_set_false:N \l__problems_mcc_f_bool
8206 \tl_clear:N \l__problems_mcc_Ttext_tl
8207 \tl_clear:N \l__problems_mcc_Ftext_tl
8208 \str_clear:N \l__problems_mcc_id_str
8209 \keys_set:nn { problem / mcc }{ #1 }
8210 }

```

**\mcc**

```

8211 \def\mccTrueText{\textbf{\prob@correct@kw!~}}
8212 \def\mccFalseText{\textbf{\prob@wrong@kw!~}}
8213 \newcommand\mcc[2][] {
8214   \l__problems_mcc_args:n{ #1 }
8215   \item[$\Box$] #2
8216   \bool_if:NT \c__problems_solutions_bool{
8217     \
8218     \bool_if:NT \l__problems_mcc_t_bool {
8219       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccTrueText\l__problems_mcc_Ttext_tl
8220     }
8221     \bool_if:NT \l__problems_mcc_f_bool {
8222       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccFalseText\l__problems_mcc_Ftext_tl
8223     }
8224     \tl_if_empty:NF \l__problems_mcc_feedback_tl {
8225       \emph{\l__problems_mcc_feedback_tl}
8226     }
8227   }
8228 } %solutions

```

(End definition for \mcc. This function is documented on page 65.)

## 39.5 Filling in Concrete Solutions

**\includeproblem** This is embarrassingly simple, but can grow over time.

```

8229 \newcommand\fillinsol[1]{\quad%
8230   \ifsolutions\textcolor{red}{\#1!}\else%
8231   \fbox{\phantom{\huge{\#1}}}%
8232   \fi}

```

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

## 39.6 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.

```

8233
8234 \keys_define:nn{ problem / inclproblem }{
8235   id      .str_set_x:N = \l__problems_inclprob_id_str,
8236   pts     .tl_set:N    = \l__problems_inclprob_pts_tl,
8237   min     .tl_set:N    = \l__problems_inclprob_min_tl,

```

```

8238 title .tl_set:N = \l__problems_inclprob_title_tl,
8239 refnum .int_set:N = \l__problems_inclprob_refnum_int,
8240 type .tl_set:N = \l__problems_inclprob_type_tl,
8241 mhrepos .str_set_x:N = \l__problems_inclprob_mhrepos_str
8242 }
8243 \cs_new_protected:Nn \l__problems_inclprob_args:n {
8244 \str_clear:N \l__problems_prob_id_str
8245 \tl_clear:N \l__problems_inclprob_pts_tl
8246 \tl_clear:N \l__problems_inclprob_min_tl
8247 \tl_clear:N \l__problems_inclprob_title_tl
8248 \tl_clear:N \l__problems_inclprob_type_tl
8249 \int_zero_new:N \l__problems_inclprob_refnum_int
8250 \str_clear:N \l__problems_inclprob_mhrepos_str
8251 \keys_set:nn { problem / inclproblem }{ #1 }
8252 \tl_if_empty:NT \l__problems_inclprob_pts_tl {
8253 \let\l__problems_inclprob_pts_tl\undefined
8254 }
8255 \tl_if_empty:NT \l__problems_inclprob_min_tl {
8256 \let\l__problems_inclprob_min_tl\undefined
8257 }
8258 \tl_if_empty:NT \l__problems_inclprob_title_tl {
8259 \let\l__problems_inclprob_title_tl\undefined
8260 }
8261 \tl_if_empty:NT \l__problems_inclprob_type_tl {
8262 \let\l__problems_inclprob_type_tl\undefined
8263 }
8264 \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
8265 \let\l__problems_inclprob_refnum_int\undefined
8266 }
8267 }
8268
8269 \cs_new_protected:Nn \l__problems_inclprob_clear: {
8270 \let\l__problems_inclprob_id_str\undefined
8271 \let\l__problems_inclprob_pts_tl\undefined
8272 \let\l__problems_inclprob_min_tl\undefined
8273 \let\l__problems_inclprob_title_tl\undefined
8274 \let\l__problems_inclprob_type_tl\undefined
8275 \let\l__problems_inclprob_refnum_int\undefined
8276 \let\l__problems_inclprob_mhrepos_str\undefined
8277 }
8278 \l__problems_inclprob_clear:
8279
8280 \newcommand\includeproblem[2][]{
8281 \l__problems_inclprob_args:n{ #1 }
8282 \exp_args:No \stex_in_repository:nn\l__problems_inclprob_mhrepos_str{
8283 \stex_html_backend:TF {
8284 \str_clear:N \l_tmpa_str
8285 \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
8286 \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
8287 }
8288 \stex_annotate_invisible:nnn{includeproblem}{
8289 \l_tmpa_str / #2
8290 }{}}
8291 }{

```

```

8292     \begingroup
8293     \inputreftrue
8294     \tl_if_empty:nTF{ ##1 }{
8295       \input{#2}
8296     }{
8297       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
8298     }
8299   \endgroup
8300 }
8301 }
8302 \__problems_inclprob_clear:
8303 }

```

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

## 39.7 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.

```

8304 \AddToHook{enddocument}{
8305   \bool_if:NT \c__problems_pts_bool {
8306     \message{Total:~\arabic{pts}~points}
8307   }
8308   \bool_if:NT \c__problems_min_bool {
8309     \message{Total:~\arabic{min}~minutes}
8310   }
8311 }

```

The margin pars are reader-visible, so we need to translate

```

8312 \def\pts#1{
8313   \bool_if:NT \c__problems_pts_bool {
8314     \marginpar{#1~\prob@pt@kw}
8315   }
8316 }
8317 \def\min#1{
8318   \bool_if:NT \c__problems_min_bool {
8319     \marginpar{#1~\prob@min@kw}
8320   }
8321 }

```

`\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.

```

8322 \newcounter{pts}
8323 \def\show@pts{
8324   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
8325     \bool_if:NT \c__problems_pts_bool {
8326       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
8327       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
8328     }
8329   }{
8330     \tl_if_exist:NT \l__problems_prob_pts_tl {
8331       \bool_if:NT \c__problems_pts_bool {

```

```

8332         \tl_if_empty:NT\l__problems_prob_pts_tl{
8333             \tl_set:Nn \l__problems_prob_pts_tl {0}
8334         }
8335         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
8336         \addtocounter{pts}{\l__problems_prob_pts_tl}
8337     }
8338 }
8339 }
8340 }

```

(End definition for \show@pts. This function is documented on page ??.)  
and now the same for the minutes

\show@min

```

8341 \newcounter{min}
8342 \def\show@min{
8343     \tl_if_exist:NTF \l__problems_inclprob_min_tl {
8344         \bool_if:NT \c__problems_min_bool {
8345             \marginpar{\l__problems_inclprob_pts_tl\ min}
8346             \addtocounter{min}{\l__problems_inclprob_min_tl}
8347         }
8348     }{
8349         \tl_if_exist:NT \l__problems_prob_min_tl {
8350             \bool_if:NT \c__problems_min_bool {
8351                 \tl_if_empty:NT\l__problems_prob_min_tl{
8352                     \tl_set:Nn \l__problems_prob_min_tl {0}
8353                 }
8354                 \marginpar{\l__problems_prob_min_tl\ min}
8355                 \addtocounter{min}{\l__problems_prob_min_tl}
8356             }
8357         }
8358     }
8359 }
8360 \end{package}

```

(End definition for \show@min. This function is documented on page ??.)

## Chapter 40

# Implementation: The hwexam Package

### 40.1 Package Options

The first step is to declare (a few) package options that handle whether certain information is printed or not. Some come with their own conditionals that are set by the options, the rest is just passed on to the `problems` package.

```
8361 {*package}
8362 \ProvidesExplPackage{hwexam}{2022/08/08}{3.2.0}{homework assignments and exams}
8363 \RequirePackage{13keys2e}
8364
8365 \newif\iftest\testfalse
8366 \DeclareOption{test}{\testtrue}
8367 \newif\ifmultiple\multiplefalse
8368 \DeclareOption{multiple}{\multipletrue}
8369 \DeclareOption{lang}{\PassOptionsToPackage{\CurrentOption}{problem}}
8370 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
8371 \ProcessOptions
```

Then we make sure that the necessary packages are loaded (in the right versions).

```
8372 \RequirePackage{keyval}[1997/11/10]
8373 \RequirePackage{problem}
```

`\hwexam@*@kw` For multilinguality, we define internal macros for keywords that can be specialized in `*.ldf` files.

```
8374 \newcommand\hwexam@assignment@kw{Assignment}
8375 \newcommand\hwexam@given@kw{Given}
8376 \newcommand\hwexam@due@kw{Due}
8377 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~blank~for~extra~space}
8378 \newcommand\hwexam@minutes@kw{minutes}
8379 \newcommand\correction@probs@kw{prob.}
8380 \newcommand\correction@pts@kw{total}
8381 \newcommand\correction@reached@kw{reached}
8382 \newcommand\correction@sum@kw{Sum}
8383 \newcommand\correction@grade@kw{grade}
8384 \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

```

8385 \AddToHook{begindocument}{
8386 \ltx@ifpackageloaded{babel}{
8387 \makeatletter
8388 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
8389 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
8390 \input{hwexam-ngerman.ldf}
8391 }
8392 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
8393 \input{hwexam-finnish.ldf}
8394 }
8395 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
8396 \input{hwexam-french.ldf}
8397 }
8398 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
8399 \input{hwexam-russian.ldf}
8400 }
8401 \makeatother
8402 }{}
8403 }
8404

```

## 40.2 Assignments

Then we set up a counter for problems and make the problem counter inherited from `problem.sty` depend on it. Furthermore, we specialize the `\prob@label` macro to take the assignment counter into account.

```

8405 \newcounter{assignment}
8406 %\numberproblemsin{assignment}

We will prepare the keyval support for the assignment environment.

8407 \keys_define:nn { hwexam / assignment } {
8408 id .str_set:N = \l_@@_assign_id_str,
8409 number .int_set:N = \l_@@_assign_number_int,
8410 title .tl_set:N = \l_@@_assign_title_tl,
8411 type .tl_set:N = \l_@@_assign_type_tl,
8412 given .tl_set:N = \l_@@_assign_given_tl,
8413 due .tl_set:N = \l_@@_assign_due_tl,
8414 loadmodules .code:n = {
8415 \bool_set_true:N \l_@@_assign_loadmodules_bool
8416 }
8417 }
8418 \cs_new_protected:Nn \_@@_assignment_args:n {
8419 \str_clear:N \l_@@_assign_id_str
8420 \int_set:Nn \l_@@_assign_number_int {-1}
8421 \tl_clear:N \l_@@_assign_title_tl
8422 \tl_clear:N \l_@@_assign_type_tl
8423 \tl_clear:N \l_@@_assign_given_tl
8424 \tl_clear:N \l_@@_assign_due_tl
8425 \bool_set_false:N \l_@@_assign_loadmodules_bool
8426 \keys_set:nn { hwexam / assignment }{ #1 }
8427 }

```

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.

```

8428 \newcommand\given@due[2]{
8429 \bool_lazy_all:nF {
8430 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
8431 { \tl_if_empty_p:V \l_@@_assign_given_tl }
8432 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
8433 { \tl_if_empty_p:V \l_@@_assign_due_tl }
8434 }{ #1 }
8435
8436 \tl_if_empty:NTF \l_@@_inclasssign_given_tl {
8437 \tl_if_empty:NF \l_@@_assign_given_tl {
8438 \hwexam@given@kw\xspace\l_@@_assign_given_tl
8439 }
8440 }{
8441 \hwexam@given@kw\xspace\l_@@_inclasssign_given_tl
8442 }
8443
8444 \bool_lazy_or:nnF {
8445 \bool_lazy_and_p:nn {
8446 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
8447 }{
8448 \tl_if_empty_p:V \l_@@_assign_due_tl
8449 }
8450 }{
8451 \bool_lazy_and_p:nn {
8452 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
8453 }{
8454 \tl_if_empty_p:V \l_@@_assign_due_tl
8455 }
8456 }{ ,~ }
8457
8458 \tl_if_empty:NTF \l_@@_inclasssign_due_tl {
8459 \tl_if_empty:NF \l_@@_assign_due_tl {
8460 \hwexam@due@kw\xspace \l_@@_assign_due_tl
8461 }
8462 }{
8463 \hwexam@due@kw\xspace \l_@@_inclasssign_due_tl
8464 }
8465
8466 \bool_lazy_all:nF {
8467 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
8468 { \tl_if_empty_p:V \l_@@_assign_given_tl }
8469 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
8470 { \tl_if_empty_p:V \l_@@_assign_due_tl }
8471 }{ #2 }
8472 }

```

`\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.

```

8473 \newcommand\assignment@title[3]{
8474 \tl_if_empty:NTF \l_@@_inclasssign_title_tl {
8475 \tl_if_empty:NTF \l_@@_assign_title_tl {
8476 #1
8477 }{
8478 #2\l_@@_assign_title_tl#3
8479 }
8480 }{
8481 #2\l_@@_inclasssign_title_tl#3
8482 }
8483 }

```

(End definition for \assignment@title. This function is documented on page ??.)

**\assignment@number** Like \assignment@title only for the number, and no around part.

```

8484 \newcommand\assignment@number{
8485 \int_compare:nNnTF \l_@@_inclasssign_number_int = {-1} {
8486 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8487 \arabic{assignment}
8488 } {
8489 \int_use:N \l_@@_assign_number_int
8490 }
8491 }{
8492 \int_use:N \l_@@_inclasssign_number_int
8493 }
8494 }

```

(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 (env.)** For the **assignment** environment we delegate the work to the **@assignment** environment that depends on whether **multiple** option is given.

```

8495 \newenvironment{assignment}[1][]{
8496 \_@@_assignment_args:n { #1 }
8497 %\sref@target
8498 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8499 \global\stepcounter{assignment}
8500 }{
8501 \global\setcounter{assignment}{\int_use:N\l_@@_assign_number_int}
8502 }
8503 \setcounter{sproblem}{0}
8504 \renewcommand\prob@label[1]{\assignment@number.##1}
8505 \def\current@section@level{\document@hwexamtype}
8506 %\sref@label{id{\document@hwexamtype \thesection}
8507 \begin{@assignment}
8508 }{
8509 \end{@assignment}
8510 }

```

In the multi-assignment case we just use the omdoc environment for suitable sectioning.

```

8511 \def\ass@title{
8512 {\protect\document@hwexamtype}\arabic{assignment}
8513 \assignment@title{}\{;\}\{;\} -- \given@due{}\{;\}
8514 }
8515 \ifmultiple
8516 \newenvironment{@assignment}{
8517 \bool_if:NTF \l_@@_assign_loadmodules_bool {
8518 \begin{sfragment}[loadmodules]{\ass@title}
8519 }{
8520 \begin{sfragment}{\ass@title}
8521 }
8522 }{
8523 \end{sfragment}
8524 }

```

for the single-page case we make a title block from the same components.

```

8525 \else
8526 \newenvironment{@assignment}{
8527 \begin{center}\bf
8528 \Large@title\strut\
8529 \document@hwexamtype}\arabic{assignment}\assignment@title{}\{;\}\{;\}\{;\}
8530 \large\given@due{--;\}\{;\}--}
8531 \end{center}
8532 }{}
8533 \fi% multiple

```

## 40.3 Including Assignments

**\in\*assignment** This macro is essentially a glorified `\include` statement, it just sets some internal macros first that overwrite the local points. Importantly, it resets the `inclassig` keys after the input.

```

8534 \keys_define:nn { hwexam / inclassignment } {
8535 %id .str_set_x:N = \l_@@_assign_id_str,
8536 number .int_set:N = \l_@@_inclassign_number_int,
8537 title .tl_set:N = \l_@@_inclassign_title_tl,
8538 type .tl_set:N = \l_@@_inclassign_type_tl,
8539 given .tl_set:N = \l_@@_inclassign_given_tl,
8540 due .tl_set:N = \l_@@_inclassign_due_tl,
8541 mhrepos .str_set_x:N = \l_@@_inclassign_mhrepos_str
8542 }
8543 \cs_new_protected:Nn \_@@_inclassignment_args:n {
8544 \int_set:Nn \l_@@_inclassign_number_int {-1}
8545 \tl_clear:N \l_@@_inclassign_title_tl
8546 \tl_clear:N \l_@@_inclassign_type_tl
8547 \tl_clear:N \l_@@_inclassign_given_tl
8548 \tl_clear:N \l_@@_inclassign_due_tl
8549 \str_clear:N \l_@@_inclassign_mhrepos_str
8550 \keys_set:nn { hwexam / inclassignment }{ #1 }
8551 }
8552 \_@@_inclassignment_args:n {}
8553
8554 \newcommand\inputassignment[2][]{

```

```

8555 \_@@_inclassignment_args:n { #1 }
8556 \str_if_empty:NTF \l_@@_inclassign_mhrepos_str {
8557   \input{#2}
8558 }{
8559   \stex_in_repository:nn{\l_@@_inclassign_mhrepos_str}{
8560     \input{\mhpath{\l_@@_inclassign_mhrepos_str}{#2}}
8561   }
8562 }
8563 \_@@_inclassignment_args:n {}
8564 }
8565 \newcommand\includeassignment[2][]{
8566   \newpage
8567   \inputassignment[#1]{#2}
8568 }

```

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

## 40.4 Typesetting Exams

\quizheading

```

8569 \ExplSyntaxOff
8570 \newcommand\quizheading[1]{%
8571   \def\@tas{#1}%
8572   \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
8573   \ifx\@tas\@empty\else%
8574     \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
8575   \fi%
8576 }
8577 \ExplSyntaxOn

```

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

\testheading

```

8578
8579 \def\hwexamheader{\input{hwexam-default.header}}
8580
8581 \def\hwexamminutes{
8582   \tl_if_empty:NTF \testheading@duration {
8583     {\testheading@min}~\hwexam@minutes@kw
8584   }{
8585     \testheading@duration
8586   }
8587 }
8588
8589 \keys_define:nn { hwexam / testheading } {
8590   min .tl_set:N = \testheading@min,
8591   duration .tl_set:N = \testheading@duration,
8592   reqpts .tl_set:N = \testheading@reqpts,
8593   tools .tl_set:N = \testheading@tools
8594 }
8595 \cs_new_protected:Nn \_@@_testheading_args:n {
8596   \tl_clear:N \testheading@min
8597   \tl_clear:N \testheading@duration

```

```

8598 \tl_clear:N \testheading@reqpts
8599 \tl_clear:N \testheading@tools
8600 \keys_set:nn { hwexam / testheading }{ #1 }
8601 }
8602 \newenvironment{testheading}[1][]{
8603 \_@@_testheading_args:n{ #1 }
8604 \newcount\check@time\check@time=\testheading@min
8605 \advance\check@time by -\theassignment@totalmin
8606 \newif\if@bonuspoints
8607 \tl_if_empty:NTF \testheading@reqpts {
8608 \@bonuspointsfalse
8609 }{
8610 \newcount\bonus@pts
8611 \bonus@pts=\theassignment@totalpts
8612 \advance\bonus@pts by -\testheading@reqpts
8613 \edef\bonus@pts{\the\bonus@pts}
8614 \@bonuspointstrue
8615 }
8616 \edef\check@time{\the\check@time}
8617
8618 \makeatletter\hwexamheader\makeatother
8619 }{
8620 \newpage
8621 }

```

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

\testspace

```

8622 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

8623 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

8624 \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.

```

8625 <@@=problems>
8626 \renewcommand\@problem[3]{
8627 \stepcounter{assignment@probs}
8628 \def\__problemspts{#2}
8629 \ifx\__problemspts\@empty\else
8630 \addtocounter{assignment@totalpts}{#2}
8631 \fi
8632 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
8633 \xdef\correction@probs{\correction@probs & #1}%
8634 \xdef\correction@pts{\correction@pts & #2}
8635 \xdef\correction@reached{\correction@reached &}

```

```

8636 }
8637 <@@=hwexam>

```

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

```

\correction@table This macro generates the correction table

8638 \newcounter{assignment@probs}
8639 \newcounter{assignment@totalpts}
8640 \newcounter{assignment@totalmin}
8641 \def\correction@probs{\correction@probs@kw}
8642 \def\correction@pts{\correction@pts@kw}
8643 \def\correction@reached{\correction@reached@kw}
8644 \stepcounter{assignment@probs}
8645 \newcommand\correction@table{
8646 \resizebox{\textwidth}{!}{%
8647 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
8648 &\multicolumn{\theassignment@probs}{c|}||%|
8649 {\footnotesize\correction@forgrading@kw} &\\ \hline
8650 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
8651 \correction@pts & \theassignment@totalpts & \\ \hline
8652 \correction@reached & & \[.7cm]\hline
8653 \end{tabular}}
8654 </package>

```

(End definition for \correction@table. This function is documented on page ??.)

## 40.5 Leftovers

at some point, we may want to reactivate the logos font, then we use

here we define the logos that characterize the assignment

```

\font\bierfont=../assignments/bierglas
\font\denkerfont=../assignments/denker
\font\uhrfont=../assignments/uhr
\font\warnschildfont=../assignments/achtung

\newcommand\bierglas{{\bierfont\char65}}
\newcommand\denker{{\denkerfont\char65}}
\newcommand\uhr{{\uhrfont\char65}}
\newcommand\warnschild{{\warnschildfont\char 65}}
\newcommand\hardA{\warnschild}
\newcommand\longA{\uhr}
\newcommand\thinkA{\denker}
\newcommand\discussA{\bierglas}

```

# Chapter 41

## References

EdN:13

13

- [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).