

# The sTeX3 Package Collection \*

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

2022-05-13

## Abstract

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

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

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

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

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

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

---

\*Version 3.0 (last revised 2022-05-13)

# Contents

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

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

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

<b>28</b>	<b>STEX-Module Inheritance Implementation</b>	<b>134</b>
28.1	SMS Mode . . . . .	134
28.2	Inheritance . . . . .	138
<b>29</b>	<b>STEX-Symbols Implementation</b>	<b>144</b>
29.1	Symbol Declarations . . . . .	144
29.2	Notations . . . . .	151
29.3	Variables . . . . .	160
<b>30</b>	<b>STEX-Terms Implementation</b>	<b>167</b>
30.1	Symbol Invocations . . . . .	167
30.2	Terms . . . . .	174
30.3	Notation Components . . . . .	178
30.4	Variables . . . . .	180
30.5	Sequences . . . . .	183
<b>31</b>	<b>STEX-Structural Features Implementation</b>	<b>184</b>
31.1	Imports with modification . . . . .	185
31.2	The feature environment . . . . .	193
31.3	Structure . . . . .	193
<b>32</b>	<b>STEX-Statements Implementation</b>	<b>204</b>
32.1	Definitions . . . . .	204
32.2	Assertions . . . . .	210
32.3	Examples . . . . .	213
32.4	Logical Paragraphs . . . . .	216
<b>33</b>	<b>The Implementation</b>	<b>221</b>
33.1	Proofs . . . . .	221
33.2	Justifications . . . . .	232
<b>34</b>	<b>STEX-Others Implementation</b>	<b>233</b>
<b>35</b>	<b>STEX-Metattheory Implementation</b>	<b>234</b>
<b>36</b>	<b>Tikzinput Implementation</b>	<b>237</b>
<b>37</b>	<b>document-structure.sty Implementation</b>	<b>240</b>
37.1	Package Options . . . . .	240
37.2	Document Structure . . . . .	241
37.3	Front and Backmatter . . . . .	245
37.4	Global Variables . . . . .	247
<b>38</b>	<b>NotesSlides – Implementation</b>	<b>248</b>
38.1	Class and Package Options . . . . .	248
38.2	Notes and Slides . . . . .	250
38.3	Header and Footer Lines . . . . .	254
38.4	Frame Images . . . . .	256
38.5	Colors and Highlighting . . . . .	257
38.6	Sectioning . . . . .	258
38.7	Excursions . . . . .	260

<b>39 The Implementation</b>	<b>262</b>
39.1 Package Options . . . . .	262
39.2 Problems and Solutions . . . . .	263
39.3 Multiple Choice Blocks . . . . .	270
39.4 Including Problems . . . . .	271
39.5 Reporting Metadata . . . . .	273
<b>40 Implementation: The hwexam Package</b>	<b>275</b>
40.1 Package Options . . . . .	275
40.2 Assignments . . . . .	276
40.3 Including Assignments . . . . .	279
40.4 Typesetting Exams . . . . .	280
40.5 Leftovers . . . . .	282
<b>41 References</b>	<b>283</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.



# Chapter 2

## Quickstart

### 2.1 Setup

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

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

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

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

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

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

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

If you want use the latest and greatest  $\text{\texttt{sTeX}}$  packages, you can 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`:

---

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

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

### 2.1.3 sTeX Archives (Manual Setup)

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

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/smgglom/<archive>.git
```

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

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

### 2.1.4 The sTeX IDE

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

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

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

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

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

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

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

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

---

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

## 2.2 A First sTeX Document

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

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

```

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

```

Compiling this document with `pdflatex` should yield the output

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

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

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

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

### Remark 2.2.1:

Note that all of the highlighting, tooltips, coloring and the environment headers come from `stexthm` – by default, the amount of additional packages loaded is kept to a minimum and all the presentations can be customized, see [chapter 6](#).

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

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

**smodule** First, we open a new *module* called `GeometricSeries`. The main purpose of the `smodule` environment is to group the contents and associate it with a *globally unique* identifier (URI), which is computed from the name `GeometricSeries` and the document context. (Depending on your pdf viewer), the URI should pop up in a tooltip if you hover over the word `geometric series`.

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

---

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

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

---

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

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

---

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

---

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

```

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

```

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

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

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

---

`\symname`

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

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

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

---

`\symref`

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

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

---

`\definame`  
`\definiendum`

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

```

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

```

The next snippet – set in a math environment – uses several semantic macros imported from (or recursively via) `series` and `realarithmetics`, such as `\defeq`, `\infinitesum`, etc. In math mode, using a semantic macro inserts its (default) definition. A semantic

macro can have several notations – in that case, we can explicitly choose a specific notation by providing its identifier as an optional argument; e.g. `\realdivide[frac]{a}{b}` will use the explicit notation named `frac` of the semantic macro `\realdivide`, which yields  $\frac{a}{b}$  instead of  $a/b$ .

---

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

---



---

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

---

### 2.2.1 OMDoc/xhtml Conversion

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

#### TODO VSCode Plugin

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

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

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

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

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

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

**Remark 2.2.2:**

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

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

## Chapter 3

# Creating sTeX Content

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

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

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

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

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

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

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

### 3.1 How Knowledge is Organized in sTeX

sTeX content is organized on multiple levels:

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

$\hookrightarrow M \rightarrow$

$\hookrightarrow M \rightarrow$

$\hookrightarrow T \rightarrow$

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

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



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

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

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

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

### 3.2.1 The Local MathHub-Directory

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

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

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

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

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

Each such archive needs two subdirectories:

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

An additional `lib`-directory is optional, and is where `STEX` 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 `STEX` archive:

- `/source/mod/` – individual `STEX` 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.<sup>3</sup>

### 3.2.3 MANIFEST.MF-Files

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

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

Many of these are in fact ignored by `STEX`, but some are important:

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

`source-base` or

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

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

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

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

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

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

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

---

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

---

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

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

---

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

---

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

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

---

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

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

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

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

---

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

**Remark 3.2.1:**

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

```

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

```

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

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

## 3.3 Module, Symbol and Notation Declarations

### 3.3.1 The `smodule`-Environment

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

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

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

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

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

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

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

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

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

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

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

`creators` ( $\langle string \rangle^*$ ) names of the creators.  
`contributors` ( $\langle string \rangle^*$ ) names of contributors.  
`srccite` ( $\langle string \rangle$ ) a source citation for the content of this module.

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

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

### Example 1

Input:

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

Output:

Hello World

## \stexpatchmodule

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

For example:

### Example 2

Input:

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

Output:

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

### 3.3.2 Declaring New Symbols and Notations

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

---

`\symdecl`

---

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

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

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

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

#### Example 3

Input:

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

Output:

Given a `foo`, we can...

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

#### Example 4

Input:

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

Output:

`this` is a symbol taking two arguments.

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

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

`\notation`

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




#### Example 5

Input:

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

Output:

First:  $a$ ; Second:  $b$

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

`\comp`

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

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

#### Example 6

Input:

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

Output:

First:  $a$ ; Second:  $b$



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

Note that it is required that

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

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

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



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

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

## `\symdef`

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

### Example 7

Input:

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

Output:

1.: *a*; 2.: *b*

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

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



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

## $\backslash\text{setnotation}$

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

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

## Operator Notations

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

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

### Example 8

Input:

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

Output:

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

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

## 3.3.3 Argument Modes

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

## Mode-b Arguments

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

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

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

### Example 9

Input:

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

Output:

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

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

## Mode-a Arguments

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

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

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

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

### Example 10

Input:

```

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

```

Output:

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

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

### Example 11

Input:

```

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

```

Output:

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

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

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

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

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

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

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

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

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

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

## Mode-B Arguments

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

### Example 12

Input:

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

Output:

$\forall x,y,z.P$

## 3.3.4 Type and Definiens Components

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

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

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

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

### Example 13

Input:

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

Output:

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

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

#### Example 14

Input:

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

Output:

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

### 3.3.5 Precedences and Automated Bracketing

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

#### Example 15

Input:

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

Output:

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

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

#### Example 16

Input:

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

Output:

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

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

#### Example 17

Input:

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

Output:

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

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

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

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

#### Example 18

Input:

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

Output:

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

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

---

`\infprec`  
`\neginfprec`

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

More precisely, each notation takes

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

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

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

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

In the example above:



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

### 3.3.6 Variables

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

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

---

`\svar`

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

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

---

**\vardef**

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

#### Example 19

Input:

```

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

```

Output:

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

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

TODO: bind=forall/exists

### 3.3.7 Variable Sequences

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

---

**\varseq**

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

This is best shown by example:

#### Example 20

Input:



```

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

```

Output:

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

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

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

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

### Example 21

Input:

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

Output:

$a_1 + \dots + a_n$

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

### Example 22

Input:

```

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

```

Output:

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

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

### Example 23

Input:

```

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

```

Output:

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

### 3.4 Module Inheritance and Structures

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

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

#### 3.4.1 Multilinguality and Translations

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

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

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

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

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

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

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

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

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

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

### 3.4.2 Simple Inheritance and Namespaces

---

`\importmodule`  
`\usemodule`

---

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

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

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



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

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

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

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



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



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

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

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

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

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

---

## `\STEXexport`

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



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

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



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

### 3.4.3 The `mathstructure` Environment

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

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

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

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

How this works is again best demonstrated by example:

#### Example 24

Input:

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

Output:

A `monoid` is...

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

#### Example 25

Input:

```

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

```

Output:

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

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

### Example 26

Input:

```

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

```

Output:

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

### \instantiate

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

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

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

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

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

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

`\varinstantiate`

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

This allows us to do things like:

#### Example 27

Input:

```

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

```

Output:

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

.

and

#### Example 28

Input:

```

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

```

Output:

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

.

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

### 3.4.4 The copymodule Environment

TODO: explain

Given modules:

#### Example 29

Input:

```

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

```

Output:

.

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

#### Example 30

Input:

```

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

```

Output:

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



TODO: explain donotclone

### 3.4.5 The `interpretmodule` Environment

TODO: explain

#### Example 31

Input:

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

Output:

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

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

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

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

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

---

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

## Chapter 4

# Using $\text{\TeX}$ Symbols

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

### 4.1 `\symref` and its variants

---

`\symref`  
`\symname`

---

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

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

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

#### Example 32

Input:

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

Output:

A natural number is...

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

`\Symname`

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

### Example 33

Input:

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

Output:

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



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

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

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

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

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

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

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

### Example 34

Input:

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

Output:

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

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

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



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

## \arg

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

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

### Example 35

Input:

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

Output:

```
Addition is...
```

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

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

### Example 36

Input:

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

Output:

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

adding  $k$  yields...

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

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

### Example 37

Input:

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

Output:

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

## 4.3 Referencing Symbols and Statements

TODO: references documentation

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

## Chapter 5

# sTeX Statements

### 5.1 Definitions, Theorems, Examples, Paragraphs

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

The corresponding environments for that are:

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

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

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

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

#### Example 38

Input:

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

Output:

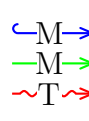

**Example 5.1.1** (An Example).  $2+3$  is 5,  $2\cdot 3$  is 6.

---

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

---

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

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

---

`\definiens`

---

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

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

### Example 39

Input:

---

<sup>7</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...

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

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

```

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

```

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



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

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

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

```

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

```

This yields the following result:

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

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

**1.** For the induction we have to consider the following cases:

**1.1.**  $n = 1$ : then we compute  $1 = 1^2$  □

**1.2.**  $n = 2$ : This case is not really necessary, but we do it for the fun of it (and to get more intuition). We compute  $1 + 3 = 2^2 = 4$  □

**1.3.**  $n > 1$ :

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

**1.3.2.** We have to show that we can derive the assertion for  $n = k + 1$  from this assumption, i.e.  $\sum_{i=1}^{k+1} (2i - 1) = (k + 1)^2$ .

**1.3.3.** We obtain  $\sum_{i=1}^{k+1} (2i - 1) = \sum_{i=1}^k (2i - 1) + 2(k + 1) - 1$  by splitting the sum.

**1.3.4.** Thus we have  $\sum_{i=1}^{k+1} (2i - 1) = k^2 + 2k + 1$  by inductive hypothesis.

**1.3.5.** We can simplify the right-hand side to  $(k + 1)^2$ , which proves the assertion. □

**1.4.** We have considered all the cases, so we have proven the assertion. □

**spproof** The **spproof** environment is the main container for proofs. It takes an optional **KeyVal** argument that allows to specify the **id** (identifier) and **for** (for which assertion is this a proof) keys. The regular argument of the **proof** environment contains an introductory comment, that may be used to announce the proof style. The **proof** environment contains a sequence of **spfststep**, **spfstcomment**, and **spfstcases** environments that are used to markup the proof steps.

---

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

---

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

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

<hr/> <hr/> <code>\spfjust</code>	This evidence is marked up with the <code>\spfjust</code> macro in the <code>stex-proofs</code> package. This environment totally invisible to the formatted result; it wraps the text in the proof step that corresponds to the evidence. The environment takes an optional <code>KeyVal</code> argument, which can have the <code>method</code> key, whose value is the name of a proof method (this will only need to mean something to the application that consumes the semantic annotations). Furthermore, the justification can contain “premises” (specifications to assertions that were used justify the step) and “arguments” (other information taken into account by the proof method).
<hr/> <hr/> <code>\premise</code>	The <code>\premise</code> macro allows to mark up part of the text as reference to an assertion that is used in the argumentation. In the running example we have used the <code>\premise</code> macro to identify the inductive hypothesis.
<hr/> <hr/> <code>\justarg</code>	<p>The <code>\justarg</code> macro is very similar to <code>\premise</code> with the difference that it is used to mark up arguments to the proof method. Therefore the content of the first argument is interpreted as a mathematical object rather than as an identifier as in the case of <code>\premise</code>. In our example, we specified that the simplification should take place on the right hand side of the equation. Other examples include proof methods that instantiate. Here we would indicate the substituted object in a <code>\justarg</code> macro.</p> <p>Note that both <code>\premise</code> and <code>\justarg</code> can be used with an empty second argument to mark up premises and arguments that are not explicitly mentioned in the text.</p>
<code>subproof</code>	The <code>spfcases</code> environment is used to mark up a subproof. This environment takes an optional <code>KeyVal</code> argument for semantic annotations and a second argument that allows to specify an introductory comment (just like in the <code>proof</code> environment). The <code>method</code> key can be used to give the name of the proof method executed to make this subproof.
<code>spfcases</code>	The <code>spfcases</code> environment is used to mark up a proof by cases. Technically it is a variant of the <code>subproof</code> where the <code>method</code> is <code>by-cases</code> . Its contents are <code>spfcase</code> environments that mark up the cases one by one.
<code>spfcase</code>	The content of a <code>spfcases</code> environment are a sequence of case proofs marked up in the <code>spfcase</code> environment, which takes an optional <code>KeyVal</code> argument for semantic annotations. The second argument is used to specify the the description of the case under consideration. The content of a <code>spfcase</code> environment is the same as that of a <code>sproof</code> , i.e. <code>spfsteps</code> , <code>spfcomments</code> , and <code>spfcases</code> environments.
<hr/> <hr/> <code>\spfcasesketch</code>	<code>\spfcasesketch</code> is a variant of the <code>spfcase</code> environment that takes the same arguments, but instead of the <code>spfsteps</code> in the body uses a third argument for a proof sketch.
<code>spfcomment</code>	The <code>spfcomment</code> environment is much like a <code>step</code> , only that it does not have an object-level assertion of its own. Rather than asserting some fact that is relevant for the proof, it is used to explain where the proof is going, what we are attempting to to, or what we have achieved so far. As such, it cannot be the target of a <code>\premise</code> .

---

---

**`\sproofend`**

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

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

---

---

**`\sProofEndSymbol`**

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

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

## Chapter 6

# Highlighting and Presentation Customizations

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

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

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

---

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

---

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

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

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

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

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

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

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

```

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

```

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

```

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

```

---

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

---

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

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

---

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

---

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

```

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

```

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

## Chapter 7

# Additional Packages

### 7.1 Tikzinput: Treating TIKZ code as images

---

**image**

---

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

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

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

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

---

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

---

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

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

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

---

`\mhtikzinput`  
`\cmhtikzinput`

---

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

---

`\libusetikzlibrary`

---

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

## 7.2 Modular Document Structuring

The `document-structure` package supplies an infrastructure for writing OMDOC documents in L<sup>A</sup>T<sub>E</sub>X. This includes a simple structure sharing mechanism for S<sub>T</sub>E<sub>X</sub> 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 S<sub>T</sub>E<sub>X</sub> 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 S<sub>T</sub>E<sub>X</sub> sources, or after translation. Currently, trans-document referencing provided by this package can only be used in the S<sub>T</sub>E<sub>X</sub> collection.

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

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

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

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

```

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

---

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



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

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

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

```

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

```

Here we use two levels of `blindfragment`:

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

---

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

---

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

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

---

`\currentsectionlevel`  
`\CurrentSectionLevel`

---

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

---

`\prematurestop`  
`\afterprematurestop`

---

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

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

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

---

`\setSGvar`  
`\useSGvar`

---

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

---

`\ifSGvar`

---

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

## 7.3 Slides and Course Notes

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

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

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

---

slides  
notes  
sectocframes  
frameimages  
fiboxed

---

The notesslides class takes a variety of class options:

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

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



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

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

```

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

```

---

`\ifnotes`

---

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

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



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



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

---

#### `\inputref*`

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

`nexample`, `nsproof`, `nassertion`

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

---

#### `\setslidelogo`

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

---

#### `\setsource`

The default footer line of the `notesslides` package mentions copyright and licensing. In the `beamer` class, `\source` stores the author's name as the copyright holder . By default it is *Michael Kohlhase* in the `notesslides` package since he is the main user and designer of this package. `\setsource{<name>}` can change the writer's name.

---

#### `\setlicensing`

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

Sometimes, we want to integrate slides as images after all – e.g. because we already have a PowerPoint presentation, to which we want to add `STEX` notes.

---

`\frameimage`  
`\mhframeimage`

---

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

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

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

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

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

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

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

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

---

`\textwarning`

---

The `\textwarning` macro generates a warning sign: 

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

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

---

`\excursion`

---

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

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

---

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

---

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

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

---

`\excursiongroup`

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

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



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

## 7.4 Representing Problems and Solutions

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

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

---

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

---

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

The `boxed` option specifies that problems should be formatted in framed boxes so that they are more visible in the text. Finally, the `test` option signifies that we are in a test situation, so this option does not show the solutions (of course), but leaves space for the students to solve them.

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

---

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

## Example 40

Input:

```

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

```

Output:

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

---

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

---

**Note:** Justify your answer

---

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

---

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

---

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

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

---

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

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

---

---

`\ifsolutions`

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

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

---

---

`\mcc`

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

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

If we start the solutions, then we get

#### Example 41

Input:

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

Output:

##### Problem 7.4.2 (Functions)

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

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

---

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



### Example 42

Input:

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

Output:

#### Problem 7.4.3 (Functions)

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

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

### \includeproblem

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

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

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

## 7.5 Homeworks, Quizzes and Exams

The `hwexam` package and class supplies an infrastructure that allows to format nice-looking assignment sheets by simply including problems from problem files marked up

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

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

Will result in

Name:

Matriculation Number:

## 320101 General Computer Science (Fall 2010)

2022-05-13

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

Write the solutions to the sheet.

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

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

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

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

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

good luck

EdN:13

13

---

### `\inputassignment`

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

---

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

## Part II

# Documentation

# Chapter 8

## sTeX-Basics

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

### 8.1 Macros and Environments

---

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

---

---

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

---

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

#### 8.1.1 HTML Annotations

---

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

---

---

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

---

---

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

---

---

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

---

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

---

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

---

Annotates the HTML generated by  $\langle content \rangle$  with

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

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

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

---

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

---

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

---

---

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

---

---

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

---

---

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

---

---

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

---



---

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

---

### 9.1.2 MathHub Archives

---

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

---

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

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

---

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

---

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

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

---

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

---

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

---

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

---



---

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

---



### 9.1.3 Using Content in Archives

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

# Chapter 10

## STEX-References

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

### 10.1 Macros and Environments

---

**\STEXreftitle**

---

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

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

---

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

---

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

---

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

---

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

---

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

---

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

---

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

---

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

#### 10.1.1 Setting Reference Targets

---

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

---

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

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

---

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

---

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

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

### 10.1.2 Using References

---

`\sref`    `\sref[<opt-args>]{<id>}`

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

---

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

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

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

---

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

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

# Chapter 11

## sTeX-Modules

This sub package contains code related to Modules

### 11.1 Macros and Environments

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

<hr/> <hr/> <code>\c_stex_module_&lt;URI&gt;_prop</code>	<p>A property list with the following fields:</p> <p><b>name</b> The <i>name</i> of the module,</p> <p><b>ns</b> the <i>namespace</i> in field <b>ns</b>,</p> <p><b>file</b> the <i>file</i> containing the module, as a sequence of path fragments</p> <p><b>lang</b> the module's <i>language</i>,</p> <p><b>sig</b> the language of the signature module, if the current file is a translation from some other language,</p> <p><b>deprecate</b> if this module is deprecated, the module that replaces it,</p> <p><b>meta</b> the metatheory of the module.</p>
--	---

<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
---	--

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

---

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

---

Computes the current namespace as follows:

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

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

### 11.1.1 The smodule environment

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

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

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

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

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

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

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

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

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

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

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

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

---

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

---

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

---

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

---

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

---

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

---

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

---

**\stex\_invoke\_module:n** `\stex_invoke_module:n`

---

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

---

**`\stex_activate_module:n`**

---

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

## Chapter 12

# STEX-Module Inheritance

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

### 12.1 Macros and Environments

#### 12.1.1 SMS Mode

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

---

`\g_stex_smsmode_allowedmacros_tl`

---

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

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

---

`\g_stex_smsmode_allowedmacros_escape_tl`

---

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

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

---

`\g_stex_smsmode_allowedenvs_seq`

---

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

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

---

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

---

Tests whether SMS mode is currently active.



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

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

### 12.1.2 Imports and Inheritance

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

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

---

**`\stex_import_module_uri:nn`**

---

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

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

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

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

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

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

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

2. Otherwise:

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

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

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

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

---

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

---

stores the result in these four variables.

---

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

---

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

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

# Chapter 13

## STEX-Symbols

Code related to symbol declarations and notations

### 13.1 Macros and Environments

---

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

---

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

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

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

# Chapter 14

## STEX-Terms

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

### 14.1 Macros and Environments

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

<hr/> <code>\infprec</code> <code>\neginfprec</code> <hr/>	Maximal and minimal notation precedences.
<hr/> <code>\dobrackets</code> <hr/>	<code>\dobrackets {⟨body⟩}</code>  Puts <i>⟨body⟩</i> in parentheses; scaled if in display mode unscaled otherwise. Uses the current $\text{\S}$ TEX brackets (by default ( and )), which can be changed temporarily using <code>\withbrackets</code> .
<hr/> <code>\withbrackets</code> <hr/>	<code>\withbrackets ⟨left⟩ ⟨right⟩ {⟨body⟩}</code>  Temporarily (i.e. within <i>⟨body⟩</i> ) sets the brackets used by $\text{\S}$ TEX for automated bracketing (by default ( and )) to <i>⟨left⟩</i> and <i>⟨right⟩</i> . Note that <i>⟨left⟩</i> and <i>⟨right⟩</i> need to be allowed after <code>\left</code> and <code>\right</code> in display-mode.
<hr/> <code>\stex_term_custom:nn</code> <hr/>	<code>\stex_term_custom:nn{⟨URI⟩}{⟨args⟩}</code>  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 <i>⟨args⟩</i> 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` TODO

## Chapter 16

# TeX-Statements

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

### 16.1 Macros and Environments

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



## Chapter 17

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

## Chapter 18

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

### 18.1 Symbols

**Part III**  
**Extensions**

## Chapter 19

# Tikzinput: Treating TIKZ code as images

### 19.1 Macros and Environments

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

## Chapter 20

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

## Chapter 21

# NotesSlides – Slides and Course Notes

## Chapter 22

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

## Chapter 23

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



**Part IV**  
**Implementation**

## Chapter 24

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

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

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

```
1 \<cls>
2
3 %%%%%%%%% basics.dtx %%%%%%%%%
4
5 \RequirePackage{expl3,l3keys2e}
6 \ProvidesExplClass{stex}{2022/03/03}{3.1.0}{sTeX document class}
7
8 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
9 \ProcessOptions
10
11 \bool_set_true:N \c_stex_document_class_bool
12
13 \RequirePackage{stex}
14
15 \stex_html_backend:TF {
16   \LoadClass{article}
17 }{
18   \LoadClass[border=1px,varwidth,crop=false]{standalone}
19   \setlength\textwidth{15cm}
20 }
21 \RequirePackage{standalone}
22
23
24 \clist_if_empty:NT \c_stex_languages_clist {
25   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
26   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
27   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
28   \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
29     \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
30       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
```

```

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

```

## 24.2 Preliminaries

```

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

```

Package options:

```

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

```

**\stex** The sTeX logo:

**\sTeX**

```

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

```

```

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

```

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

## 24.3 Messages and logging

```

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

```

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

```

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

```

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

Redirecting messages:

```

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

```

## 24.4 HTML Annotations

116 `<@=stex_annotate>`

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

117 `\tl_new:N \l_stex_html_arg_tl`

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

`\_stex_html_checkempty:n`

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

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

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

`\stex_if_do_html:TF`

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

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

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

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

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

`\stex_annotate:bnx`

`\stex_annotate_invisible:n`

`\stex_annotate_invisible:nnn`

We define four macros for introducing attributes in the HTML output. The definitions depend on the “backend” used (L<sup>A</sup>T<sub>E</sub>X<sub>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.

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

```

146     \def\stex@backend{latexml}
147     \else
148     \def\stex@backend{pdflatex}
149     \fi
150     \fi
151 }
152 \input{stex-backend-\stex@backend.cfg}
153
154 \newif\ifstexhtml
155 \stex_html_backend:TF\stexhtmltrue\stexhtmlfalse
156

```

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

## 24.5 Babel Languages

```

157 <@=stex_language>

```

`\c_stex_languages_prop`  
`\c_stex_language_abbrevs_prop`

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

```

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

```

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

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

```

183 \cs_new_protected:Nn \stex_set_language:Nn {
184     \str_set:Nx \l_tmpa_str {#2}
185     \prop_get:NoNT \c_stex_languages_prop \l_tmpa_str #1 {
186         \ifx\@onlypreamble\@notprerr
187         \ltx@ifpackageloaded{babel}{

```

```

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

```

## 24.6 Persistence

```

236 <@@=stex_persist>
237 \bool_if:NTF \c_stex_persist_mode_bool {

```

```

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

```

## 24.7 Auxiliary Methods

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

```

258 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
259 \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
260 \def#1{
261 \msg_error:nnnn{stex}{error/deactivated-macro}{\detokenize{#1}}{#2}
262 }
263 }

```

(End definition for \stex\_deactivate\_macro:Nn. This function is documented on page 64.)

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

```

264 \cs_new_protected:Nn \stex_reactivate_macro:N {
265 \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname
266 }

```

(End definition for \stex\_reactivate\_macro:N. This function is documented on page 64.)

**\ignorespacesandpars**

```

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

```



```

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

```

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

`\MMTrule`

```

334 \NewDocumentCommand \MMTrule {m m}{
335   \seq_set_split:Nnn \l_tmpa_seq , {#2}
336   \int_zero:N \l_tmpa_int
337   \stex_annotate_invisible:nnn{mmtrule}{scala://#1}{
338     \seq_if_empty:NF \l_tmpa_seq {
339       $\seq_map_inline:Nn \l_tmpa_seq {
340         \int_incr:N \l_tmpa_int
341         \stex_annotate:nnn{arg}{i\int_use:N \l_tmpa_int}{##1}
342       }$
343     }
344   }
345 }
346
347 \NewDocumentCommand \MMTinclude {m}{
348   \stex_annotate_invisible:nnn{import}{#1}{ }
349 }
350
351 \tl_new:N \g_stex_document_title
352 \cs_new_protected:Npn \STEXtitle #1 {
353   \tl_if_empty:NT \g_stex_document_title {
354     \tl_gset:Nn \g_stex_document_title { #1 }
355   }
356 }
357 \cs_new_protected:Nn \stex_document_title:n {
358   \tl_if_empty:NT \g_stex_document_title {
359     \tl_gset:Nn \g_stex_document_title { #1 }
360     \stex_annotate_invisible:n{\noindent
361       \stex_annotate:nnn{doctitle}{ }{ #1 }
362     }
363   }
364 }
365 \AtBeginDocument {
366   \let \STEXtitle \stex_document_title:n
367   \tl_if_empty:NF \g_stex_document_title {
368     \stex_annotate_invisible:n{\noindent
369       \stex_annotate:nnn{doctitle}{ }{ \g_stex_document_title }
370     }
371   }
372   \let \stex_maketitle \maketitle
373   \def \maketitle {
374     \tl_if_empty:NF \@title {
375       \exp_args:No \stex_document_title:n \@title
376     }
377     \stex_maketitle:
378   }
379 }
380
381 \cs_new_protected:Nn \stex_par: {
382   \mode_if_vertical:F{
383     \if@minipage\else\if@nobreak\else\par\fi\fi
384   }

```

385 }

386

387  $\langle$ /package $\rangle$

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

## Chapter 25

# STEX -MathHub Implementation

```
388 <*package>
389
390 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
391
392 <@@=stex_path>
393
394 Warnings and error messages
395 \msg_new:nnn{stex}{error/norepository}{
396   No~archive~#1~found~in~#2
397 }
398 \msg_new:nnn{stex}{error/notinarchive}{
399   Not~currently~in~an~archive,~but~\detokenize{#1}~
400   needs~one!
401 }
402 \msg_new:nnn{stex}{error/nofile}{
403   \detokenize{#1}~could~not~find~file~#2
404 }
405 \msg_new:nnn{stex}{error/twofiles}{
406   \detokenize{#1}~found~two~candidates~for~#2
407 }
```

### 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`

```
406 \cs_new_protected:Nn \stex_path_from_string:Nn {
407   \str_set:Nx \l_tmpa_str { #2 }
408   \str_if_empty:NTF \l_tmpa_str {
409     \seq_clear:N #1
410   }{
411     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
412     \sys_if_platform_windows:T{
413       \seq_clear:N \l_tmpa_tl
```

```

414     \seq_map_inline:Nn #1 {
415       \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
416       \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
417     }
418     \seq_set_eq:NN #1 \l_tmpa_tl
419   }
420   \stex_path_canonicalize:N #1
421 }
422 }
423

```

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

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

```

424 \cs_new_protected:Nn \stex_path_to_string:NN {
425   \exp_args:Nne \str_set:Nn #2 { \seq_use:Nn #1 / }
426 }
427
428 \cs_new:Nn \stex_path_to_string:N {
429   \seq_use:Nn #1 /
430 }

```

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

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

```

431 \str_const:Nn \c__stex_path_dot_str {.}
432 \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.

```

433 \cs_new_protected:Nn \stex_path_canonicalize:N {
434   \seq_if_empty:NF #1 {
435     \seq_clear:N \l_tmpa_seq
436     \seq_get_left:NN #1 \l_tmpa_tl
437     \str_if_empty:NT \l_tmpa_tl {
438       \seq_put_right:Nn \l_tmpa_seq {}
439     }
440     \seq_map_inline:Nn #1 {
441       \str_set:Nn \l_tmpa_tl { ##1 }
442       \str_if_eq:NNF \l_tmpa_tl \c__stex_path_dot_str {
443         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
444           \seq_if_empty:NNTF \l_tmpa_seq {
445             \exp_args:Nno \seq_put_right:Nn \l_tmpa_seq {
446               \c__stex_path_up_str
447             }
448           }{
449             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
450             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
451               \exp_args:Nno \seq_put_right:Nn \l_tmpa_seq {
452                 \c__stex_path_up_str
453               }
454             }{

```

```

455         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
456     }
457 }
458 }{
459     \str_if_empty:NF \l_tmpa_tl {
460         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
461     }
462 }
463 }
464 }
465 \seq_gset_eq:NN #1 \l_tmpa_seq
466 }
467 }

```

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

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

```

468 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
469     \seq_if_empty:NTF #1 {
470         \prg_return_false:
471     }{
472         \seq_get_left:NN #1 \l_tmpa_tl
473         \sys_if_platform_windows:TF{
474             \str_if_in:NnTF \l_tmpa_tl {:}{
475                 \prg_return_true:
476             }{
477                 \prg_return_false:
478             }
479         }{
480             \str_if_empty:NTF \l_tmpa_tl {
481                 \prg_return_true:
482             }{
483                 \prg_return_false:
484             }
485         }
486     }
487 }

```

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

## 25.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

488 \str_new:N\l_stex_kpsewhich_return_str
489 \cs_new_protected:Nn \stex_kpsewhich:n {\begingroup
490     \catcode'\ =12
491     \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
492     \tl_gset_eq:NN \l_tmpa_tl \l_tmpa_tl
493     \endgroup
494     \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
495     \tl_trim_spaces:N \l_stex_kpsewhich_return_str
496 }

```

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

We determine the PWD

`\c_stex_pwd_seq`  
`\c_stex_pwd_str`

```

497 \sys_if_platform_windows:TF{
498   \begingroup\escapechar=-1\catcode'\=12
499   \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
500   \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
501   \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_
502   }}{
503   \stex_kpsewhich:n{-var-value~PWD}
504   }
505
506 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
507 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
508 \stex_debug:nn {mathhub} {PWD:~\str_use:N\c_stex_pwd_str}

```

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

## 25.3 File Hooks and Tracking

509 `<@@=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

```

510 \seq_gclear_new:N\g_stex_files_stack

```

(End definition for `\g_stex_files_stack`.)

`\c_stex_mainfile_seq`  
`\c_stex_mainfile_str`

```

511 \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
512 \stex_path_from_string:Nn \c_stex_mainfile_seq
513   \c_stex_mainfile_str

```

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

`\g_stex_currentfile_seq`

```

514 \seq_gclear_new:N\g_stex_currentfile_seq

```

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

`\stex_filestack_push:n`

```

515 \cs_new_protected:Nn \stex_filestack_push:n {
516   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
517   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
518     \stex_path_from_string:Nn\g_stex_currentfile_seq{
519       \c_stex_pwd_str/#1
520     }

```

```

521 }
522 \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
523 \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
524 }

```

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

**`\stex_filestack_pop:`**

```

525 \cs_new_protected:Nn \stex_filestack_pop: {
526   \seq_if_empty:NF\g__stex_files_stack{
527     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
528   }
529   \seq_if_empty:NTF\g__stex_files_stack{
530     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
531   }{
532     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
533     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
534   }
535 }

```

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

Hooks for the current file:

```

536 \AddToHook{file/before}{
537   \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
538 }
539 \AddToHook{file/after}{
540   \stex_filestack_pop:
541 }

```

## 25.4 MathHub Repositories

```

542 <@@=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.

```

543 \str_if_empty:NTF\mathhub{
544   \sys_if_platform_windows:TF{
545     \begingroup\escapechar=-1\catcode'\=12
546     \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str MATHHUB\c_percent_str}
547     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
548     \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_kpsewhich_return_str}}
549   }{
550     \stex_kpsewhich:n{-var-value-MATHHUB}
551   }
552   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
553 }
554 \str_if_empty:NT \c_stex_mathhub_str {
555   \sys_if_platform_windows:TF{
556     \begingroup\escapechar=-1\catcode'\=12
557     \exp_args:Nx\stex_kpsewhich:n{-var-value-HOME}
558     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
559     \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_kpsewhich_return_str}}
560   }{

```



```

561     \stex_kpsewhich:n{-var-value~HOME}
562   }
563   \ior_open:NnT \l_tmpa_ior{\l_stex_kpsewhich_return_str / .stex / mathhub.path}{
564     \begingroup\escapechar=-1\catcode'\=12
565     \ior_str_get:NN \l_tmpa_ior \l_tmpa_str
566     \sys_if_platform_windows:T{
567       \exp_args:NNx\str_replace_all:Nnn\l_tmpa_str{c_backslash_str}/
568     }
569     \str_gset_eq:NN \c_stex_mathhub_str\l_tmpa_str
570     \endgroup
571     \ior_close:N \l_tmpa_ior
572   }
573 }
574 \str_if_empty:NTF\c_stex_mathhub_str{
575   \msg_warning:nn{stex}{warning/nomathhub}
576 }{
577   \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
578   \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
579 }
580 }{
581   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
582   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
583     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
584       \c_stex_pwd_str/\mathhub
585     }
586   }
587   \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
588   \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
589 }

```

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

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

```

590 \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
591   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
592     \str_set:Nx \l_tmpa_str { #1 }
593     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
594     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
595     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
596     \__stex_mathhub_find_manifest:N \l_tmpa_seq
597     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
598       \msg_error:nnxx{stex}{error/norepository}{#1}{
599         \stex_path_to_string:N \c_stex_mathhub_str
600       }
601     } {
602       \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
603     }
604   }
605 }

```

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

`\l_stex_mathhub_manifest_file_seq`

```
606 \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`:

```
607 \cs_new_protected:Nn \__stex_mathhub_find_manifest:N {
608   \seq_set_eq:NN\l_tmpa_seq #1
609   \bool_set_true:N\l_tmpa_bool
610   \bool_while_do:Nn \l_tmpa_bool {
611     \seq_if_empty:NTF \l_tmpa_seq {
612       \bool_set_false:N\l_tmpa_bool
613     }{
614       \file_if_exist:nTF{
615         \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
616       }{
617         \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
618         \bool_set_false:N\l_tmpa_bool
619       }{
620         \file_if_exist:nTF{
621           \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
622         }{
623           \seq_put_right:Nn\l_tmpa_seq{META-INF}
624           \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
625           \bool_set_false:N\l_tmpa_bool
626         }{
627           \file_if_exist:nTF{
628             \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
629           }{
630             \seq_put_right:Nn\l_tmpa_seq{meta-inf}
631             \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
632             \bool_set_false:N\l_tmpa_bool
633           }{
634             \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
635           }
636         }
637       }
638     }
639   }
640   \seq_set_eq:NN\l__stex_mathhub_manifest_file_seq\l_tmpa_seq
641 }
```

(End definition for `\__stex_mathhub_find_manifest:N`.)

`\c__stex_mathhub_manifest_ior` File variable used for MANIFEST-files

```
642 \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:

```
643 \cs_new_protected:Nn \__stex_mathhub_parse_manifest:n {
644   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
645   \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
646   \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
```

```

647 \str_set:Nn \l_tmpa_str {##1}
648 \exp_args:NNoo \seq_set_split:Nnn
649   \l_tmpb_seq \c_colon_str \l_tmpa_str
650 \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
651   \exp_args:NNe \str_set:Nn \l_tmpb_tl {
652     \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
653   }
654   \exp_args:No \str_case:nnTF \l_tmpa_tl {
655     {id} {
656       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
657       { id } \l_tmpb_tl
658     }
659     {narration-base} {
660       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
661       { narr } \l_tmpb_tl
662     }
663     {url-base} {
664       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
665       { docurl } \l_tmpb_tl
666     }
667     {source-base} {
668       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
669       { ns } \l_tmpb_tl
670     }
671     {ns} {
672       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
673       { ns } \l_tmpb_tl
674     }
675     {dependencies} {
676       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
677       { deps } \l_tmpb_tl
678     }
679   }{}{}
680 }{}
681 }
682 \ior_close:N \c__stex_mathhub_manifest_ior
683 \stex_persist:x {
684   \prop_set_from_keyval:cn{ c_stex_mathhub_#1_manifest_prop }{
685     \exp_after:wN \prop_to_keyval:N \csname c_stex_mathhub_#1_manifest_prop\endcsname
686   }
687 }
688 }

```

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

`\stex_set_current_repository:n`

```

689 \cs_new_protected:Nn \stex_set_current_repository:n {
690   \stex_require_repository:n { #1 }
691   \prop_set_eq:Nc \l_stex_current_repository_prop {
692     c_stex_mathhub_#1_manifest_prop
693   }
694 }

```

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

`\stex_require_repository:n`

```

695 \cs_new_protected:Nn \stex_require_repository:n {
696   \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
697     \stex_debug:nn{mathhub}{Opening~archive:~#1}
698     \__stex_mathhub_do_manifest:n { #1 }
699   }
700 }

```

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

`\l_stex_current_repository_prop`

Current MathHub repository

```

701 %\prop_new:N \l_stex_current_repository_prop
702 \bool_if:NF \c_stex_persist_mode_bool {
703   \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
704   \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
705     \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
706   } {
707     \__stex_mathhub_parse_manifest:n { main }
708     \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
709     \l_tmpa_str
710     \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str_manifest_prop }
711     \c_stex_mathhub_main_manifest_prop
712     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
713     \stex_debug:nn{mathhub}{Current~repository:~
714     \prop_item:Nn \l_stex_current_repository_prop {id}
715   }
716 }
717 }

```

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

`\stex_in_repository:nn`

Executes the code in the second argument in the context of the repository whose ID is provided as the first argument.

```

718 \cs_new_protected:Nn \stex_in_repository:nn {
719   \str_set:Nx \l_tmpa_str { #1 }
720   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
721   \str_if_empty:NTF \l_tmpa_str {
722     \prop_if_exist:NTF \l_stex_current_repository_prop {
723       \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
724       \exp_args:Ne \l_tmpa_cs{
725         \prop_item:Nn \l_stex_current_repository_prop { id }
726       }
727     }{
728       \l_tmpa_cs{}
729     }
730   }{
731     \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}
732     \stex_require_repository:n \l_tmpa_str
733     \str_set:Nx \l_tmpa_str { #1 }
734     \exp_args:Nne \use:nn {
735       \stex_set_current_repository:n \l_tmpa_str
736       \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
737     }{
738       \stex_debug:nn{mathhub}{switching~back~to:~

```

```

739     \prop_if_exist:NTF \l_stex_current_repository_prop {
740       \prop_item:Nn \l_stex_current_repository_prop { id } :~
741       \meaning\l_stex_current_repository_prop
742     }{
743       no~repository
744     }
745   }
746   \prop_if_exist:NTF \l_stex_current_repository_prop {
747     \stex_set_current_repository:n {
748       \prop_item:Nn \l_stex_current_repository_prop { id }
749     }
750   }{
751     \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
752   }
753 }
754 }
755 }

```

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

## 25.5 Using Content in Archives

`\mhpath`

```

756 \def \mhpath #1 #2 {
757   \exp_args:Ne \tl_if_empty:nTF{#1}{
758     \c_stex_mathhub_str /
759     \prop_item:Nn \l_stex_current_repository_prop { id }
760     / source / #2
761   }{
762     \c_stex_mathhub_str / #1 / source / #2
763   }
764 }

```

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

`\inputref`

`\mhinput`

```

765 \newif \ifinputref \inputreffalse
766
767 \cs_new_protected:Nn \__stex_mathhub_mhinput:nn {
768   \stex_in_repository:nn {#1} {
769     \ifinputref
770       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
771     \else
772       \inputreftrue
773       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
774       \inputreffalse
775     \fi
776   }
777 }
778 \NewDocumentCommand \mhinput { 0{} m }{
779   \__stex_mathhub_mhinput:nn{ #1 }{ #2 }
780 }
781

```

```

782 \cs_new_protected:Nn \__stex_mathhub_inputref:nn {
783   \stex_in_repository:nn {#1} {
784     \stex_html_backend:TF {
785       \str_clear:N \l_tmpa_str
786       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
787         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
788       }
789
790       \tl_if_empty:nTF{ ##1 }{
791         \IfFileExists{#2}{
792           \stex_annotate_invisible:nnn{inputref}{
793             \l_tmpa_str / #2
794           }{}
795         }{
796           \input{#2}
797         }
798       }{
799         \IfFileExists{ \c_stex_mathhub_str / ##1 / source / #2 }{
800           \stex_annotate_invisible:nnn{inputref}{
801             \l_tmpa_str / #2
802           }{}
803         }{
804           \input{ \c_stex_mathhub_str / ##1 / source / #2 }
805         }
806       }
807
808     }{
809       \begingroup
810       \inputreftrue
811       \tl_if_empty:nTF{ ##1 }{
812         \input{#2}
813       }{
814         \input{ \c_stex_mathhub_str / ##1 / source / #2 }
815       }
816       \endgroup
817     }
818   }
819 }
820 \NewDocumentCommand \inputref { 0{} m}{
821   \__stex_mathhub_inputref:nn{ #1 }{ #2 }
822 }

```

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

#### `\addmhbibresource`

```

823 \cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {
824   \stex_in_repository:nn {#1} {
825     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
826   }
827 }
828 \newcommand\addmhbibresource[2][]{
829   \__stex_mathhub_mhbibresource:nn{ #1 }{ #2 }
830 }

```

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

## `\libinput`

```
831 \cs_new_protected:Npn \libinput #1 {
832   \prop_if_exist:NF \l_stex_current_repository_prop {
833     \msg_error:nnn{stex}{error/notinarchive}\libinput
834   }
835   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
836     \msg_error:nnn{stex}{error/notinarchive}\libinput
837   }
838   \seq_clear:N \l__stex_mathhub_libinput_files_seq
839   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
840   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
841
842   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
843     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}
844     \IfFileExists{ \l_tmpa_str }{
845       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
846     }{}
847     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
848     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
849   }
850
851   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
852   \IfFileExists{ \l_tmpa_str }{
853     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
854   }{}
855
856   \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
857     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
858   }{
859     \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
860       \input{ ##1 }
861     }
862   }
863 }
```

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

## `\libusepackage`

```
864 \NewDocumentCommand \libusepackage {0{ } m} {
865   \prop_if_exist:NF \l_stex_current_repository_prop {
866     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
867   }
868   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
869     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
870   }
871   \seq_clear:N \l__stex_mathhub_libinput_files_seq
872   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
873   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
874
875   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
876     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2}
877     \IfFileExists{ \l_tmpa_str.sty }{
878       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
879     }{}
880   }
```

```

880 \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
881 \seq_put_right:No \l_tmpa_seq \l_tmpa_str
882 }
883
884 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2}
885 \IfFileExists{ \l_tmpa_str.sty }{
886 \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
887 }{}
888
889 \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
890 \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
891 }{
892 \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
893 \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
894 \usepackage[#1]{ ##1 }
895 }
896 }{
897 \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
898 }
899 }
900 }

```

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

`\mhgraphics`  
`\cmhgraphics`

```

901
902 \AddToHook{begindocument}{
903 \ltx@ifpackageloaded{graphicx}{
904 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
905 \newcommand\mhgraphics[2][]{\%
906 \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
907 \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}}
908 \newcommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
909 }{}

```

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

`\lstinputmhlisting`  
`\clstinputmhlisting`

```

910 \ltx@ifpackageloaded{listings}{
911 \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
912 \newcommand\lstinputmhlisting[2][]{\%
913 \def\lst@mhrepos{}\setkeys{lst}{#1}%
914 \lstinputlisting[#1]{\mhp\lst@mhrepos{#2}}}
915 \newcommand\clstinputmhlisting[2][]{\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
916 }{}
917 }
918
919 </package>

```

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



## Chapter 26

# STEX -References Implementation

```
920 <*package>
921
922 %%%%%%%%%% references.dtx %%%%%%%%%%
923
924 <@@=stex_refs>
    Warnings and error messages
925
```

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

```
926 %\iow_new:N \c__stex_refs_refs_iow
927 \AtBeginDocument{
928 % \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
929 }
930 \AtEndDocument{
931 % \iow_close:N \c__stex_refs_refs_iow
932 }
```

`\STEXreftitle`

```
933 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
934
935 \NewDocumentCommand \STEXreftitle { m } {
936   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
937 }
```

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

### 26.1 Document URIs and URLs

`\l_stex_current_docns_str`

```
938 \str_new:N \l_stex_current_docns_str
```

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

`\stex_get_document_uri:`

```
939 \cs_new_protected:Nn \stex_get_document_uri: {
940   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
941   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
942   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
943   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
944   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
945
946   \str_clear:N \l_tmpa_str
947   \prop_if_exist:NT \l_stex_current_repository_prop {
948     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
949       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
950     }
951   }
952
953   \str_if_empty:NTF \l_tmpa_str {
954     \str_set:Nx \l_stex_current_docns_str {
955       file:/\stex_path_to_string:N \l_tmpa_seq
956     }
957   }{
958     \bool_set_true:N \l_tmpa_bool
959     \bool_while_do:Nn \l_tmpa_bool {
960       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
961       \exp_args:No \str_case:nnTF { \l_tmpb_str } {
962         {source} { \bool_set_false:N \l_tmpa_bool }
963       }{}{
964         \seq_if_empty:NT \l_tmpa_seq {
965           \bool_set_false:N \l_tmpa_bool
966         }
967       }
968     }
969
970     \seq_if_empty:NTF \l_tmpa_seq {
971       \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
972     }{
973       \str_set:Nx \l_stex_current_docns_str {
974         \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
975       }
976     }
977   }
978 }
```

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

`\l_stex_current_docurl_str`

```
979 \str_new:N \l_stex_current_docurl_str
```

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

`\stex_get_document_url:`

```
980 \cs_new_protected:Nn \stex_get_document_url: {
981   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
982   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
983   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
```

```

984 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
985 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
986
987 \str_clear:N \l_tmpa_str
988 \prop_if_exist:NT \l_stex_current_repository_prop {
989   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
990     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
991       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
992     }
993   }
994 }
995
996 \str_if_empty:NTF \l_tmpa_str {
997   \str_set:Nx \l_stex_current_docurl_str {
998     file:/\stex_path_to_string:N \l_tmpa_seq
999   }
1000 }{
1001   \bool_set_true:N \l_tmpa_bool
1002   \bool_while_do:Nn \l_tmpa_bool {
1003     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1004     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1005       {source} { \bool_set_false:N \l_tmpa_bool }
1006     }{}{
1007       \seq_if_empty:NT \l_tmpa_seq {
1008         \bool_set_false:N \l_tmpa_bool
1009       }
1010     }
1011   }
1012
1013   \seq_if_empty:NTF \l_tmpa_seq {
1014     \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
1015   }{
1016     \str_set:Nx \l_stex_current_docurl_str {
1017       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
1018     }
1019   }
1020 }
1021 }

```

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

## 26.2 Setting Reference Targets

```

1022 \str_const:Nn \c__stex_refs_url_str{URL}
1023 \str_const:Nn \c__stex_refs_ref_str{REF}
1024 \str_new:N \l__stex_refs_curr_label_str
1025 % @currentlabel -> number
1026 % @currentlabelname -> title
1027 % @currentHref -> name.number <- id of some kind
1028 % \theH# -> \arabic{section}
1029 % \the# -> number
1030 % \hyper@makecurrent{#}
1031 \int_new:N \l__stex_refs_unnamed_counter_int

```

`\stex_ref_new_doc_target:n`

```

1032 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
1033   \stex_get_document_uri:
1034   \str_clear:N \l__stex_refs_curr_label_str
1035   \str_set:Nx \l_tmpa_str { #1 }
1036   \str_if_empty:NT \l_tmpa_str {
1037     \int_incr:N \l__stex_refs_unnamed_counter_int
1038     \str_set:Nx \l_tmpa_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
1039   }
1040   \str_set:Nx \l__stex_refs_curr_label_str {
1041     \l_stex_current_docns_str?\l_tmpa_str
1042   }
1043   \seq_if_exist:cF{g__stex_refs_labels_\l_tmpa_str_seq}{
1044     \seq_new:c {g__stex_refs_labels_\l_tmpa_str_seq}
1045   }
1046   \seq_if_in:coF{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str {
1047     \seq_gput_right:co{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str
1048   }
1049   \stex_if_smsmode:TF {
1050     \stex_get_document_url:
1051     \str_gset_eq:cN {sref_url_\l__stex_refs_curr_label_str_str}\l_stex_current_docurl_str
1052     \str_gset_eq:cN {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_url_str
1053   }{
1054     %\iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter\unexpanded\expandafter{
1055     \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
1056     \immediate\write\@auxout{\stexauxadddocref{\l_stex_current_docns_str}{\l_tmpa_str}}
1057     \str_gset:cx {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_ref_str
1058   }
1059 }

```

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

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

```

1060 \cs_new_protected:Npn \stexauxadddocref #1 #2 {
1061   \str_set:Nn \l_tmpa_str {#1?#2}
1062   \str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
1063   \seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
1064     \seq_new:c {g__stex_refs_labels_#2_seq}
1065   }
1066   \seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
1067     \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
1068   }
1069 }

```

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

```

1070 \AtEndDocument{
1071   \def\stexauxadddocref#1 #2 {}{}
1072 }

```

`\stex_ref_new_sym_target:n`

```

1073 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
1074   \stex_if_smsmode:TF {
1075     \str_if_exist:cF{sref_sym_#1_type}{
1076       \stex_get_document_url:
1077       \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str

```

```

1078     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
1079   }
1080 }{
1081   \str_if_empty:NF \l__stex_refs_curr_label_str {
1082     \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
1083     \immediate\write\@auxout{
1084       \exp_not:N\expandafter\def\exp_not:N\csname \exp_not:N\detokenize{sref_sym_#1_label_
1085         \l__stex_refs_curr_label_str
1086       }
1087     }
1088   }
1089 }
1090 }

```

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

## 26.3 Using References

```

1091 \str_new:N \l__stex_refs_indocument_str

```

**\sref** Optional arguments:

```

1092
1093 \keys_define:nn { stex / sref } {
1094   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
1095   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
1096   pre           .tl_set:N = \l__stex_refs_pre_tl ,
1097   post          .tl_set:N = \l__stex_refs_post_tl ,
1098 }
1099 \cs_new_protected:Nn \__stex_refs_args:n {
1100   \tl_clear:N \l__stex_refs_linktext_tl
1101   \tl_clear:N \l__stex_refs_fallback_tl
1102   \tl_clear:N \l__stex_refs_pre_tl
1103   \tl_clear:N \l__stex_refs_post_tl
1104   \str_clear:N \l__stex_refs_repo_str
1105   \keys_set:nn { stex / sref } { #1 }
1106 }

```

The actual macro:

```

1107 \NewDocumentCommand \sref { 0{} m}{
1108   \__stex_refs_args:n { #1 }
1109   \str_if_empty:NTF \l__stex_refs_indocument_str {
1110     \str_set:Nx \l_tmpa_str { #2 }
1111     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
1112     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
1113       \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
1114         \seq_get_left:cNF {g__stex_refs_labels_\l_tmpa_str _seq} \l_tmpa_str {
1115           \str_clear:N \l_tmpa_str
1116         }
1117       }{
1118         \str_clear:N \l_tmpa_str
1119       }
1120     }{
1121       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1122       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str

```

```

1123 \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
1124 \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str_seq}{
1125   \str_set_eq:NN \l_tmpc_str \l_tmpa_str
1126   \str_clear:N \l_tmpa_str
1127   \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str_seq} {
1128     \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
1129       \str_range:nnn { ##1 }{ -\l_tmpa_int}{ -1 }
1130     }{
1131       \seq_map_break:n {
1132         \str_set:Nn \l_tmpa_str { ##1 }
1133       }
1134     }
1135   }
1136 }{
1137   \str_clear:N \l_tmpa_str
1138 }
1139 }
1140 \str_if_empty:NTF \l_tmpa_str {
1141   \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_lin
1142 }{
1143   \str_if_eq:cNTF {sref_\l_tmpa_str_type} \c__stex_refs_ref_str {
1144     \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1145       \cs_if_exist:cTF{autoref}{
1146         \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1147       }{
1148         \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1149       }
1150     }{
1151       \ltx@ifpackageloaded{hyperref}{
1152         \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
1153       }{
1154         \l__stex_refs_linktext_tl
1155       }
1156     }
1157   }{
1158     \ltx@ifpackageloaded{hyperref}{
1159       \href{\use:c{sref_url_\l_tmpa_str_str}}{\tl_if_empty:NTF \l__stex_refs_linktext_t
1160     }{
1161       \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs
1162     }
1163   }
1164 }
1165 }{
1166   % TODO
1167 }
1168 }

```

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

## `\srefsym`

```

1169 \NewDocumentCommand \srefsym { 0{} m}{
1170   \stex_get_symbol:n { #2 }
1171   \__stex_refs_sym_aux:nn{##1}{\l_stex_get_symbol_uri_str}
1172 }

```

```

1173
1174 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1175   \str_if_exist:cTF {sref_sym_#2 _label_str }{
1176     \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1177   }{
1178     \__stex_refs_args:n { #1 }
1179     \str_if_empty:NTF \l__stex_refs_indocument_str {
1180       \tl_if_exist:cTF{sref_sym_#2 _type}{
1181         % doc uri in \l_tmpb_str
1182         \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1183         \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1184           % reference
1185           \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1186             \cs_if_exist:cTF{autoref}{
1187               \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1188             }{
1189               \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1190             }
1191           }{
1192             \ltx@ifpackageloaded{hyperref}{
1193               \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1194             }{
1195               \l__stex_refs_linktext_tl
1196             }
1197           }
1198         }{
1199           % URL
1200           \ltx@ifpackageloaded{hyperref}{
1201             \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl \
1202           }{
1203             \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_re
1204           }
1205         }
1206       }{
1207         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_l
1208       }
1209     }{
1210       % TODO
1211     }
1212   }
1213 }

```

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

**\srefsymuri**

```

1214 \cs_new_protected:Npn \srefsymuri #1 #2 {
1215   \__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1216 }

```

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

```

1217 </package>

```

## Chapter 27

# STEX -Modules Implementation

```
1218 <*package>
1219
1220 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1221
1222 <@@=stex_modules>
1223
1224   Warnings and error messages
1225   \msg_new:nnn{stex}{error/unknownmodule}{
1226     No~module~#1~found
1227   }
1228   \msg_new:nnn{stex}{error/syntax}{
1229     Syntax~error:~#1
1230   }
1231   \msg_new:nnn{stex}{error/siglanguage}{
1232     Module~#1~declares~signature~#2,~but~does~not~
1233     declare~its~language
1234   }
1235   \msg_new:nnn{stex}{warning/deprecated}{
1236     #1~is~deprecated;~please~use~#2~instead!
1237   }
1238   \msg_new:nnn{stex}{error/conflictingmodules}{
1239     Conflicting~imports~for~module~#1
1240   }
```

```
\l_stex_current_module_str The current module:
1240 \str_new:N \l_stex_current_module_str
(End definition for \l_stex_current_module_str. This variable is documented on page 71.)
```

```
\l_stex_all_modules_seq Stores all available modules
1241 \seq_new:N \l_stex_all_modules_seq
(End definition for \l_stex_all_modules_seq. This variable is documented on page 71.)
```



```

\stex_if_in_module_p:
\stex_if_in_module:TF
1242 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1243   \str_if_empty:NTF \l_stex_current_module_str
1244   \prg_return_false: \prg_return_true:
1245 }

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

```

```

\stex_if_module_exists_p:n
\stex_if_module_exists:nTF
1246 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1247   \prop_if_exist:cTF { c_stex_module_#1_prop }
1248   \prg_return_true: \prg_return_false:
1249 }

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

```

```

\stex_add_to_current_module:n
\STEXexport
1250 \cs_new_protected:Nn \stex_execute_in_module:n { \stex_if_in_module:T {
1251   \stex_add_to_current_module:n { #1 }
1252   \stex_do_up_to_module:n { #1 }
1253 }}
1254 \cs_generate_variant:Nn \stex_execute_in_module:n {x}
1255
1256 \cs_new_protected:Nn \stex_add_to_current_module:n {
1257   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1258 }
1259 \cs_generate_variant:Nn \stex_add_to_current_module:n {x}
1260 \cs_new_protected:Npn \STEXexport {
1261   \begingroup
1262   \newlinechar=-1\relax
1263   \endlinechar=-1\relax
1264   %\catcode'\ = 9\relax
1265   \expandafter\endgroup\__stex_modules_export:n
1266 }
1267 \cs_new_protected:Nn \__stex_modules_export:n {
1268   \ignorespaces #1
1269   \stex_add_to_current_module:n { \ignorespaces #1 }
1270   \stex_smsmode_do:
1271 }
1272 \stex_deactivate_macro:Nn \STEXexport {module~environments}

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

```

```

\stex_add_constant_to_current_module:n
1273 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1274   \str_set:Nx \l_tmpa_str { #1 }
1275   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1276 }

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

```

`\stex_add_import_to_current_module:n`

```

1277 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1278   \str_set:Nx \l_tmpa_str { #1 }
1279   \exp_args:Nno
1280   \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1281     \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1282   }
1283 }

```

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

`\stex_collect_imports:n`

```

1284 \cs_new_protected:Nn \stex_collect_imports:n {
1285   \seq_clear:N \l_stex_collect_imports_seq
1286   \__stex_modules_collect_imports:n {#1}
1287 }
1288 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1289   \seq_map_inline:cn {c_stex_module_#1_imports} {
1290     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1291       \__stex_modules_collect_imports:n { ##1 }
1292     }
1293   }
1294   \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1295     \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1296   }
1297 }

```

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

`\stex_do_up_to_module:n`

```

1298 \int_new:N \l__stex_modules_group_depth_int
1299 \cs_new_protected:Nn \stex_do_up_to_module:n {
1300   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1301     #1
1302   }{
1303     #1
1304     \expandafter \tl_gset:Nn
1305     \csname l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1306     \expandafter\expandafter\expandafter\endcsname
1307     \expandafter\expandafter\expandafter { \csname
1308       l__stex_modules_aftergroup_\l_stex_current_module_str _tl\endcsname #1 }
1309     \aftergroup\__stex_modules_aftergroup_do:
1310   }
1311 }
1312 \cs_generate_variant:Nn \stex_do_up_to_module:n {x}
1313 \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1314   \stex_debug:nn{aftergroup}{\cs_meaning:c{
1315     l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1316   }}
1317   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1318     \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1319     \tl_gclear:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1320   }{
1321     \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}

```

```

1322 \aftergroup\__stex_modules_aftergroup_do:
1323 }
1324 }
1325 \cs_new_protected:Nn \stex_reset_up_to_module:n {
1326 \expandafter\let\csname l__stex_modules_aftergroup_#1_tl\endcsname\undefined
1327 }

```

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

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

```

1328

```

(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.

```

1329 \str_new:N \l_stex_module_ns_str
1330 \str_new:N \l_stex_module_subpath_str
1331 \cs_new_protected:Nn \__stex_modules_compute_namespace:nN {
1332 \seq_set_eq:NN \l_tmpa_seq #2
1333 % split off file extension
1334 \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str % <- filename
1335 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1336 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str % <- filename without suffixes
1337 \seq_put_right:No \l_tmpa_seq \l_tmpb_str % <- file path including name without suffixes
1338
1339 \bool_set_true:N \l_tmpa_bool
1340 \bool_while_do:Nn \l_tmpa_bool {
1341 \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1342 \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1343 {source} { \bool_set_false:N \l_tmpa_bool }
1344 }{}{
1345 \seq_if_empty:NT \l_tmpa_seq {
1346 \bool_set_false:N \l_tmpa_bool
1347 }
1348 }
1349 }
1350
1351 \stex_path_to_string:NN \l_tmpa_seq \l_stex_module_subpath_str
1352 % \l_tmpa_seq <- sub-path relative to archive
1353 \str_if_empty:NTF \l_stex_module_subpath_str {
1354 \str_set:Nx \l_stex_module_ns_str {#1}
1355 }{
1356 \str_set:Nx \l_stex_module_ns_str {
1357 #1/\l_stex_module_subpath_str
1358 }
1359 }
1360 }
1361
1362 \cs_new_protected:Nn \stex_modules_current_namespace: {
1363 \str_clear:N \l_stex_module_subpath_str
1364 \prop_if_exist:NTF \l_stex_current_repository_prop {
1365 \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str

```

```

1366     \__stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1367   }{
1368     % split off file extension
1369     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1370     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1371     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1372     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1373     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1374     \str_set:Nx \l_stex_module_ns_str {
1375       file:/\stex_path_to_string:N \l_tmpa_seq
1376     }
1377   }
1378 }

```

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

## 27.1 The smodule environment

smodule arguments:

```

1379 \keys_define:nn { stex / module } {
1380   title      .tl_set:N      = \smodulename ,
1381   type       .str_set_x:N   = \smodulename ,
1382   id         .str_set_x:N   = \smoduleid ,
1383   deprecate  .str_set_x:N   = \l_stex_module_deprecate_str ,
1384   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1385   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1386   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1387   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1388   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1389   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1390   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1391 }
1392
1393 \cs_new_protected:Nn \__stex_modules_args:n {
1394   \str_clear:N \smodulename
1395   \str_clear:N \smodulename
1396   \str_clear:N \smoduleid
1397   \str_clear:N \l_stex_module_ns_str
1398   \str_clear:N \l_stex_module_deprecate_str
1399   \str_clear:N \l_stex_module_lang_str
1400   \str_clear:N \l_stex_module_sig_str
1401   \str_clear:N \l_stex_module_creators_str
1402   \str_clear:N \l_stex_module_contributors_str
1403   \str_clear:N \l_stex_module_meta_str
1404   \str_clear:N \l_stex_module_srccite_str
1405   \keys_set:nn { stex / module } { #1 }
1406 }
1407
1408 % module parameters here? In the body?
1409

```

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

```

1410 \cs_new_protected:Nn \stex_module_setup:nn {

```

```

1411 \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1412 \str_set:Nx \l_stex_module_name_str { #2 }
1413 \__stex_modules_args:n { #1 }

```

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

```

1414 \stex_if_in_module:TF {
1415   % Nested module
1416   \prop_get:cnN {c_stex_module\l_stex_current_module_str _prop}
1417   { ns } \l_stex_module_ns_str
1418   \str_set:Nx \l_stex_module_name_str {
1419     \prop_item:cn {c_stex_module\l_stex_current_module_str _prop}
1420     { name } / \l_stex_module_name_str
1421   }
1422   \str_if_empty:NT \l_stex_module_lang_str {
1423     \str_set:Nx \l_stex_module_lang_str {
1424       \prop_item:cn {c_stex_module\l_stex_current_module_str _prop}
1425       { lang }
1426     }
1427   }
1428 }{
1429   % not nested:
1430   \str_if_empty:NT \l_stex_module_ns_str {
1431     \stex_modules_current_namespace:
1432     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1433       / {\l_stex_module_ns_str}
1434     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1435     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1436       \str_set:Nx \l_stex_module_ns_str {
1437         \stex_path_to_string:N \l_tmpa_seq
1438       }
1439     }
1440   }
1441 }

```

Next, we determine the language of the module:

```

1442 \str_if_empty:NT \l_stex_module_lang_str {
1443   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1444   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1445   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1446   \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
1447     \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
1448       \exp_args:NNNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
1449     }
1450   }
1451   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1452   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
1453     \seq_pop_right:NN \l_tmpa_seq \l_stex_module_lang_str
1454     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1455       inferred~from~file~name}
1456   }
1457 }
1458
1459 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {

```

```

1460 \exp_args:NNo \stex_set_language:Nn \l_tmpa_str \l_stex_module_lang_str
1461 }}

```

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

```

1462 \str_if_empty:NTF \l_stex_module_sig_str {
1463   \exp_args:Nnx \prop_gset_from_keyval:cn {
1464     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1465   } {
1466     name      = \l_stex_module_name_str ,
1467     ns        = \l_stex_module_ns_str ,
1468     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1469     lang      = \l_stex_module_lang_str ,
1470     sig       = \l_stex_module_sig_str ,
1471     deprecate = \l_stex_module_deprecate_str ,
1472     meta      = \l_stex_module_meta_str
1473   }
1474   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1475   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1476   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _copymodules}
1477   \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1478   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1479 \str_if_empty:NT \l_stex_module_meta_str {
1480   \str_set:Nx \l_stex_module_meta_str {
1481     \c_stex_metatheory_ns_str ? Metatheory
1482   }
1483 }
1484 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1485   \bool_set_true:N \l_stex_in_meta_bool
1486   \exp_args:Nx \stex_add_to_current_module:n {
1487     \bool_set_true:N \l_stex_in_meta_bool
1488     \stex_activate_module:n {\l_stex_module_meta_str}
1489     \bool_set_false:N \l_stex_in_meta_bool
1490   }
1491   \stex_activate_module:n {\l_stex_module_meta_str}
1492   \bool_set_false:N \l_stex_in_meta_bool
1493 }
1494 }{
1495   \str_if_empty:NT \l_stex_module_lang_str {
1496     \msg_error:nnxx{stex}{error/siglanguage}{
1497       \l_stex_module_ns_str?\l_stex_module_name_str
1498     }\l_stex_module_sig_str}
1499   }
1500   \stex_debug:nn{modules}{Signature~\l_stex_module_sig_str~for~\l_stex_module_ns_str?\l_st
1501   \stex_if_module_exists:nTF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1502     \stex_debug:nn{modules}{(already exists)}
1503   }{
1504     \stex_debug:nn{modules}{(needs loading)}
1505     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1506     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1507     \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1508     \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex

```

```

1509 \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1510 \str_set:Nx \l_tmpa_str {
1511   \stex_path_to_string:N \l_tmpa_seq /
1512   \l_tmpa_str . \l_stex_module_sig_str .tex
1513 }
1514 \IfFileExists \l_tmpa_str {
1515   \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1516     \str_clear:N \l_stex_current_module_str
1517     \seq_clear:N \l_stex_all_modules_seq
1518     \stex_debug:nn{modules}{Loading~signature}
1519   }
1520   {{
1521     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1522   }
1523 }
1524 \stex_if_smsmode:F {
1525   \stex_activate_module:n {
1526     \l_stex_module_ns_str ? \l_stex_module_name_str
1527   }
1528 }
1529 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1530 }
1531 \str_if_empty:NF \l_stex_module_deprecate_str {
1532   \msg_warning:nnxx{stex}{warning/deprecated}{
1533     Module~\l_stex_current_module_str
1534   }{
1535     \l_stex_module_deprecate_str
1536   }
1537 }
1538 \seq_put_right:Nx \l_stex_all_modules_seq {
1539   \l_stex_module_ns_str ? \l_stex_module_name_str
1540 }
1541 \tl_clear:c{l_stex_modules_aftergroup\l_stex_module_ns_str ? \l_stex_module_name_str _tl}
1542 }

```

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

**smodule** The module environment.

```

\__stex_modules_begin_module: implements \begin{smodule}
1543 \cs_new_protected:Nn \__stex_modules_begin_module: {
1544   \stex_reactivate_macro:N \STEXexport
1545   \stex_reactivate_macro:N \importmodule
1546   \stex_reactivate_macro:N \symdecl
1547   \stex_reactivate_macro:N \notation
1548   \stex_reactivate_macro:N \symdef
1549
1550   \stex_debug:nn{modules}{
1551     New~module:\\
1552     Namespace:~\l_stex_module_ns_str\\
1553     Name:~\l_stex_module_name_str\\
1554     Language:~\l_stex_module_lang_str\\
1555     Signature:~\l_stex_module_sig_str\\
1556     Metatheory:~\l_stex_module_meta_str\\

```

```

1557   File:~\stex_path_to_string:N \g_stex_currentfile_seq
1558 }
1559
1560 \stex_if_do_html:T{
1561   \begin{stex_annotate_env} {theory} {
1562     \l_stex_module_ns_str ? \l_stex_module_name_str
1563   }
1564
1565   \stex_annotate_invisible:nnn{header}{} {
1566     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1567     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1568     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1569       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1570     }
1571     \str_if_empty:NF \smoduletype {
1572       \stex_annotate:nnn{type}{\smoduletype}{}
1573     }
1574   }
1575 }
1576 % TODO: Inherit metatheory for nested modules?
1577 }
1578 \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}

```

1579 \cs_new_protected:Nn \_stex_modules_end_module: {
1580   \stex_debug:nn{modules}{Closing module~\prop_item:cn {c_stex_module_\l_stex_current_module_str}
1581   \stex_reset_up_to_module:n \l_stex_current_module_str
1582   \stex_if_smsmode:T {
1583     \stex_persist:x {
1584       \prop_set_from_keyval:cn{c_stex_module_\l_stex_current_module_str _prop}{
1585         \exp_after:wN \prop_to_keyval:N \csname c_stex_module_\l_stex_current_module_str _pr
1586       }
1587       \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _constants}{
1588         \seq_use:cn{c_stex_module_\l_stex_current_module_str _constants},
1589       }
1590       \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _imports}{
1591         \seq_use:cn{c_stex_module_\l_stex_current_module_str _imports},
1592       }
1593       \tl_set:cn {c_stex_module_\l_stex_current_module_str _code}
1594     }
1595     \exp_after:wN \let \exp_after:wN \l_tmpa_tl \csname c_stex_module_\l_stex_current_module_str
1596     \exp_after:wN \stex_persist:n \exp_after:wN { \exp_after:wN { \l_tmpa_tl } }
1597   }
1598 }

```

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

The core environment

```

1599 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1600 \NewDocumentEnvironment { smodule } { 0 } { m } {
1601   \stex_module_setup:nn{#1}{#2}
1602   %\par
1603   \stex_if_smsmode:F{

```



```

1604 \tl_if_empty:NF \smodulename {
1605   \exp_args:No \stex_document_title:n \smodulename
1606 }
1607 \tl_clear:N \l_tmpa_tl
1608 \clist_map_inline:Nn \smodulename {
1609   \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1610     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1611   }
1612 }
1613 \tl_if_empty:NTF \l_tmpa_tl {
1614   \__stex_modules_smodule_start:
1615 }{
1616   \l_tmpa_tl
1617 }
1618 }
1619 \__stex_modules_begin_module:
1620 \str_if_empty:NF \smoduleid {
1621   \stex_ref_new_doc_target:n \smoduleid
1622 }
1623 \stex_smsmode_do:
1624 } {
1625   \__stex_modules_end_module:
1626   \stex_if_smsmode:F {
1627     \end{stex_annotate_env}
1628     \clist_set:No \l_tmpa_clist \smodulename
1629     \tl_clear:N \l_tmpa_tl
1630     \clist_map_inline:Nn \l_tmpa_clist {
1631       \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1632         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1633       }
1634     }
1635     \tl_if_empty:NTF \l_tmpa_tl {
1636       \__stex_modules_smodule_end:
1637     }{
1638       \l_tmpa_tl
1639     }
1640   }
1641 }

```

### **\stexpatchmodule**

```

1642 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1643 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1644
1645 \newcommand\stexpatchmodule[3] [] {
1646   \str_set:Nx \l_tmpa_str{ #1 }
1647   \str_if_empty:NTF \l_tmpa_str {
1648     \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1649     \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1650   }{
1651     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1652     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1653   }
1654 }

```

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

## 27.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n 1655 \NewDocumentCommand \STEXModule { m } {
1656   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1657   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1658   \tl_set:Nn \l_tmpa_tl {
1659     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1660   }
1661   \seq_map_inline:Nn \l_stex_all_modules_seq {
1662     \str_set:Nn \l_tmpb_str { ##1 }
1663     \str_if_eq:eeT { \l_tmpa_str } {
1664       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1665     } {
1666       \seq_map_break:n {
1667         \tl_set:Nn \l_tmpa_tl {
1668           \stex_invoke_module:n { ##1 }
1669         }
1670       }
1671     }
1672   }
1673   \l_tmpa_tl
1674 }
1675
1676 \cs_new_protected:Nn \stex_invoke_module:n {
1677   \stex_debug:nn{modules}{Invoking~module~#1}
1678   \peek_charcode_remove:NTF ! {
1679     \__stex_modules_invoke_uri:nN { #1 }
1680   } {
1681     \peek_charcode_remove:NTF ? {
1682       \__stex_modules_invoke_symbol:nn { #1 }
1683     } {
1684       \msg_error:nnx{stex}{error/syntax}{
1685         ?~or~!~expected~after~
1686         \c_backslash_str STEXModule{#1}
1687       }
1688     }
1689   }
1690 }
1691
1692 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1693   \str_set:Nn #2 { #1 }
1694 }
1695
1696 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1697   \stex_invoke_symbol:n{#1?#2}
1698 }

```

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

```

\stex_activate_module:n
1699 \bool_new:N \l_stex_in_meta_bool
1700 \bool_set_false:N \l_stex_in_meta_bool

```

```

1701 \cs_new_protected:Nn \stex_activate_module:n {
1702   \stex_debug:nn{modules}{Activating~module~#1}
1703   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1704     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1705     \use:c{ c_stex_module_#1_code }
1706   }
1707 }

```

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

```

1708 \endpackage

```

## Chapter 28

# STEX -Module Inheritance Implementation

```
1709 <*package>
1710
1711 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1712
```

### 28.1 SMS Mode

```
1713 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1714 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1715 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1716 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1717
1718 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1719   \makeatletter
1720   \makeatother
1721   \ExplSyntaxOn
1722   \ExplSyntaxOff
1723   \rustexBREAK
1724 }
1725
1726 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1727   \symdef
1728   \importmodule
1729   \notation
1730   \symdecl
1731   \STEXexport
1732   \inlineass
1733   \inlinedef
1734   \inlineex
1735   \endinput
1736   \setnotation
```

```

1737 \copynotation
1738 \assign
1739 \renamedekl
1740 \donotcopy
1741 \instantiate
1742 }
1743
1744 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1745   \tl_to_str:n {
1746     smodule,
1747     copymodule,
1748     interpretmodule,
1749     sdefinition,
1750     sexample,
1751     sassertion,
1752     sparagraph,
1753     mathstructure
1754   }
1755 }

```

(End definition for `\g_stex_smsmode_allowedmacros_tl`, `\g_stex_smsmode_allowedmacros_escape_tl`, and `\g_stex_smsmode_allowedenvs_seq`. These variables are documented on page 74.)

```

\stex_if_smsmode_p:
\stex_if_smsmode:TF
1756 \bool_new:N \g__stex_smsmode_bool
1757 \bool_set_false:N \g__stex_smsmode_bool
1758 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1759   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1760 }

```

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

```

\__stex_smsmode_in_smsmode:nn
1761 \cs_new_protected:Nn \__stex_smsmode_in_smsmode:nn { \stex_suppress_html:n {
1762   \vbox_set:Nn \l_tmpa_box {
1763     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1764     \bool_gset_true:N \g__stex_smsmode_bool
1765     #2
1766     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1767   }
1768   \box_clear:N \l_tmpa_box
1769 } }

```

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

```

\stex_file_in_smsmode:nn
1770 \quark_new:N \q__stex_smsmode_break
1771
1772 \NewDocumentCommand \__stex_smsmode_importmodule: { 0{} m } {
1773   \seq_gput_right:Nn \l__stex_smsmode_importmodules_seq {{#1}{#2}}
1774   \stex_smsmode_do:
1775 }
1776
1777 \cs_new_protected:Nn \__stex_smsmode_module:nn {
1778   \__stex_modules_args:n{#1}

```

```

1779 \stex_if_in_module:F {
1780   \str_if_empty:NF \l_stex_module_sig_str {
1781     \stex_modules_current_namespace:
1782     \str_set:Nx \l_stex_module_name_str { #2 }
1783     \stex_if_module_exists:nF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1784       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1785       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1786       \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1787       \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1788       \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1789       \str_set:Nx \l_tmpa_str {
1790         \stex_path_to_string:N \l_tmpa_seq /
1791         \l_tmpa_str . \l_stex_module_sig_str .tex
1792       }
1793       \IfFileExists \l_tmpa_str {
1794         \exp_args:NNx \seq_gput_right:Nn \l__stex_smsmode_sigmodules_seq \l_tmpa_str
1795       }{
1796         \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1797       }
1798     }
1799   }
1800 }
1801 }
1802
1803 \prg_new_conditional:Nnn \__stex_smsmode_check_import_pair:nn {T,F,TF} {
1804   %\stex_debug:nn{import-pair}{\detokenize{#{1}~#{2}}}}
1805   \tl_if_empty:nTF{#{1}}{
1806     \prop_if_exist:NTF \l_stex_current_repository_prop
1807     {
1808       %\stex_debug:nn{import-pair}{in repository \prop_item:Nn \l_stex_current_repository_
1809       \prg_return_true:
1810     } {
1811       \seq_set_split:Nnn \l_tmpa_seq ? {#2}
1812       \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
1813       \tl_if_empty:NT \l_tmpa_tl {
1814         \seq_pop_left:NN \l_tmpa_seq \l_tmpa_tl
1815       }
1816       %\stex_debug:nn{import-pair}{\seq_use:Nn \l_tmpa_seq,~of~length~\seq_count:N \l_tmpa
1817       \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1
1818         \prg_return_true: \prg_return_false:
1819     }
1820   }\prg_return_true:
1821 }
1822
1823 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1824   \stex_filestack_push:n{#1}
1825   \seq_gclear:N \l__stex_smsmode_importmodules_seq
1826   \seq_gclear:N \l__stex_smsmode_sigmodules_seq
1827   % ----- new -----
1828   \__stex_smsmode_in_smsmode:nn{#1}{
1829     \let\importmodule\__stex_smsmode_importmodule:
1830     \let\stex_module_setup:nn\__stex_smsmode_module:nn
1831     \let\__stex_modules_begin_module:\relax
1832     \let\__stex_modules_end_module:\relax

```

```

1833 \seq_clear:N \g_stex_smsmode_allowedenvs_seq
1834 \exp_args:NNx \seq_put_right:Nn \g_stex_smsmode_allowedenvs_seq {\tl_to_str:n{smodule}}
1835 \tl_clear:N \g_stex_smsmode_allowedmacros_tl
1836 \tl_clear:N \g_stex_smsmode_allowedmacros_escape_tl
1837 \tl_put_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\importmodule}
1838 \everyeof{\q__stex_smsmode_break\noexpand}
1839 \expandafter\expandafter\expandafter
1840 \stex_smsmode_do:
1841 \csname @ @ input\endcsname "#1"\relax
1842
1843 \seq_map_inline:Nn \l__stex_smsmode_sigmodules_seq {
1844   \stex_filestack_push:n{##1}
1845   \expandafter\expandafter\expandafter
1846   \stex_smsmode_do:
1847   \csname @ @ input\endcsname "##1"\relax
1848   \stex_filestack_pop:
1849 }
1850 }
1851 % ----- new -----
1852 \__stex_smsmode_in_smsmode:nn{#1} {
1853   #2
1854   % ----- new -----
1855   \begingroup
1856   %\stex_debug:nn{smsmode}{Here:~\seq_use:Nn\l__stex_smsmode_importmodules_seq, }
1857   \seq_map_inline:Nn \l__stex_smsmode_importmodules_seq {
1858     \__stex_smsmode_check_import_pair:nnT ##1 { \begingroup
1859       \stex_import_module_uri:nn ##1
1860       \stex_import_require_module:nnnn
1861       \l_stex_import_ns_str
1862       \l_stex_import_archive_str
1863       \l_stex_import_path_str
1864       \l_stex_import_name_str \endgroup
1865     }
1866   }
1867   \endgroup
1868   \stex_debug:nn{smsmode}{Actually~loading~file~#1}
1869   % ----- new -----
1870   \everyeof{\q__stex_smsmode_break\noexpand}
1871   \expandafter\expandafter\expandafter
1872   \stex_smsmode_do:
1873   \csname @ @ input\endcsname "#1"\relax
1874 }
1875 \stex_filestack_pop:
1876 }

```

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

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

```

1877 \cs_new_protected:Npn \stex_smsmode_do: {
1878   \stex_if_smsmode:T {
1879     \__stex_smsmode_do:w
1880   }
1881 }

```

```

1882 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1883   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
1884     \expandafter\if\expandafter\relax\noexpand#1
1885     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1886     \else\expandafter\__stex_smsmode_do:w\fi
1887   }{
1888     \__stex_smsmode_do:w %#1
1889   }
1890 }
1891 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1892   \cs_if_eq:NNTF #1 \q__stex_smsmode_break {
1893     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1894       #1\__stex_smsmode_do:w
1895     }{
1896       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1897         #1
1898       }{
1899         \cs_if_eq:NNTF \begin #1 {
1900           \__stex_smsmode_check_begin:n
1901         }{
1902           \cs_if_eq:NNTF \end #1 {
1903             \__stex_smsmode_check_end:n
1904           }{
1905             \__stex_smsmode_do:w
1906           }
1907         }
1908       }
1909     }
1910   }
1911 }
1912
1913 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1914   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1915     \begin{#1}
1916   }{
1917     \__stex_smsmode_do:w
1918   }
1919 }
1920 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1921   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1922     \end{#1}\__stex_smsmode_do:w
1923   }{
1924     \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1925   }
1926 }

```

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

## 28.2 Inheritance

```

1927 <@@=stex_importmodule>

```

`\stex_import_module_uri:nn`

```

1928 \cs_new_protected:Nn \stex_import_module_uri:nn {

```



```

1929 \str_set:Nx \l_stex_import_archive_str { #1 }
1930 \str_set:Nn \l_stex_import_path_str { #2 }
1931
1932 \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1933 \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1934 \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1935
1936 \stex_modules_current_namespace:
1937 \bool_lazy_all:nTF {
1938   {\str_if_empty_p:N \l_stex_import_archive_str}
1939   {\str_if_empty_p:N \l_stex_import_path_str}
1940   {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1941 }{
1942   \str_set_eq:NN \l_stex_import_path_str \l_stex_module_subpath_str
1943   \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1944 }{
1945   \str_if_empty:NT \l_stex_import_archive_str {
1946     \prop_if_exist:NT \l_stex_current_repository_prop {
1947       \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1948     }
1949   }
1950   \str_if_empty:NTF \l_stex_import_archive_str {
1951     \str_if_empty:NF \l_stex_import_path_str {
1952       \stex_path_from_string:Nn \l_tmpb_seq {
1953         \l_stex_module_ns_str / .. / \l_stex_import_path_str
1954       }
1955       \str_set:Nx \l_stex_import_ns_str {\stex_path_to_string:N \l_tmpb_seq}
1956       \str_replace_once:Nnn \l_stex_import_ns_str {file:/} {file://}
1957     }
1958   }{
1959     \stex_require_repository:n \l_stex_import_archive_str
1960     \prop_get:cnN { c_stex_mathhub\l_stex_import_archive_str _manifest_prop } { ns }
1961     \l_stex_import_ns_str
1962     \str_if_empty:NF \l_stex_import_path_str {
1963       \str_set:Nx \l_stex_import_ns_str {
1964         \l_stex_import_ns_str / \l_stex_import_path_str
1965       }
1966     }
1967   }
1968 }
1969 }

```

```

\l_stex_import_name_str Store the return values of \stex_import_module_uri:nn.
\l_stex_import_archive_str 1970 \str_new:N \l_stex_import_name_str
\l_stex_import_path_str 1971 \str_new:N \l_stex_import_archive_str
\l_stex_import_ns_str 1972 \str_new:N \l_stex_import_path_str
1973 \str_new:N \l_stex_import_ns_str

```

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

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

---

```
1974 \cs_new_protected:Nn \stex_import_require_module:nnnn {
```

```

1975 \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1976
1977   \stex_debug:nn{requiremodule}{Here:\~1:~#1\~2:~#2\~3:~#3\~4:~#4}
1978
1979   \exp_args:NNxx \seq_set_split:Nnn \l_tmpa_seq {\tl_to_str:n{/}} {#4}
1980   \seq_get_left:NN \l_tmpa_seq \l_tmpc_str
1981
1982   %\stex_debug:nn{requiremodule}{Top-module:\l_tmpc_str}
1983
1984   % archive
1985   \str_set:Nx \l_tmpa_str { #2 }
1986   \str_if_empty:NTF \l_tmpa_str {
1987     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1988     \seq_put_right:Nn \l_tmpa_seq {...}
1989   } {
1990     \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1991     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1992     \seq_put_right:Nn \l_tmpa_seq { source }
1993   }
1994
1995   % path
1996   \str_set:Nx \l_tmpb_str { #3 }
1997   \str_if_empty:NTF \l_tmpb_str {
1998     \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / \l_tmpc_str }
1999
2000     \ltx@ifpackageloaded{babel} {
2001       \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
2002         { \language } \l_tmpb_str {
2003         \msg_error:nnx{stex}{error/unknownlanguage}{\language}
2004       }
2005     } {
2006       \str_clear:N \l_tmpb_str
2007     }
2008
2009     \stex_debug:nn{modules}{Checking~a1~\l_tmpa_str.\l_tmpb_str.tex}
2010     \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2011       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2012     }{
2013       \stex_debug:nn{modules}{Checking~a2~\l_tmpa_str.tex}
2014       \IfFileExists{ \l_tmpa_str.tex }{
2015         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2016       }{
2017         % try english as default
2018         \stex_debug:nn{modules}{Checking~a3~\l_tmpa_str.en.tex}
2019         \IfFileExists{ \l_tmpa_str.en.tex }{
2020           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2021         }{
2022           \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2023         }
2024       }
2025     }
2026
2027   } {
2028     \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str

```

```

2029 \seq_concat:NNN \l_tmpb_seq \l_tmpa_seq \l_tmpb_seq
2030
2031 \ltx@ifpackageloaded{babel} {
2032   \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
2033     { \language } \l_tmpb_str {
2034       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
2035     }
2036   } {
2037     \str_clear:N \l_tmpb_str
2038   }
2039
2040 \stex_path_canonicalize:N \l_tmpb_seq
2041 \stex_path_to_string:NN \l_tmpb_seq \l_tmpa_str
2042
2043 \stex_debug:nn{modules}{Checking~b1~\l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex}
2044 \IfFileExists{ \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }{
2045   \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }
2046 }{
2047   \stex_debug:nn{modules}{Checking~b2~\l_tmpa_str/\l_tmpc_str.tex}
2048   \IfFileExists{ \l_tmpa_str/\l_tmpc_str.tex }{
2049     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.tex }
2050   }{
2051     % try english as default
2052     \stex_debug:nn{modules}{Checking~b3~\l_tmpa_str/\l_tmpc_str.en.tex}
2053     \IfFileExists{ \l_tmpa_str/\l_tmpc_str.en.tex }{
2054       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.en.tex }
2055     }{
2056       \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.\l_tmpb_str.tex}
2057       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2058         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2059       }{
2060         \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.tex}
2061         \IfFileExists{ \l_tmpa_str.tex }{
2062           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2063         }{
2064           % try english as default
2065           \stex_debug:nn{modules}{Checking~b5~\l_tmpa_str.en.tex}
2066           \IfFileExists{ \l_tmpa_str.en.tex }{
2067             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2068           }{
2069             \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2070           }
2071         }
2072       }
2073     }
2074   }
2075 }
2076
2077
2078 \str_if_eq:eeF{\g__stex_importmodule_file_str}{\seq_use:Nn \g_stex_currentfile_seq /}{
2079   \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
2080     \seq_clear:N \l_stex_all_modules_seq
2081     \str_clear:N \l_stex_current_module_str
2082     \str_set:Nx \l_tmpb_str { #2 }

```

```

2083     \str_if_empty:NF \l_tmpb_str {
2084       \stex_set_current_repository:n { #2 }
2085     }
2086     \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
2087   }
2088
2089   \stex_if_module_exists:nF { #1 ? #4 } {
2090     \msg_error:nnx{stex}{error/unknownmodule}{
2091       #1?#4~(in~file~\g__stex_importmodule_file_str)
2092     }
2093   }
2094 }
2095
2096 }
2097 \stex_activate_module:n { #1 ? #4 }
2098 }

```

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

## `\importmodule`

```

2099 \NewDocumentCommand \importmodule { 0{} m } {
2100   \stex_import_module_uri:nn { #1 } { #2 }
2101   \stex_debug:nn{modules}{Importing~module:~
2102     \l_stex_import_ns_str ? \l_stex_import_name_str
2103   }
2104   \stex_import_require_module:nnnn
2105   { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2106   { \l_stex_import_path_str } { \l_stex_import_name_str }
2107   \stex_if_smsmode:F {
2108     \stex_annotate_invisible:nnn
2109     {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
2110   }
2111   \exp_args:Nx \stex_add_to_current_module:n {
2112     \stex_import_require_module:nnnn
2113     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2114     { \l_stex_import_path_str } { \l_stex_import_name_str }
2115   }
2116   \exp_args:Nx \stex_add_import_to_current_module:n {
2117     \l_stex_import_ns_str ? \l_stex_import_name_str
2118   }
2119   \stex_smsmode_do:
2120   \ignorespacesandpars
2121 }
2122 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

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

## `\usemodule`

```

2123 \NewDocumentCommand \usemodule { 0{} m } {
2124   \stex_if_smsmode:F {
2125     \stex_import_module_uri:nn { #1 } { #2 }
2126     \stex_import_require_module:nnnn
2127     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2128     { \l_stex_import_path_str } { \l_stex_import_name_str }
2129     \stex_annotate_invisible:nnn

```

```

2130     {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
2131   }
2132   \stex_smsmode_do:
2133   \ignorespacesandpars
2134 }

```

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

```

2135 \cs_new_protected:Nn \stex_csl_to_imports:Nn {
2136   \tl_if_empty:nF{#2}{
2137     \clist_set:Nn \l_tmpa_clist {#2}
2138     \clist_map_inline:Nn \l_tmpa_clist {
2139       \tl_if_head_eq_charcode:nNTF {##1} [{
2140         #1 ##1
2141       } {
2142         #1{##1}
2143       }
2144     }
2145   }
2146 }
2147 \cs_generate_variant:Nn \stex_csl_to_imports:Nn {No}
2148
2149
2150 \</package>

```

## Chapter 29

# STEX -Symbols Implementation

```
2151 <*package>
2152
2153 %%%%%%%%%% symbols.dtx %%%%%%%%%%
2154
2155 Warnings and error messages
2156 \msg_new:nnn{stex}{error/wrongargs}{
2157   args~value~in~symbol~declaration~for~#1~
2158   needs~to~be~i,~a,~b~or~B,~but~#2~given
2159 }
2160 \msg_new:nnn{stex}{error/unknownsymbol}{
2161   No~symbol~#1~found!
2162 }
2163 \msg_new:nnn{stex}{error/seqlength}{
2164   Expected~#1~arguments;~got~#2!
2165 }
2166 \msg_new:nnn{stex}{error/unknownnotation}{
2167   Unknown~notation~#1~for~#2!
2168 }
```

### 29.1 Symbol Declarations

```
2168 <@@=stex_symdecl>
\stex_all_symbols:n Map over all available symbols
2169 \cs_new_protected:Nn \stex_all_symbols:n {
2170   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
2171   \seq_map_inline:Nn \l_stex_all_modules_seq {
2172     \seq_map_inline:cn{c_stex_module_##1_constants}{
2173       \__stex_symdecl_all_symbols_cs{##1?####1}
2174     }
2175   }
2176 }
```

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

## `\STEXsymbol`

```
2177 \NewDocumentCommand \STEXsymbol { m } {
2178   \stex_get_symbol:n { #1 }
2179   \exp_args:No
2180   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
2181 }
```

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

`symdecl` arguments:

```
2182 \keys_define:nn { stex / symdecl } {
2183   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2184   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2185   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2186   type      .tl_set:N = \l_stex_symdecl_type_tl ,
2187   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,
2188   align     .str_set:N = \l_stex_symdecl_align_str , % TODO(?)
2189   gfc       .str_set:N = \l_stex_symdecl_gfc_str , % TODO(?)
2190   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
2191   def       .tl_set:N = \l_stex_symdecl_definiens_tl ,
2192   reorder   .str_set_x:N = \l_stex_symdecl_reorder_str ,
2193   assoc     .choices:nn =
2194     {bin,binl,binr,pre,conj,pwconj}
2195     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}
2196 }
2197
2198 \bool_new:N \l_stex_symdecl_make_macro_bool
2199
2200 \cs_new_protected:Nn \__stex_symdecl_args:n {
2201   \str_clear:N \l_stex_symdecl_name_str
2202   \str_clear:N \l_stex_symdecl_args_str
2203   \str_clear:N \l_stex_symdecl_deprecate_str
2204   \str_clear:N \l_stex_symdecl_reorder_str
2205   \str_clear:N \l_stex_symdecl_assoctype_str
2206   \bool_set_false:N \l_stex_symdecl_local_bool
2207   \tl_clear:N \l_stex_symdecl_type_tl
2208   \tl_clear:N \l_stex_symdecl_definiens_tl
2209
2210   \keys_set:nn { stex / symdecl } { #1 }
2211 }
```

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

```
2212
2213 \NewDocumentCommand \symdecl { s m O{} } {
2214   \__stex_symdecl_args:n { #3 }
2215   \IfBooleanTF #1 {
2216     \bool_set_false:N \l_stex_symdecl_make_macro_bool
2217   } {
2218     \bool_set_true:N \l_stex_symdecl_make_macro_bool
2219   }
2220   \stex_symdecl_do:n { #2 }
2221   \stex_smsmode_do:
2222 }
```

```

2223
2224 \cs_new_protected:Nn \stex_symdecl_do:nn {
2225   \__stex_symdecl_args:n{#1}
2226   \bool_set_false:N \l_stex_symdecl_make_macro_bool
2227   \stex_symdecl_do:n{#2}
2228 }
2229
2230 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

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

```

2231 \cs_new_protected:Nn \stex_symdecl_do:n {
2232   \stex_if_in_module:F {
2233     % TODO throw error? some default namespace?
2234   }
2235
2236   \str_if_empty:NT \l_stex_symdecl_name_str {
2237     \str_set:Nx \l_stex_symdecl_name_str { #1 }
2238   }
2239
2240   \prop_if_exist:cT { l_stex_symdecl_
2241     \l_stex_current_module_str ?
2242     \l_stex_symdecl_name_str
2243     _prop
2244   }{
2245     % TODO throw error (beware of circular dependencies)
2246   }
2247
2248   \prop_clear:N \l_tmpa_prop
2249   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
2250   \seq_clear:N \l_tmpa_seq
2251   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
2252   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
2253
2254   \str_if_empty:NT \l_stex_symdecl_deprecate_str {
2255     \str_if_empty:NF \l_stex_module_deprecate_str {
2256       \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
2257     }
2258   }
2259   \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
2260
2261   \exp_args:No \stex_add_constant_to_current_module:n {
2262     \l_stex_symdecl_name_str
2263   }
2264
2265   % arity/args
2266   \int_zero:N \l_tmpb_int
2267
2268   \bool_set_true:N \l_tmpa_bool
2269   \str_map_inline:Nn \l_stex_symdecl_args_str {
2270     \token_case_meaning:NnF ##1 {
2271       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2272       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }

```



```

2273     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2274     {\tl_to_str:n a} {
2275         \bool_set_false:N \l_tmpa_bool
2276         \int_incr:N \l_tmpb_int
2277     }
2278     {\tl_to_str:n B} {
2279         \bool_set_false:N \l_tmpa_bool
2280         \int_incr:N \l_tmpb_int
2281     }
2282 }{
2283     \msg_error:nnxx{stex}{error/wrongargs}{
2284         \l_stex_current_module_str ?
2285         \l_stex_symdecl_name_str
2286     }{##1}
2287 }
2288 }
2289 \bool_if:NTF \l_tmpa_bool {
2290     % possibly numeric
2291     \str_if_empty:NTF \l_stex_symdecl_args_str {
2292         \prop_put:Nnn \l_tmpa_prop { args } {}
2293         \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2294     }{
2295         \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
2296         \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2297         \str_clear:N \l_tmpa_str
2298         \int_step_inline:nn \l_tmpa_int {
2299             \str_put_right:Nn \l_tmpa_str i
2300         }
2301         \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
2302     }
2303 } {
2304     \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2305     \prop_put:Nnx \l_tmpa_prop { arity }
2306     { \str_count:N \l_stex_symdecl_args_str }
2307 }
2308 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2309
2310 \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
2311     \prop_put:Nnx \l_tmpa_prop { defined }{ false }
2312 }{
2313     \prop_put:Nnx \l_tmpa_prop { defined }{ true }
2314 }
2315
2316 % semantic macro
2317
2318 \bool_if:NT \l_stex_symdecl_make_macro_bool {
2319     \exp_args:Nx \stex_do_up_to_module:n {
2320         \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2321             \l_stex_current_module_str ? \l_stex_symdecl_name_str
2322         }}
2323     }
2324 }
2325
2326 \stex_debug:nn{symbols}{New~symbol:~

```

```

2327 \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2328 Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2329 Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
2330 Definiens:~\exp_not:o {\l_stex_symdecl_definiens_tl}
2331 }
2332
2333 % circular dependencies require this:
2334 \stex_if_do_html:T {
2335   \stex_annotate_invisible:nnn {symdecl} {
2336     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2337   } {
2338     \tl_if_empty:NF \l_stex_symdecl_type_tl {
2339       \stex_annotate_invisible:nnn{type}{\l_stex_symdecl_type_tl$}
2340     }
2341     \stex_annotate_invisible:nnn{args}{\prop_item:Nn \l_tmpa_prop { args }}{ }
2342     \stex_annotate_invisible:nnn{macroname}{#1}{ }
2343     \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2344       \stex_annotate_invisible:nnn{definiens}{ }
2345       {\l_stex_symdecl_definiens_tl$}
2346     }
2347     \str_if_empty:NF \l_stex_symdecl_assoc_type_str {
2348       \stex_annotate_invisible:nnn{assoc_type}{\l_stex_symdecl_assoc_type_str}{ }
2349     }
2350     \str_if_empty:NF \l_stex_symdecl_reorder_str {
2351       \stex_annotate_invisible:nnn{reorderargs}{\l_stex_symdecl_reorder_str}{ }
2352     }
2353   }
2354 }
2355 \prop_if_exist:cF {
2356   \l_stex_symdecl_
2357   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2358   _prop
2359 } {
2360   \bool_if:NTF \l_stex_symdecl_local_bool \stex_do_up_to_module:x \stex_execute_in_module:
2361     \__stex_symdecl_restore_symbol:nnnnnnn
2362     {\l_stex_symdecl_name_str}
2363     { \prop_item:Nn \l_tmpa_prop {args} }
2364     { \prop_item:Nn \l_tmpa_prop {arity} }
2365     { \prop_item:Nn \l_tmpa_prop {assocs} }
2366     { \prop_item:Nn \l_tmpa_prop {defined} }
2367     {\bool_if:NT \l_stex_symdecl_make_macro_bool {#1} }
2368     {\l_stex_current_module_str}
2369   }
2370 }
2371 }
2372 \cs_new_protected:Nn \__stex_symdecl_restore_symbol:nnnnnnn {
2373   \prop_clear:N \l_tmpa_prop
2374   \prop_put:Nnn \l_tmpa_prop { module } { #7 }
2375   \prop_put:Nnn \l_tmpa_prop { name } { #1 }
2376   \prop_put:Nnn \l_tmpa_prop { args } { #2 }
2377   \prop_put:Nnn \l_tmpa_prop { arity } { #3 }
2378   \prop_put:Nnn \l_tmpa_prop { assocs } { #4 }
2379   \prop_put:Nnn \l_tmpa_prop { defined } { #5 }
2380   \tl_if_empty:nF{#6}{

```

```

2381 \tl_set:cx{#6}{\stex_invoke_symbol:n{\detokenize{#7 ? #1}}}
2382 }
2383 \prop_set_eq:cN{l_stex_symdecl_ \detokenize{#7 ? #1} _prop}\l_tmpa_prop
2384 \seq_clear:c{l_stex_symdecl_ \detokenize{#7 ? #1} _notations}
2385 }

```

(End definition for \stex\_symdecl\_do:n. This function is documented on page 78.)

**\stex\_get\_symbol:n**

```

2386 \str_new:N \l_stex_get_symbol_uri_str
2387
2388 \cs_new_protected:Nn \stex_get_symbol:n {
2389   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2390     \tl_set:Nn \l_tmpa_tl { #1 }
2391     \__stex_symdecl_get_symbol_from_cs:
2392   }{
2393     % argument is a string
2394     % is it a command name?
2395     \cs_if_exist:cTF { #1 }{
2396       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2397       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2398       \str_if_empty:NTF \l_tmpa_str {
2399         \exp_args:Nx \cs_if_eq:NNTF {
2400           \tl_head:N \l_tmpa_tl
2401         } \stex_invoke_symbol:n {
2402           \__stex_symdecl_get_symbol_from_cs:
2403         }{
2404           \__stex_symdecl_get_symbol_from_string:n { #1 }
2405         }
2406       } {
2407         \__stex_symdecl_get_symbol_from_string:n { #1 }
2408       }
2409     }{
2410       % argument is not a command name
2411       \__stex_symdecl_get_symbol_from_string:n { #1 }
2412       % \l_stex_all_symbols_seq
2413     }
2414   }
2415   \str_if_eq:eeF {
2416     \prop_item:cn {
2417       l_stex_symdecl_\l_stex_get_symbol_uri_str _prop
2418     }{ deprecate }
2419   }{}{
2420     \msg_warning:nnxx{stex}{warning/deprecated}{
2421       Symbol~\l_stex_get_symbol_uri_str
2422     }{
2423       \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{ deprecate }
2424     }
2425   }
2426 }
2427
2428 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2429   \tl_set:Nn \l_tmpa_tl {
2430     \msg_error:nnn{stex}{error/unknownsymbol}{#1}

```

```

2431 }
2432 \str_set:Nn \l_tmpa_str { #1 }
2433
2434 %\int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2435
2436 \str_if_in:NnTF \l_tmpa_str ? {
2437   \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
2438   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
2439   \str_set:Nx \l_tmpb_str {\seq_use:Nn \l_tmpa_seq ?}
2440 }{
2441   \str_clear:N \l_tmpb_str
2442 }
2443 \str_if_empty:NTF \l_tmpb_str {
2444   \seq_map_inline:Nn \l_stex_all_modules_seq {
2445     \seq_map_inline:cn{c_stex_module_###1_constants}{
2446       \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2447         \seq_map_break:n{\seq_map_break:n{
2448           \tl_set:Nn \l_tmpa_tl {
2449             \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2450           }
2451         }}
2452       }
2453     }
2454   }
2455 }{
2456   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpb_str }
2457   \seq_map_inline:Nn \l_stex_all_modules_seq {
2458     \str_if_eq:eeT{ \l_tmpb_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2459       \seq_map_inline:cn{c_stex_module_###1_constants}{
2460         \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2461           \seq_map_break:n{\seq_map_break:n{
2462             \tl_set:Nn \l_tmpa_tl {
2463               \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2464             }
2465           }}
2466         }
2467       }
2468     }
2469   }
2470 }
2471
2472 \l_tmpa_tl
2473 }
2474
2475 \cs_new_protected:Nn \_stex_symdecl_get_symbol_from_cs: {
2476   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2477   { \tl_tail:N \l_tmpa_tl }
2478   \tl_if_single:NTF \l_tmpa_tl {
2479     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2480       \exp_after:wN \str_set:Nn \exp_after:wN
2481       \l_stex_get_symbol_uri_str \l_tmpa_tl
2482     }{
2483       % TODO
2484       % tail is not a single group

```

```

2485     }
2486   }{
2487     % TODO
2488     % tail is not a single group
2489   }
2490 }

```

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

## 29.2 Notations

```

2491 <@@=stex_notation>

notation arguments:
2492 \keys_define:nn { stex / notation } {
2493   % lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2494   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2495   prec     .str_set_x:N = \l__stex_notation_prec_str ,
2496   op       .tl_set:N = \l__stex_notation_op_tl ,
2497   primary .bool_set:N = \l__stex_notation_primary_bool ,
2498   primary .default:n = {true} ,
2499   unknown .code:n = \str_set:Nx
2500     \l__stex_notation_variant_str \l_keys_key_str
2501 }
2502
2503 \cs_new_protected:Nn \_stex_notation_args:n {
2504   % \str_clear:N \l__stex_notation_lang_str
2505   \str_clear:N \l__stex_notation_variant_str
2506   \str_clear:N \l__stex_notation_prec_str
2507   \tl_clear:N \l__stex_notation_op_tl
2508   \bool_set_false:N \l__stex_notation_primary_bool
2509
2510   \keys_set:nn { stex / notation } { #1 }
2511 }

```

**\notation**

```

2512 \NewDocumentCommand \notation { s m O{}} {
2513   \_stex_notation_args:n { #3 }
2514   \tl_clear:N \l_stex_symdecl_definiens_tl
2515   \stex_get_symbol:n { #2 }
2516   \tl_set:Nn \l_stex_notation_after_do_tl {
2517     \__stex_notation_final:
2518     \IfBooleanTF#1{
2519       \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2520     }{}
2521     \stex_smsmode_do:\ignorespacesandpars
2522   }
2523   \stex_notation_do:nnnnn
2524     { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str_prop} { args } }
2525     { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str_prop} { arity } }
2526     { \l__stex_notation_variant_str }
2527     { \l__stex_notation_prec_str }
2528   }
2529   \stex_deactivate_macro:Nn \notation {module~environments}

```

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

\stex\_notation\_do:nnnnn

```

2530 \seq_new:N \l__stex_notation_precedences_seq
2531 \tl_new:N \l__stex_notation_opprec_tl
2532 \int_new:N \l__stex_notation_currarg_int
2533 \tl_new:N \stex_symbol_after_invokation_tl
2534
2535 \cs_new_protected:Nn \stex_notation_do:nnnnn {
2536   \let\l_stex_current_symbol_str\relax
2537   \seq_clear:N \l__stex_notation_precedences_seq
2538   \tl_clear:N \l__stex_notation_opprec_tl
2539   \str_set:Nx \l__stex_notation_args_str { #1 }
2540   \str_set:Nx \l__stex_notation_arity_str { #2 }
2541   \str_set:Nx \l__stex_notation_suffix_str { #3 }
2542   \str_set:Nx \l__stex_notation_prec_str { #4 }
2543
2544   % precedences
2545   \str_if_empty:NTF \l__stex_notation_prec_str {
2546     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2547       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2548     }{
2549       \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2550     }
2551   } {
2552     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2553       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2554       \int_step_inline:nn { \l__stex_notation_arity_str } {
2555         \exp_args:NNno
2556         \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2557       }
2558     }{
2559       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2560       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2561         \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2562         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2563           \exp_args:NNno \exp_args:NNno \seq_set_split:Nnn
2564           \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2565           \seq_map_inline:Nn \l_tmpa_seq {
2566             \seq_put_right:Nn \l_tmpb_seq { ##1 }
2567           }
2568         }
2569       }{
2570         \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2571           \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2572         }{
2573           \tl_set:No \l__stex_notation_opprec_tl { 0 }
2574         }
2575       }
2576     }
2577   }
2578
2579   \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2580   \int_step_inline:nn { \l__stex_notation_arity_str } {

```

```

2581 \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
2582 \exp_args:NNo
2583 \seq_put_right:No \l__stex_notation_precedences_seq {
2584 \l__stex_notation_opprec_tl
2585 }
2586 }
2587 }
2588 \tl_clear:N \l_stex_notation_dummyargs_tl
2589
2590 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2591 \exp_args:NNe
2592 \cs_set:Npn \l_stex_notation_macrocode_cs {
2593 \stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2594 { \l__stex_notation_suffix_str }
2595 { \l__stex_notation_opprec_tl }
2596 { \exp_not:n { #5 } }
2597 }
2598 \l_stex_notation_after_do_tl
2599 }{
2600 \str_if_in:NnTF \l__stex_notation_args_str b {
2601 \exp_args:Nne \use:nn
2602 {
2603 \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2604 \cs_set:Npn \l__stex_notation_arity_str } { {
2605 \stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2606 { \l__stex_notation_suffix_str }
2607 { \l__stex_notation_opprec_tl }
2608 { \exp_not:n { #5 } }
2609 }}
2610 }{
2611 \str_if_in:NnTF \l__stex_notation_args_str B {
2612 \exp_args:Nne \use:nn
2613 {
2614 \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2615 \cs_set:Npn \l__stex_notation_arity_str } { {
2616 \stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2617 { \l__stex_notation_suffix_str }
2618 { \l__stex_notation_opprec_tl }
2619 { \exp_not:n { #5 } }
2620 } }
2621 }{
2622 \exp_args:Nne \use:nn
2623 {
2624 \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2625 \cs_set:Npn \l__stex_notation_arity_str } { {
2626 \stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2627 { \l__stex_notation_suffix_str }
2628 { \l__stex_notation_opprec_tl }
2629 { \exp_not:n { #5 } }
2630 } }
2631 }
2632 }
2633
2634 \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str

```

```

2635 \int_zero:N \l__stex_notation_currarg_int
2636 \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2637 \__stex_notation_arguments:
2638 }
2639 }

```

(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

```

2640 \cs_new_protected:Nn \__stex_notation_arguments: {
2641 \int_incr:N \l__stex_notation_currarg_int
2642 \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2643 \l_stex_notation_after_do_tl
2644 }{
2645 \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2646 \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2647 \str_if_eq:VnTF \l_tmpa_str a {
2648 \__stex_notation_argument_assoc:nn{a}
2649 }{
2650 \str_if_eq:VnTF \l_tmpa_str B {
2651 \__stex_notation_argument_assoc:nn{B}
2652 }{
2653 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpb_str
2654 \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2655 { \_stex_term_math_arg:nnn
2656 { \l_tmpa_str\int_use:N \l__stex_notation_currarg_int }
2657 { \l_tmpb_str }
2658 { ####\int_use:N \l__stex_notation_currarg_int }
2659 }
2660 }
2661 \__stex_notation_arguments:
2662 }
2663 }
2664 }
2665 }

```

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

\\_stex\_notation\_argument\_assoc:nn

```

2666 \cs_new_protected:Nn \_stex_notation_argument_assoc:nn {
2667
2668 \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2669 {\l__stex_notation_arity_str}{
2670 #2
2671 }
2672 \int_zero:N \l_tmpa_int
2673 \tl_clear:N \l_tmpa_tl
2674 \str_map_inline:Nn \l__stex_notation_args_str {
2675 \int_incr:N \l_tmpa_int
2676 \tl_put_right:Nx \l_tmpa_tl {
2677 \str_if_eq:nnTF {##1}{a}{ {} }{
2678 \str_if_eq:nnTF {##1}{B}{ {} }{
2679 {\_stex_term_arg:nn{##1}\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa
2680 }

```



```

2681     }
2682   }
2683 }
2684 \exp_after:wN\exp_after:wN\exp_after:wN \def
2685 \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2686 \exp_after:wN\exp_after:wN\exp_after:wN ##
2687 \exp_after:wN\exp_after:wN\exp_after:wN 1
2688 \exp_after:wN\exp_after:wN\exp_after:wN ##
2689 \exp_after:wN\exp_after:wN\exp_after:wN 2
2690 \exp_after:wN\exp_after:wN\exp_after:wN {
2691   \exp_after:wN \exp_after:wN \exp_after:wN
2692   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2693     \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2694   }
2695 }
2696
2697 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2698 \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
2699   \stex_term_math_assoc_arg:nnnn
2700   { #1\int_use:N \l__stex_notation_currarg_int }
2701   { \l_tmpa_str }
2702   { ####\int_use:N \l__stex_notation_currarg_int }
2703   { \l_tmpa_cs {####1} {####2} }
2704 } }
2705 \__stex_notation_arguments:
2706 }

```

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

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

```

2707 \cs_new_protected:Nn \__stex_notation_restore_notation:nnnnn {
2708   \cs_generate_from_arg_count:cNnn{stex_notation_\detokenize{#1} \c_hash_str \detokenize{#2}}
2709   \cs_set_nopar:Npn {#3}{#4}
2710   \tl_if_empty:nF {#5}{
2711     \tl_set:cn{stex_op_notation_\detokenize{#1} \c_hash_str \detokenize{#2}_cs}{ \comp{ #5 }
2712   }
2713   \seq_if_exist:cT { l_stex_symdecl_\detokenize{#1} _notations }{
2714     \seq_put_right:cx { l_stex_symdecl_\detokenize{#1} _notations } { \detokenize{#2} }
2715   }
2716 }
2717
2718 \cs_new_protected:Nn \__stex_notation_final: {
2719
2720   \stex_execute_in_module:x {
2721     \__stex_notation_restore_notation:nnnnn
2722     {\l_stex_get_symbol_uri_str}
2723     {\l__stex_notation_suffix_str}
2724     {\l__stex_notation_arity_str}
2725     {
2726       \exp_after:wN \exp_after:wN \exp_after:wN
2727       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2728       { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2729     }
2730     {\exp_args:No \exp_not:n \l__stex_notation_op_tl }

```

```

2731 }
2732
2733 \stex_debug:nn{symbols}{
2734   Notation~\l__stex_notation_suffix_str
2735   ~for~\l_stex_get_symbol_uri_str^^J
2736   Operator~precedence:~\l__stex_notation_opprec_tl^^J
2737   Argument~precedences:~
2738   \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2739   Notation: \cs_meaning:c {
2740     stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2741     \l__stex_notation_suffix_str
2742     _cs
2743   }
2744 }
2745 % HTML annotations
2746 \stex_if_do_html:T {
2747   \stex_annotate_invisible:nnn { notation }
2748   { \l_stex_get_symbol_uri_str } {
2749     \stex_annotate_invisible:nnn { notationfragment }
2750     { \l__stex_notation_suffix_str }{}
2751     \stex_annotate_invisible:nnn { precedence }
2752     { \l__stex_notation_prec_str }{}
2753
2754     \int_zero:N \l_tmpa_int
2755     \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2756     \tl_clear:N \l_tmpa_tl
2757     \int_step_inline:nn { \l__stex_notation_arity_str }{
2758       \int_incr:N \l_tmpa_int
2759       \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2760       \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_rema
2761       \str_if_eq:VnTF \l_tmpb_str a {
2762         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2763           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2764           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2765         } }
2766       }{
2767         \str_if_eq:VnTF \l_tmpb_str B {
2768           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2769             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2770             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2771           } }
2772         }{
2773           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2774             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
2775           } }
2776         }
2777       }
2778     }
2779     \stex_annotate_invisible:nnn { notationcomp }{}{
2780       \str_set:Nx \l_stex_current_symbol_str {\l_stex_get_symbol_uri_str }
2781       $ \exp_args:Nno \use:nn { \use:c {
2782         stex_notation_ \l_stex_current_symbol_str
2783         \c_hash_str \l__stex_notation_suffix_str _cs
2784       } } { \l_tmpa_tl } $

```

```

2785     }
2786     \tl_if_empty:NF \l__stex_notation_op_tl {
2787         \stex_annotate_invisible:nnn { notationopcomp }{}{
2788             $\l__stex_notation_op_tl$
2789         }
2790     }
2791 }
2792 }
2793 }

```

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

## \setnotation

```

2794 \keys_define:nn { stex / setnotation } {
2795   % lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2796   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2797   unknown .code:n = \str_set:Nx
2798       \l__stex_notation_variant_str \l_keys_key_str
2799 }
2800
2801 \cs_new_protected:Nn \_stex_setnotation_args:n {
2802   % \str_clear:N \l__stex_notation_lang_str
2803   \str_clear:N \l__stex_notation_variant_str
2804   \keys_set:nn { stex / setnotation } { #1 }
2805 }
2806
2807 \cs_new_protected:Nn \_stex_notation_setnotation:nn {
2808   \seq_if_exist:cT{l_stex_symdecl_#1_notations}{
2809     \seq_remove_all:cn { l_stex_symdecl_#1_notations }{ #2 }
2810     \seq_put_left:cn { l_stex_symdecl_#1_notations }{ #2 }
2811   }
2812 }
2813
2814 \cs_new_protected:Nn \_stex_setnotation:n {
2815   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1_notations }
2816   { \l__stex_notation_variant_str }{
2817     \stex_execute_in_module:x{ \_stex_notation_setnotation:nn {#1}{\l__stex_notation_vari
2818     \stex_debug:nn {notations}{
2819       Setting~default~notation~
2820       {\l__stex_notation_variant_str }~for~
2821       #1 \\
2822       \expandafter\meaning\csname
2823       l_stex_symdecl_#1_notations\endcsname
2824     }
2825   }{
2826     \msg_error:nxxx{stex}{unknownnotation}{\l__stex_notation_variant_str}{#1}
2827   }
2828 }
2829
2830 \NewDocumentCommand \setnotation {m m} {
2831   \stex_get_symbol:n { #1 }
2832   \_stex_setnotation_args:n { #2 }
2833   \stex_setnotation:n{\l_stex_get_symbol_uri_str}
2834   \stex_smsmode_do:\ignorespacesandpars

```

```

2835 }
2836
2837 \cs_new_protected:Nn \stex_copy_notations:nn {
2838   \stex_debug:nn {notations}{
2839     Copying~notations~from~#2~to~#1\\
2840     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
2841   }
2842   \tl_clear:N \l_tmpa_tl
2843   \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2844     \tl_put_right:Nn \l_tmpa_tl { {##### #1} }
2845   }
2846   \seq_map_inline:cn {l_stex_symdecl_#2_notations}{\begingroup
2847     \stex_debug:nn{Here}{Here:~##1}
2848     \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2849     \edef \l_tmpa_tl {
2850       \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2851       \exp_after:wN\exp_after:wN\exp_after:wN {
2852         \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2853       }
2854     }
2855
2856     \exp_after:wN \def \exp_after:wN \l_tmpa_tl
2857     \exp_after:wN ####\exp_after:wN 1 \exp_after:wN ####\exp_after:wN 2
2858     \exp_after:wN { \l_tmpa_tl }
2859
2860     \edef \l_tmpa_tl {
2861       \exp_after:wN \exp_not:n \exp_after:wN {
2862         \l_tmpa_tl {##### 1}{##### 2}
2863       }
2864     }
2865
2866     \stex_debug:nn{Here}{Here:~\expandafter\detokenize\expandafter{\l_tmpa_tl}}
2867
2868   \stex_execute_in_module:x {
2869     \__stex_notation_restore_notation:nnnnn
2870     {#1}{##1}
2871     { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }
2872     { \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl} }
2873     {
2874       \cs_if_exist:cT{stex_op_notation_ #2\c_hash_str ##1 _cs}{
2875         \exp_args:NNo\exp_args:No\exp_not:n{\csname stex_op_notation_ #2\c_hash_str ##1
2876       }
2877     }
2878   }\endgroup
2879 }
2880 }
2881
2882 \NewDocumentCommand \copynotation {m m} {
2883   \stex_get_symbol:n { #1 }
2884   \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2885   \stex_get_symbol:n { #2 }
2886   \exp_args:Noo
2887   \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2888   \stex_smsmode_do:\ignorespacesandpars

```

```

2889 }
2890

```

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

**\symdef**

```

2891 \keys_define:nn { stex / symdef } {
2892   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2893   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2894   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2895   type      .tl_set:N   = \l_stex_symdecl_type_tl ,
2896   def       .tl_set:N   = \l_stex_symdecl_definiens_tl ,
2897   reorder   .str_set_x:N = \l_stex_symdecl_reorder_str ,
2898   op        .tl_set:N   = \l__stex_notation_op_tl ,
2899   % lang     .str_set_x:N = \l__stex_notation_lang_str ,
2900   variant   .str_set_x:N = \l__stex_notation_variant_str ,
2901   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2902   assoc     .choices:nn =
2903     {bin,binl,binr,pre,conj,pwconj}
2904     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
2905   unknown   .code:n     = \str_set:Nx
2906     \l__stex_notation_variant_str \l_keys_key_str
2907 }
2908
2909 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2910   \str_clear:N \l_stex_symdecl_name_str
2911   \str_clear:N \l_stex_symdecl_args_str
2912   \str_clear:N \l_stex_symdecl_assoctype_str
2913   \str_clear:N \l_stex_symdecl_reorder_str
2914   \bool_set_false:N \l_stex_symdecl_local_bool
2915   \tl_clear:N \l_stex_symdecl_type_tl
2916   \tl_clear:N \l_stex_symdecl_definiens_tl
2917   % \str_clear:N \l__stex_notation_lang_str
2918   \str_clear:N \l__stex_notation_variant_str
2919   \str_clear:N \l__stex_notation_prec_str
2920   \tl_clear:N \l__stex_notation_op_tl
2921
2922   \keys_set:nn { stex / symdef } { #1 }
2923 }
2924
2925 \NewDocumentCommand \symdef { m O{} } {
2926   \__stex_notation_symdef_args:n { #2 }
2927   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2928   \stex_symdecl_do:n { #1 }
2929   \tl_set:Nn \l_stex_notation_after_do_tl {
2930     \__stex_notation_final:
2931     \stex_smsmode_do:\ignorespacesandpars
2932   }
2933   \str_set:Nx \l_stex_get_symbol_uri_str {
2934     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2935   }
2936   \exp_args:Nx \stex_notation_do:nnnnn
2937     { \prop_item:cn { \l_stex_symdecl \l_stex_get_symbol_uri_str _prop } { args } }
2938     { \prop_item:cn { \l_stex_symdecl \l_stex_get_symbol_uri_str _prop } { arity } }

```

```

2939 { \l__stex_notation_variant_str }
2940 { \l__stex_notation_prec_str}
2941 }
2942 \stex_deactivate_macro:Nn \symdef {module~environments}

```

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

## 29.3 Variables

```

2943 <@@=stex_variables>
2944
2945 \keys_define:nn { stex / vardef } {
2946   name      .str_set_x:N = \l__stex_variables_name_str ,
2947   args      .str_set_x:N = \l__stex_variables_args_str ,
2948   type      .tl_set:N    = \l__stex_variables_type_tl ,
2949   def       .tl_set:N    = \l__stex_variables_def_tl ,
2950   op        .tl_set:N    = \l__stex_variables_op_tl ,
2951   prec      .str_set_x:N = \l__stex_variables_prec_str ,
2952   reorder   .str_set_x:N = \l__stex_variables_reorder_str ,
2953   assoc     .choices:nn  =
2954     {bin,binl,binr,pre,conj,pwconj}
2955     {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
2956   bind      .choices:nn  =
2957     {forall,exists}
2958     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2959 }
2960
2961 \cs_new_protected:Nn \__stex_variables_args:n {
2962   \str_clear:N \l__stex_variables_name_str
2963   \str_clear:N \l__stex_variables_args_str
2964   \str_clear:N \l__stex_variables_prec_str
2965   \str_clear:N \l__stex_variables_assoctype_str
2966   \str_clear:N \l__stex_variables_reorder_str
2967   \str_clear:N \l__stex_variables_bind_str
2968   \tl_clear:N \l__stex_variables_type_tl
2969   \tl_clear:N \l__stex_variables_def_tl
2970   \tl_clear:N \l__stex_variables_op_tl
2971
2972   \keys_set:nn { stex / vardef } { #1 }
2973 }
2974
2975 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{} } {
2976   \__stex_variables_args:n {#2}
2977   \str_if_empty:NT \l__stex_variables_name_str {
2978     \str_set:Nx \l__stex_variables_name_str { #1 }
2979   }
2980   \prop_clear:N \l_tmpa_prop
2981   \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
2982
2983   \int_zero:N \l_tmpb_int
2984   \bool_set_true:N \l_tmpa_bool
2985   \str_map_inline:Nn \l__stex_variables_args_str {
2986     \token_case_meaning:NnF ##1 {
2987       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}

```

```

2988     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2989     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2990     {\tl_to_str:n a} {
2991         \bool_set_false:N \l_tmpa_bool
2992         \int_incr:N \l_tmpb_int
2993     }
2994     {\tl_to_str:n B} {
2995         \bool_set_false:N \l_tmpa_bool
2996         \int_incr:N \l_tmpb_int
2997     }
2998     ){
2999     \msg_error:nnxx{stex}{error/wrongargs}{
3000         variable~\l__stex_variables_name_str
3001     }{##1}
3002     }
3003 }
3004 \bool_if:NTF \l_tmpa_bool {
3005     % possibly numeric
3006     \str_if_empty:NTF \l__stex_variables_args_str {
3007         \prop_put:Nnn \l_tmpa_prop { args } {}
3008         \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
3009     }{
3010         \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
3011         \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
3012         \str_clear:N \l_tmpa_str
3013         \int_step_inline:nn \l_tmpa_int {
3014             \str_put_right:Nn \l_tmpa_str i
3015         }
3016         \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
3017         \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3018     }
3019 } {
3020     \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3021     \prop_put:Nnx \l_tmpa_prop { arity }
3022     { \str_count:N \l__stex_variables_args_str }
3023 }
3024 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
3025 \tl_set:cx { #1 }{ \stex_invoke_variable:n { \l__stex_variables_name_str } }
3026
3027 \prop_set_eq:cN { l_stex_variable_\l__stex_variables_name_str _prop } \l_tmpa_prop
3028
3029 \tl_if_empty:NF \l__stex_variables_op_tl {
3030     \cs_set:cpx {
3031         stex_var_op_notation_\l__stex_variables_name_str _cs
3032     } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
3033 }
3034
3035 \tl_set:Nn \l_stex_notation_after_do_tl {
3036     \exp_args:Nne \use:nn {
3037         \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str _cs }
3038         \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
3039     } {{
3040         \exp_after:wN \exp_after:wN \exp_after:wN
3041         \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN

```

```

3042 { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_symbol
3043 }
3044 \stex_if_do_html:T {
3045   \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
3046     \stex_annotate_invisible:nnn { precedence }
3047     { \l__stex_variables_prec_str }{}
3048     \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{\l__stex_variables_type_tl}{\l__stex_variables_type_tl}}
3049     \stex_annotate_invisible:nnn{args}{\l__stex_variables_args_str }{}
3050     \stex_annotate_invisible:nnn{macroname}{#1}{\l__stex_variables_macroname_tl}
3051     \tl_if_empty:NF \l__stex_variables_def_tl {
3052       \stex_annotate_invisible:nnn{definiens}{\l__stex_variables_def_tl}
3053       {\l__stex_variables_def_tl}
3054     }
3055     \str_if_empty:NF \l__stex_variables ASSOCTYPE_str {
3056       \stex_annotate_invisible:nnn{ASSOCTYPE}{\l__stex_variables ASSOCTYPE_str}{}
3057     }
3058     \str_if_empty:NF \l__stex_variables REORDER_str {
3059       \stex_annotate_invisible:nnn{REORDER}{\l__stex_variables REORDER_str}{}
3060     }
3061     \int_zero:N \l_tmpa_int
3062     \str_set_eq:NN \l__stex_variables REMAINING_ARGS_str \l__stex_variables ARGS_str
3063     \tl_clear:N \l_tmpa_tl
3064     \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
3065       \int_incr:N \l_tmpa_int
3066       \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables REMAINING_ARGS_str }
3067       \str_set:Nx \l__stex_variables REMAINING_ARGS_str { \str_tail:N \l__stex_variables REMAINING_ARGS_str }
3068       \str_if_eq:VnTF \l_tmpb_str a {
3069         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3070           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{\l__stex_variables_ARGMARKER} ,
3071           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{\l__stex_variables_ARGMARKER}
3072         } }
3073       }{
3074         \str_if_eq:VnTF \l_tmpb_str B {
3075           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3076             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{\l__stex_variables_ARGMARKER} ,
3077             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{\l__stex_variables_ARGMARKER}
3078           } }
3079         }{
3080           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3081             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{\l__stex_variables_ARGMARKER}
3082           } }
3083         }
3084       }
3085     }
3086     \stex_annotate_invisible:nnn { notationcomp }{}{
3087       \str_set:Nx \l_stex_current_symbol_str {var://\l__stex_variables_name_str }
3088       $ \exp_args:Nno \use:nn { \use:c {
3089         stex_var_notation_\l__stex_variables_name_str_cs
3090       } } { \l_tmpa_tl } $
3091     }
3092     \tl_if_empty:NF \l__stex_variables_OP_tl {
3093       \stex_annotate_invisible:nnn { notationopcomp }{}{
3094         $\l__stex_variables_OP_tl$
3095       }

```



```

3096     }
3097   }
3098   \str_if_empty:NF \l__stex_variables_bind_str {
3099     \stex_annotate_invisible:nnn {bindtype}{\l__stex_variables_bind_str,\l__stex_variab
3100   }
3101   }\ignorespacesandpars
3102 }
3103
3104 \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
3105 }
3106
3107 \cs_new:Nn \_stex_reset:N {
3108   \tl_if_exist:NTF #1 {
3109     \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
3110   }{
3111     \let \exp_not:N #1 \exp_not:N \undefined
3112   }
3113 }
3114
3115 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
3116   \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
3117   \exp_args:Nnx \use:nn {
3118     % TODO
3119     \stex_annotate_invisible:nnn {vardecl}{\clist_use:Nn\l__stex_variables_names,}{
3120       #2
3121     }
3122   }{
3123     \_stex_reset:N \varnot
3124     \_stex_reset:N \vartype
3125     \_stex_reset:N \vardefi
3126   }
3127 }
3128
3129 \NewDocumentCommand \vardef { s } {
3130   \IfBooleanTF#1 {
3131     \__stex_variables_do_complex:nn
3132   }{
3133     \__stex_variables_do_simple:nnn
3134   }
3135 }
3136
3137 \NewDocumentCommand \svar { 0{} m }{
3138   \tl_if_empty:nTF {#1}{
3139     \str_set:Nn \l_tmpa_str { #2 }
3140   }{
3141     \str_set:Nn \l_tmpa_str { #1 }
3142   }
3143   \_stex_term_omv:nn {
3144     var://\l_tmpa_str
3145   }{
3146     \exp_args:Nnx \use:nn {
3147       \def\comp{\_varcomp}
3148       \str_set:Nx \l_stex_current_symbol_str { var://\l_tmpa_str }
3149       \comp{ #2 }

```

```

3150   }{
3151     \_stex_reset:N \comp
3152     \_stex_reset:N \l_stex_current_symbol_str
3153   }
3154 }
3155 }
3156
3157
3158
3159 \keys_define:nn { stex / varseq } {
3160   name      .str_set:N = \l__stex_variables_name_str ,
3161   args      .int_set:N = \l__stex_variables_args_int ,
3162   type      .tl_set:N  = \l__stex_variables_type_tl  ,
3163   mid       .tl_set:N  = \l__stex_variables_mid_tl   ,
3164   bind      .choices:nn =
3165     {forall,exists}
3166     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
3167 }
3168
3169 \cs_new_protected:Nn \__stex_variables_seq_args:n {
3170   \str_clear:N \l__stex_variables_name_str
3171   \int_set:Nn \l__stex_variables_args_int 1
3172   \tl_clear:N \l__stex_variables_type_tl
3173   \str_clear:N \l__stex_variables_bind_str
3174
3175   \keys_set:nn { stex / varseq } { #1 }
3176 }
3177
3178 \NewDocumentCommand \varseq {m O{} m m m}{
3179   \__stex_variables_seq_args:n { #2 }
3180   \str_if_empty:NT \l__stex_variables_name_str {
3181     \str_set:Nx \l__stex_variables_name_str { #1 }
3182   }
3183   \prop_clear:N \l_tmpa_prop
3184   \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
3185
3186   \seq_set_from_clist:Nn \l_tmpa_seq {#3}
3187   \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
3188     \msg_error:nxxx{stex}{error/seqlength}
3189     {\int_use:N \l__stex_variables_args_int}
3190     {\seq_count:N \l_tmpa_seq}
3191   }
3192   \seq_set_from_clist:Nn \l_tmpb_seq {#4}
3193   \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
3194     \msg_error:nxxx{stex}{error/seqlength}
3195     {\int_use:N \l__stex_variables_args_int}
3196     {\seq_count:N \l_tmpb_seq}
3197   }
3198   \prop_put:Nnn \l_tmpa_prop {starts} {#3}
3199   \prop_put:Nnn \l_tmpa_prop {ends} {#4}
3200
3201   \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
3202     \cs_set:Npn {\int_use:N \l__stex_variables_args_int} { #5 }
3203

```

```

3204 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3205 \int_step_inline:nn \l__stex_variables_args_int {
3206   \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
3207 }
3208 \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
3209 \tl_put_right:Nn \l_tmpa_tl {\,\ellipses,}
3210 \tl_if_empty:NF \l__stex_variables_mid_tl {
3211   \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
3212   \tl_put_right:Nn \l_tmpa_tl {\,\ellipses,}
3213 }
3214 \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3215 \int_step_inline:nn \l__stex_variables_args_int {
3216   \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
3217 }
3218 \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
3219 \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
3220
3221
3222 \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl
3223
3224 \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}
3225
3226 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3227
3228 \int_step_inline:nn \l__stex_variables_args_int {
3229   \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
3230     \stex_term_math_arg:nnn{i##1}{0}{\exp_not:n{####}##1}
3231   }}
3232 }
3233
3234 \tl_set:Nx \l_tmpa_tl {
3235   \stex_term_math_oma:nnnn { varseq://\l__stex_variables_name_str}{0}{
3236     \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
3237   }
3238 }
3239
3240 \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \stex_symbol_after_invokation_tl} }
3241
3242 \exp_args:Nno \use:nn {
3243   \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
3244   \cs_set:Npn {\int_use:N \l__stex_variables_args_int}{\l_tmpa_tl}
3245
3246   \stex_debug:nn{sequences}{New~Sequence:~
3247     \expandafter\meaning\csname stex_varseq_\l__stex_variables_name_str _cs\endcsname\\~\\
3248     \prop_to_keyval:N \l_tmpa_prop
3249   }
3250   \stex_if_do_html:T{\stex_annotate_invisible:nnn{varseq}{\l__stex_variables_name_str}{
3251     \tl_if_empty:NF \l__stex_variables_type_tl {
3252       \stex_annotate:nnn {type}{\{$\seqtype\l__stex_variables_type_tl$\}
3253     }
3254     \stex_annotate:nnn {args}{\int_use:N \l__stex_variables_args_int}{}
3255     \str_if_empty:NF \l__stex_variables_bind_str {
3256       \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{}
3257     }

```

```

3258   }}
3259
3260   \prop_set_eq:cN {stex_varseq\_l\_stex_variables_name_str _prop}\l_tmpa_prop
3261   \ignorespacesandpars
3262 }
3263
3264 \end{package}

```

## Chapter 30

# STEX -Terms Implementation

```
3265 <*package>
3266
3267 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
3268
3269 <@@=stex_terms>
3270
3271   Warnings and error messages
3272 \msg_new:nnn{stex}{error/nonotation}{
3273   Symbol~#1~invoked,~but~has~no~notation~#2!
3274 }
3275 \msg_new:nnn{stex}{error/notationarg}{
3276   Error~in~parsing~notation~#1
3277 }
3278 \msg_new:nnn{stex}{error/noop}{
3279   Symbol~#1~has~no~operator~notation~for~notation~#2
3280 }
3281 \msg_new:nnn{stex}{error/notallowed}{
3282   Symbol~invocation~#1~not~allowed~in~notation~component~of~#2
3283 }
3284 \msg_new:nnn{stex}{error/doubleargument}{
3285   Argument~#1~of~symbol~#2~already~assigned
3286 }
3287 \msg_new:nnn{stex}{error/overarity}{
3288   Argument~#1~invalid~for~symbol~#2~with~arity~#3
3289 }
3290
```

### 30.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
3289
3290
3291 \bool_new:N \l_stex_allow_semantic_bool
3292 \bool_set_true:N \l_stex_allow_semantic_bool
3293
```

```

3294 \cs_new_protected:Nn \stex_invoke_symbol:n {
3295   \ifvmode\indent\fi
3296   \bool_if:NTF \l_stex_allow_semantic_bool {
3297     \str_if_eq:eeF {
3298       \prop_item:cn {
3299         l_stex_symdecl_#1_prop
3300       }{ deprecate }
3301     }{}{
3302       \msg_warning:nxxx{stex}{warning/deprecated}{
3303         Symbol~#1
3304       }{
3305         \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
3306       }
3307     }
3308     \if_mode_math:
3309       \exp_after:wN \__stex_terms_invoke_math:n
3310     \else:
3311       \exp_after:wN \__stex_terms_invoke_text:n
3312     \fi: { #1 }
3313   }{
3314     \msg_error:nxxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
3315   }
3316 }
3317
3318 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
3319   \peek_charcode_remove:NTF ! {
3320     \__stex_terms_invoke_op_custom:nn {#1}
3321   }{
3322     \__stex_terms_invoke_custom:nn {#1}
3323   }
3324 }
3325
3326 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
3327   \peek_charcode_remove:NTF ! {
3328     % operator
3329     \peek_charcode_remove:NTF * {
3330       % custom op
3331       \__stex_terms_invoke_op_custom:nn {#1}
3332     }{
3333       % op notation
3334       \peek_charcode:NTF [ {
3335         \__stex_terms_invoke_op_notation:nw {#1}
3336       }{
3337         \__stex_terms_invoke_op_notation:nw {#1}[]
3338       }
3339     }
3340   }{
3341     \peek_charcode_remove:NTF * {
3342       \__stex_terms_invoke_custom:nn {#1}
3343       % custom
3344     }{
3345       % normal
3346       \peek_charcode:NTF [ {
3347         \__stex_terms_invoke_notation:nw {#1}

```

```

3348     }{
3349         \_stex_terms_invoke_notation:nw {#1}[]
3350     }
3351 }
3352 }
3353 }
3354
3355
3356 \cs_new_protected:Nn \_stex_terms_invoke_op_custom:nn {
3357     \exp_args:Nnx \use:nn {
3358         \def\comp{\_comp}
3359         \str_set:Nn \l_stex_current_symbol_str { #1 }
3360         \bool_set_false:N \l_stex_allow_semantic_bool
3361         \stex_term_oms:nnn {#1}{#1 \c_hash_str CUSTOM-}{
3362             \comp{ #2 }
3363         }
3364     }{
3365         \stex_reset:N \comp
3366         \stex_reset:N \l_stex_current_symbol_str
3367         \bool_set_true:N \l_stex_allow_semantic_bool
3368     }
3369 }
3370
3371 \keys_define:nn { stex / terms } {
3372     % lang .tl_set_x:N = \l_stex_notation_lang_str ,
3373     variant .tl_set_x:N = \l_stex_notation_variant_str ,
3374     unknown .code:n      = \str_set:Nx
3375         \l_stex_notation_variant_str \l_keys_key_str
3376 }
3377
3378 \cs_new_protected:Nn \_stex_terms_args:n {
3379     % \str_clear:N \l_stex_notation_lang_str
3380     \str_clear:N \l_stex_notation_variant_str
3381
3382     \keys_set:nn { stex / terms } { #1 }
3383 }
3384
3385 \cs_new_protected:Nn \stex_find_notation:nn {
3386     \_stex_terms_args:n { #2 }
3387     \seq_if_empty:cTF {
3388         l_stex_symdecl_ #1 _notations
3389     } {
3390         \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
3391     } {
3392         \str_if_empty:NTF \l_stex_notation_variant_str {
3393             \seq_get_left:cN {l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
3394         }{
3395             \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
3396                 \l_stex_notation_variant_str
3397             }{
3398                 % \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3399             }{
3400                 \msg_error:nnxx{stex}{error/nonotation}{#1}{
3401                     ~\l_stex_notation_variant_str

```

```

3402     }
3403   }
3404 }
3405 }
3406 }
3407
3408 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3409   \exp_args:Nnx \use:nn {
3410     \def\comp{\_comp}
3411     \str_set:Nn \l_stex_current_symbol_str { #1 }
3412     \stex_find_notation:nn { #1 }{ #2 }
3413     \bool_set_false:N \l_stex_allow_semantic_bool
3414     \cs_if_exist:cTF {
3415       stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3416     }{
3417       \_stex_term_oms:nnn { #1 }{
3418         #1 \c_hash_str \l_stex_notation_variant_str
3419       }{
3420         \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3421       }
3422     }{
3423       \int_compare:nNnTF {\prop_item:cn {\l_stex_symdecl_#1_prop}{arity}} = 0{
3424         \cs_if_exist:cTF {
3425           stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3426         }{
3427           \tl_set:Nx \stex_symbol_after_invokation_tl {
3428             \_stex_reset:N \comp
3429             \_stex_reset:N \stex_symbol_after_invokation_tl
3430             \_stex_reset:N \l_stex_current_symbol_str
3431             \bool_set_true:N \l_stex_allow_semantic_bool
3432           }
3433           \def\comp{\_comp}
3434           \str_set:Nn \l_stex_current_symbol_str { #1 }
3435           \bool_set_false:N \l_stex_allow_semantic_bool
3436           \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3437         }{
3438           \msg_error:nnxx{stex}{error/nonotation}{#1}{
3439             ~\l_stex_notation_variant_str
3440           }
3441         }
3442       }{
3443         \msg_error:nnxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3444       }
3445     }
3446   }{
3447     \_stex_reset:N \comp
3448     \_stex_reset:N \l_stex_current_symbol_str
3449     \bool_set_true:N \l_stex_allow_semantic_bool
3450   }
3451 }
3452
3453 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {
3454   \stex_find_notation:nn { #1 }{ #2 }
3455   \cs_if_exist:cTF {

```



```

3456     stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3457   }{
3458     \tl_set:Nx \stex_symbol_after_invokation_tl {
3459       \_stex_reset:N \comp
3460       \_stex_reset:N \stex_symbol_after_invokation_tl
3461       \_stex_reset:N \l_stex_current_symbol_str
3462       \bool_set_true:N \l_stex_allow_semantic_bool
3463     }
3464     \def\comp{\_comp}
3465     \str_set:Nn \l_stex_current_symbol_str { #1 }
3466     \bool_set_false:N \l_stex_allow_semantic_bool
3467     \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3468   }{
3469     \msg_error:nnxx{stex}{error/nonotation}{#1}{
3470       ~\l_stex_notation_variant_str
3471     }
3472   }
3473 }
3474
3475 \prop_new:N \l__stex_terms_custom_args_prop
3476
3477 \cs_new_protected:Nn \__stex_terms_custom_comp:n { \bool_set_false:N \l_stex_allow_semantic_bool }
3478
3479 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3480   \exp_args:Nnx \use:nn {
3481     \def\comp{\__stex_terms_custom_comp:n}
3482     \str_set:Nn \l_stex_current_symbol_str { #1 }
3483     \prop_clear:N \l__stex_terms_custom_args_prop
3484     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3485     \prop_get:cnN {
3486       l_stex_symdecl_#1 _prop
3487     } { args } \l_tmpa_str
3488     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3489     \tl_set:Nn \arg { \__stex_terms_arg: }
3490     \str_if_empty:NTF \l_tmpa_str {
3491       \_stex_term_oms:nnn {#1}{#1\c_hash_str CUSTOM-}{\ignorespaces#2}
3492     }{
3493       \str_if_in:NnTF \l_tmpa_str b {
3494         \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
3495       }{
3496         \str_if_in:NnTF \l_tmpa_str B {
3497           \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
3498         }{
3499           \_stex_term_oma:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
3500         }
3501       }
3502     }
3503     % TODO check that all arguments exist
3504   }{
3505     \_stex_reset:N \l_stex_current_symbol_str
3506     \_stex_reset:N \arg
3507     \_stex_reset:N \comp
3508     \_stex_reset:N \l__stex_terms_custom_args_prop
3509     %\bool_set_true:N \l_stex_allow_semantic_bool

```

```

3510 }
3511 }
3512
3513 \NewDocumentCommand \__stex_terms_arg: { s O{} m}{
3514   \tl_if_empty:nTF {#2}{
3515     \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
3516     \bool_set_true:N \l_tmpa_bool
3517     \bool_do_while:Nn \l_tmpa_bool {
3518       \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int
3519         \int_incr:N \l_tmpa_int
3520       }{
3521         \bool_set_false:N \l_tmpa_bool
3522       }
3523     }
3524   }{
3525     \int_set:Nn \l_tmpa_int { #2 }
3526   }
3527   \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
3528   \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3529     \msg_error:nnxxx{stex}{error/overarity}
3530     {\int_use:N \l_tmpa_int}
3531     {\l_stex_current_symbol_str}
3532     {\str_count:N \l_tmpa_str}
3533   }
3534   \str_set:Nx \l_tmpa_str {\str_item:Nn \l_tmpa_str \l_tmpa_int}
3535   \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
3536     \bool_lazy_any:nF {
3537       {\str_if_eq_p:Vn \l_tmpa_str {a}}
3538       {\str_if_eq_p:Vn \l_tmpa_str {B}}
3539     }{
3540       \msg_error:nnxx{stex}{error/doubleargument}
3541       {\int_use:N \l_tmpa_int}
3542       {\l_stex_current_symbol_str}
3543     }
3544   }
3545   \exp_args:NNx \prop_put:Nnn \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {\ign
3546   \bool_set_true:N \l_stex_allow_semantic_bool
3547   \IfBooleanTF#1{
3548     \stex_annotate_invisible:n { %TODO
3549       \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
3550     }
3551   }{ %TODO
3552     \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
3553   }
3554   \bool_set_false:N \l_stex_allow_semantic_bool
3555 }
3556
3557
3558 \cs_new_protected:Nn \_stex_term_arg:nn {
3559   \bool_set_true:N \l_stex_allow_semantic_bool
3560   \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3561   \bool_set_false:N \l_stex_allow_semantic_bool
3562 }
3563

```

```

3564 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
3565   \exp_args:Nnx \use:nn
3566     { \int_set:Nn \l__stex_terms_downprec { #2 }
3567       \_stex_term_arg:nn { #1 }{ #3 }
3568     }
3569     { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3570   }

```

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

`\_stex_term_math_assoc_arg:nnnn`

```

3571 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
3572   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3573   \tl_set:Nn \l_tmpb_tl {\_stex_term_math_arg:nnn{#1}{#2}}
3574   \tl_if_empty:nTF { #3 }{
3575     \_stex_term_math_arg:nnn{#1}{#2}{#3}
3576   }{
3577     \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 } }{
3578       \expandafter\if\expandafter\relax\noexpand#3
3579         \tl_set:Nn \l_tmpa_tl {\_stex_terms_math_assoc_arg_maybe_sequence:Nn#3{#1}}
3580       \else
3581         \tl_set:Nn \l_tmpa_tl {\_stex_terms_math_assoc_arg_simple:nn{#1}{#3}}
3582       \fi
3583       \l_tmpa_tl
3584     }{
3585       \_stex_terms_math_assoc_arg_simple:nn{#1}{#3}
3586     }
3587   }
3588 }
3589
3590 \cs_new_protected:Nn \_stex_terms_math_assoc_arg_maybe_sequence:Nn {
3591   \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
3592   \str_if_empty:NTF \l_tmpa_str {
3593     \exp_args:Nx \cs_if_eq:NNTF {
3594       \tl_head:N #1
3595     } \stex_invoke_sequence:n {
3596       \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
3597       \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
3598       \tl_set:Nx \l_tmpa_tl {\prop_item:cn {stex_varseq\_l\_tmpa\_str\_prop}{notation}}
3599       \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
3600       \tl_set:Nx \l_tmpa_tl {\exp_not:N \exp_not:n{
3601         \exp_not:n{\exp_args:Nnx \use:nn} {
3602           \exp_not:n {
3603             \def\comp{\_varcomp}
3604             \str_set:Nn \l_stex_current_symbol_str
3605               {varseq://\l_tmpa_str}
3606             \exp_not:n{ ##1 }
3607           }{
3608             \exp_not:n {
3609               \stex_reset:N \comp
3610               \stex_reset:N \l_stex_current_symbol_str
3611             }
3612           }
3613         }}}

```

```

3614 \exp_args:Nno \use:nn {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
3615 \seq_reverse:N \l_tmpa_seq
3616 \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
3617 \seq_map_inline:Nn \l_tmpa_seq {
3618   \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3619     \exp_args:Nno
3620     \l_tmpa_cs { ##1 } \l_tmpa_tl
3621   }
3622 }
3623 \tl_set:Nx \l_tmpa_tl {
3624   \stex_term_omv:nn {varseq://\l_tmpa_str}{
3625     \exp_args:No \exp_not:n \l_tmpa_tl
3626   }
3627 }
3628 \exp_args:No\l_tmpb_tl\l_tmpa_tl
3629 }{
3630   \stex_terms_math_assoc_arg_simple:nn{#2} { #1 }
3631 }
3632 } {
3633   \stex_terms_math_assoc_arg_simple:nn{#2} { #1 }
3634 }
3635 }
3636 }
3637
3638 \cs_new_protected:Nn \stex_terms_math_assoc_arg_simple:nn {
3639   \clist_set:Nn \l_tmpa_clist{ #2 }
3640   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3641     \tl_set:Nn \l_tmpa_tl { \stex_term_arg:nn{A#1}{ #2 } }
3642   }{
3643     \clist_reverse:N \l_tmpa_clist
3644     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3645     \tl_set:Nx \l_tmpa_tl { \stex_term_arg:nn{A#1}{
3646       \exp_args:No \exp_not:n \l_tmpa_tl
3647     }}
3648     \clist_map_inline:Nn \l_tmpa_clist {
3649       \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3650         \exp_args:Nno
3651         \l_tmpa_cs { \stex_term_arg:nn{A#1}{##1} } \l_tmpa_tl
3652       }
3653     }
3654   }
3655   \exp_args:No\l_tmpb_tl\l_tmpa_tl
3656 }

```

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

## 30.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
3657 \tl_const:Nx \infprec {\int_use:N \c_max_int}
3658 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}

```

```

3659 \int_new:N \l__stex_terms_downprec
3660 \int_set_eq:NN \l__stex_terms_downprec \infpref

```

(End definition for `\infpref`, `\neginfpref`, and `\l__stex_terms_downprec`. These variables are documented on page 80.)

Bracketing:

```

\l__stex_terms_left_bracket_str
\l__stex_terms_right_bracket_str

```

```

3661 \tl_set:Nn \l__stex_terms_left_bracket_str (
3662 \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

```

3663 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
3664   \bool_if:NTF \l__stex_terms_brackets_done_bool {
3665     \bool_set_false:N \l__stex_terms_brackets_done_bool
3666     #2
3667   } {
3668     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
3669       \bool_if:NTF \l__stex_inarray_bool { #2 } {
3670         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
3671         \dobrackets { #2 }
3672       }
3673     }{ #2 }
3674   }
3675 }

```

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

**\dobrackets**

```

3676 \bool_new:N \l__stex_terms_brackets_done_bool
3677 %\RequirePackage{scalerel}
3678 \cs_new_protected:Npn \dobrackets #1 {
3679   %\ThisStyle{\if D\m@switch
3680   %   \exp_args:Nnx \use:nn
3681   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
3682   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
3683   % \else
3684   \exp_args:Nnx \use:nn
3685   {
3686     \bool_set_true:N \l__stex_terms_brackets_done_bool
3687     \int_set:Nn \l__stex_terms_downprec \infpref
3688     \l__stex_terms_left_bracket_str
3689     #1
3690   }
3691   {
3692     \bool_set_false:N \l__stex_terms_brackets_done_bool
3693     \l__stex_terms_right_bracket_str
3694     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3695   }
3696   %\fi}
3697 }

```

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

**\withbrackets**

```
3698 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
3699   \exp_args:Nnx \use:nn
3700   {
3701     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
3702     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
3703     #3
3704   }
3705   {
3706     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
3707     {\l__stex_terms_left_bracket_str}
3708     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
3709     {\l__stex_terms_right_bracket_str}
3710   }
3711 }
```

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

**\STEXinvisible**

```
3712 \cs_new_protected:Npn \STEXinvisible #1 {
3713   \stex_annotate_invisible:n { #1 }
3714 }
```

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

OMDoc terms:

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

```
3715 \cs_new_protected:Nn \_stex_term_oms:nnn {
3716   \stex_annotate:nnn{ OMID }{ #2 }{
3717     #3
3718   }
3719 }
3720
3721 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
3722   \__stex_terms_maybe_brackets:nn { #3 }{
3723     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3724   }
3725 }
```

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

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

```
3726 \cs_new_protected:Nn \_stex_term_omv:nn {
3727   \stex_annotate:nnn{ OMV }{ #1 }{
3728     #2
3729   }
3730 }
```

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

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

```
3731 \cs_new_protected:Nn \_stex_term_oma:nnn {
3732   \stex_annotate:nnn{ OMA }{ #2 }{
3733     #3
3734   }
```

```

3735 }
3736
3737 \cs_new_protected:Nn \stex_term_math_oma:nnnn {
3738   \__stex_terms_maybe_brackets:nn { #3 }{
3739     \stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3740   }
3741 }

```

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

`\stex_term_math_omb:nnnn`

```

3742 \cs_new_protected:Nn \stex_term_ombind:nnn {
3743   \stex_annotate:nnn{ OMBIND }{ #2 }{
3744     #3
3745   }
3746 }
3747
3748 \cs_new_protected:Nn \stex_term_math_omb:nnnn {
3749   \__stex_terms_maybe_brackets:nn { #3 }{
3750     \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3751   }
3752 }

```

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

`\symref`

`\symname`

```

3753 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3754
3755 \keys_define:nn { stex / symname } {
3756   pre      .tl_set_x:N      = \l__stex_terms_pre_tl ,
3757   post     .tl_set_x:N      = \l__stex_terms_post_tl ,
3758   root     .tl_set_x:N      = \l__stex_terms_root_tl
3759 }
3760
3761 \cs_new_protected:Nn \stex_symname_args:n {
3762   \tl_clear:N \l__stex_terms_post_tl
3763   \tl_clear:N \l__stex_terms_pre_tl
3764   \tl_clear:N \l__stex_terms_root_str
3765   \keys_set:nn { stex / symname } { #1 }
3766 }
3767
3768 \NewDocumentCommand \symref { m m }{
3769   \let\compemph_uri_prev:\compemph@uri
3770   \let\compemph@uri\symrefemph@uri
3771   \STEXsymbol{#1}!\{ #2 }
3772   \let\compemph@uri\compemph_uri_prev:
3773 }
3774
3775 \NewDocumentCommand \synonym { 0{ } m m }{
3776   \stex_symname_args:n { #1 }
3777   \let\compemph_uri_prev:\compemph@uri
3778   \let\compemph@uri\symrefemph@uri
3779   % TODO
3780   \STEXsymbol{#2}!\{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
3781   \let\compemph@uri\compemph_uri_prev:

```

```

3782 }
3783
3784 \NewDocumentCommand \symname { 0{} m }{
3785   \stex_symname_args:n { #1 }
3786   \stex_get_symbol:n { #2 }
3787   \str_set:Nx \l_tmpa_str {
3788     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3789   }
3790   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3791
3792   \let\compemph_uri_prev:\compemph@uri
3793   \let\compemph@uri\symrefemph@uri
3794   \exp_args:NNx \use:nn
3795   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmmode*\fi{
3796     \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
3797   } }
3798   \let\compemph@uri\compemph_uri_prev:
3799 }
3800
3801 \NewDocumentCommand \Symname { 0{} m }{
3802   \stex_symname_args:n { #1 }
3803   \stex_get_symbol:n { #2 }
3804   \str_set:Nx \l_tmpa_str {
3805     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3806   }
3807   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3808   \let\compemph_uri_prev:\compemph@uri
3809   \let\compemph@uri\symrefemph@uri
3810   \exp_args:NNx \use:nn
3811   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmmode*\fi{
3812     \exp_after:wN \stex_capitalize:n \l_tmpa_str
3813     \l__stex_terms_post_tl
3814   } }
3815   \let\compemph@uri\compemph_uri_prev:
3816 }

```

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

### 30.3 Notation Components

```

3817 <@@=stex_notationcomps>
3818 \comp
3819 \compemph@uri
3820 \compemph
3821 \defemph
3822 \defemph@uri
3823 \symrefemph
3824 \symrefemph@uri
3825 \varemp
3826 \varemp@uri
3827
3828 \cs_new_protected:Npn \_comp #1 {
3829   \str_if_empty:NF \l_stex_current_symbol_str {
3830     \stex_html_backend:TF {
3831       \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
3832     }{
3833       \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3834     }
3835   }
3836 }
3837
3838 \cs_new_protected:Npn \_varcomp #1 {

```



```

3829 \str_if_empty:NF \l_stex_current_symbol_str {
3830   \stex_html_backend:TF {
3831     \stex_annotate:nnn { varcomp }{ \l_stex_current_symbol_str }{ #1 }
3832   }{
3833     \exp_args:Nnx \varemp@uri { #1 } { \l_stex_current_symbol_str }
3834   }
3835 }
3836 }
3837
3838 \def\comp{\_comp}
3839
3840 \cs_new_protected:Npn \compemph@uri #1 #2 {
3841   \compemph{ #1 }
3842 }
3843
3844
3845 \cs_new_protected:Npn \compemph #1 {
3846   #1
3847 }
3848
3849 \cs_new_protected:Npn \defemph@uri #1 #2 {
3850   \defemph{#1}
3851 }
3852
3853 \cs_new_protected:Npn \defemph #1 {
3854   \textbf{#1}
3855 }
3856
3857 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3858   \symrefemph{#1}
3859 }
3860
3861 \cs_new_protected:Npn \symrefemph #1 {
3862   \emph{#1}
3863 }
3864
3865 \cs_new_protected:Npn \varemp@uri #1 #2 {
3866   \varemp{#1}
3867 }
3868
3869 \cs_new_protected:Npn \varemp #1 {
3870   #1
3871 }

```

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

## **\ellipses**

```

3872 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix 3873 \bool_new:N \l_stex_inparray_bool
\parrayline 3874 \bool_set_false:N \l_stex_inparray_bool
\parraylineh 3875 \NewDocumentCommand \parray { m m } {
\parraycell

```

```

3876 \begingroup
3877 \bool_set_true:N \l_stex_inarray_bool
3878 \begin{array}{#1}
3879 #2
3880 \end{array}
3881 \endgroup
3882 }
3883
3884 \NewDocumentCommand \prmatrix { m } {
3885 \begingroup
3886 \bool_set_true:N \l_stex_inarray_bool
3887 \begin{matrix}
3888 #1
3889 \end{matrix}
3890 \endgroup
3891 }
3892
3893 \def \maybepline {
3894 \bool_if:NT \l_stex_inarray_bool {\hline}
3895 }
3896
3897 \def \parrayline #1 #2 {
3898 #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3899 }
3900
3901 \def \pmrow #1 { \parrayline{}{ #1 } }
3902
3903 \def \parraylineh #1 #2 {
3904 #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3905 }
3906
3907 \def \parraycell #1 {
3908 #1 \bool_if:NT \l_stex_inarray_bool {&}
3909 }

```

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

## 30.4 Variables

```

3910 <@@=stex_variables>

```

`\stex_invoke_variable:n` Invokes a variable

```

3911 \cs_new_protected:Nn \stex_invoke_variable:n {
3912 \if_mode_math:
3913 \exp_after:wN \__stex_variables_invoke_math:n
3914 \else:
3915 \exp_after:wN \__stex_variables_invoke_text:n
3916 \fi: {#1}
3917 }
3918
3919 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
3920 \peek_charcode_remove:NTF ! {
3921 \__stex_variables_invoke_op_custom:nn {#1}
3922 }{

```

```

3923     \__stex_variables_invoke_custom:nn {#1}
3924   }
3925 }
3926
3927
3928 \cs_new_protected:Nn \__stex_variables_invoke_math:n {
3929   \peek_charcode_remove:NTF ! {
3930     \peek_charcode_remove:NTF ! {
3931       \peek_charcode:NTF [ {
3932         % TODO throw error
3933       }{
3934         \__stex_variables_invoke_op_custom:nn
3935       }
3936     }{
3937       \__stex_variables_invoke_op:n { #1 }
3938     }
3939   }{
3940     \peek_charcode_remove:NTF * {
3941       \__stex_variables_invoke_custom:nn { #1 }
3942     }{
3943       \__stex_variables_invoke_math_ii:n { #1 }
3944     }
3945   }
3946 }
3947
3948 \cs_new_protected:Nn \__stex_variables_invoke_op_custom:nn {
3949   \exp_args:Nnx \use:nn {
3950     \def\comp{\_varcomp}
3951     \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3952     \bool_set_false:N \l_stex_allow_semantic_bool
3953     \stex_term_omv:nn {var://#1}{
3954       \comp{ #2 }
3955     }
3956   }{
3957     \stex_reset:N \comp
3958     \stex_reset:N \l_stex_current_symbol_str
3959     \bool_set_true:N \l_stex_allow_semantic_bool
3960   }
3961 }
3962
3963 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
3964   \cs_if_exist:cTF {
3965     stex_var_op_notation_ #1 _cs
3966   }{
3967     \exp_args:Nnx \use:nn {
3968       \def\comp{\_varcomp}
3969       \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3970       \stex_term_omv:nn { var://#1 }{
3971         \use:c{stex_var_op_notation_ #1 _cs }
3972       }
3973     }{
3974       \stex_reset:N \comp
3975       \stex_reset:N \l_stex_current_symbol_str
3976     }

```

```

3977 }{
3978   \int_compare:nNnTF {\prop_item:cn {l_stex_variable_#1_prop}{arity}} = 0{
3979     \__stex_variables_invoke_math_ii:n {#1}
3980   }{
3981     \msg_error:nnxx{stex}{error/noop}{variable~#1}{}
3982   }
3983 }
3984 }
3985
3986 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n #1 {
3987   \cs_if_exist:cTF {
3988     stex_var_notation_#1_cs
3989   }{
3990     \tl_set:Nx \stex_symbol_after_invokation_tl {
3991       \_stex_reset:N \comp
3992       \_stex_reset:N \stex_symbol_after_invokation_tl
3993       \_stex_reset:N \l_stex_current_symbol_str
3994       \bool_set_true:N \l_stex_allow_semantic_bool
3995     }
3996     \def\comp{\_varcomp}
3997     \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3998     \bool_set_false:N \l_stex_allow_semantic_bool
3999     \use:c{stex_var_notation_#1_cs}
4000   }{
4001     \msg_error:nnxx{stex}{error/nonotation}{variable~#1}{s}
4002   }
4003 }
4004
4005 \cs_new_protected:Nn \__stex_variables_invoke_custom:nn {
4006   \exp_args:Nnx \use:nn {
4007     \def\comp{\_varcomp}
4008     \str_set:Nn \l_stex_current_symbol_str { var://#1 }
4009     \prop_clear:N \l__stex_terms_custom_args_prop
4010     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
4011     \prop_get:cnN {
4012       l_stex_variable_#1_prop
4013     }{ args } \l_tmpa_str
4014     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
4015     \tl_set:Nn \arg { \__stex_terms_arg: }
4016     \str_if_empty:NTF \l_tmpa_str {
4017       \_stex_term_omv:nn {var://#1}{\ignorespaces#2}
4018     }{
4019       \str_if_in:NnTF \l_tmpa_str b {
4020         \_stex_term_ombind:nnn {var://#1}{\ignorespaces#2}
4021       }{
4022         \str_if_in:NnTF \l_tmpa_str B {
4023           \_stex_term_ombind:nnn {var://#1}{\ignorespaces#2}
4024         }{
4025           \_stex_term_oma:nnn {var://#1}{\ignorespaces#2}
4026         }
4027       }
4028     }
4029     % TODO check that all arguments exist
4030   }{

```

```

4031 \stex_reset:N \l_stex_current_symbol_str
4032 \stex_reset:N \arg
4033 \stex_reset:N \comp
4034 \stex_reset:N \l__stex_terms_custom_args_prop
4035 %\bool_set_true:N \l_stex_allow_semantic_bool
4036 }
4037 }

```

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

## 30.5 Sequences

```

4038 <@@=stex_sequences>
4039
4040 \cs_new_protected:Nn \stex_invoke_sequence:n {
4041 \peek_charcode_remove:NTF ! {
4042 \stex_term_omv:nn {varseq://#1}{
4043 \exp_args:Nnx \use:nn {
4044 \def\comp{\_varcomp}
4045 \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
4046 \prop_item:cn{stex_varseq_#1_prop}{notation}
4047 }{
4048 \stex_reset:N \comp
4049 \stex_reset:N \l_stex_current_symbol_str
4050 }
4051 }
4052 }{
4053 \bool_set_false:N \l_stex_allow_semantic_bool
4054 \def\comp{\_varcomp}
4055 \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
4056 \tl_set:Nx \stex_symbol_after_invokation_tl {
4057 \stex_reset:N \comp
4058 \stex_reset:N \stex_symbol_after_invokation_tl
4059 \stex_reset:N \l_stex_current_symbol_str
4060 \bool_set_true:N \l_stex_allow_semantic_bool
4061 }
4062 \use:c { stex_varseq_#1_cs }
4063 }
4064 }
4065 </package>

```

## Chapter 31

# STEX -Structural Features Implementation

```
4066 <*package>
4067
4068 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4069
    Warnings and error messages
4070 \msg_new:nnn{stex}{error/copymodule/notallowed}{
4071   Symbol~#1~can~not~be~assigned~in~copymodule~#2
4072 }
4073 \msg_new:nnn{stex}{error/interpretmodule/nodfiniens}{
4074   Symbol~#1~not~assigned~in~interpretmodule~#2
4075 }
4076
4077 \msg_new:nnn{stex}{error/unknownstructure}{
4078   No~structure~#1~found!
4079 }
4080
4081 \msg_new:nnn{stex}{error/unknownfield}{
4082   No~field~#1~in~instance~#2~found!\#3
4083 }
4084
4085 \msg_new:nnn{stex}{error/keyval}{
4086   Invalid~key=value~pair:#1
4087 }
4088 \msg_new:nnn{stex}{error/instantiate/missing}{
4089   Assignments~missing~in~instantiate:~#1
4090 }
4091 \msg_new:nnn{stex}{error/incompatible}{
4092   Incompatible~signature:~#1~(#2)~and~#3~(#4)
4093 }
4094
```

## 31.1 Imports with modification

```

4095 <@@=stex_copymodule>
4096 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
4097   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4098     \tl_set:Nn \l_tmpa_tl { #1 }
4099     \__stex_copymodule_get_symbol_from_cs:
4100   }{
4101     % argument is a string
4102     % is it a command name?
4103     \cs_if_exist:cTF { #1 }{
4104       \cs_set_eq:Nc \l_tmpa_tl { #1 }
4105       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4106       \str_if_empty:NNTF \l_tmpa_str {
4107         \exp_args:Nx \cs_if_eq:NNTF {
4108           \tl_head:N \l_tmpa_tl
4109         } \stex_invoke_symbol:n {
4110           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
4111         }{
4112           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4113         }
4114       } {
4115         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4116       }
4117     }{
4118       % argument is not a command name
4119       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4120       % \l_stex_all_symbols_seq
4121     }
4122   }
4123 }
4124
4125 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
4126   \str_set:Nn \l_tmpa_str { #1 }
4127   \bool_set_false:N \l_tmpa_bool
4128   \bool_if:NF \l_tmpa_bool {
4129     \tl_set:Nn \l_tmpa_tl {
4130       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
4131     }
4132     \str_set:Nn \l_tmpa_str { #1 }
4133     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4134     \seq_map_inline:Nn #2 {
4135       \str_set:Nn \l_tmpb_str { ##1 }
4136       \str_if_eq:eeT { \l_tmpa_str } {
4137         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4138       } {
4139         \seq_map_break:n {
4140           \tl_set:Nn \l_tmpa_tl {
4141             \str_set:Nn \l_stex_get_symbol_uri_str {
4142               ##1
4143             }
4144           }
4145         }
4146       }

```

```

4147     }
4148     \l_tmpa_tl
4149   }
4150 }
4151
4152 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
4153   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4154     { \tl_tail:N \l_tmpa_tl }
4155   \tl_if_single:NTF \l_tmpa_tl {
4156     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
4157       \exp_after:wN \str_set:Nn \exp_after:wN
4158         \l_stex_get_symbol_uri_str \l_tmpa_tl
4159       \__stex_copymodule_get_symbol_check:n { #1 }
4160     }{
4161       % TODO
4162       % tail is not a single group
4163     }
4164   }{
4165     % TODO
4166     % tail is not a single group
4167   }
4168 }
4169
4170 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
4171   \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
4172     \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
4173       :~\seq_use:Nn #1 {,~}
4174     }
4175   }
4176 }
4177
4178 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
4179   % import module
4180   \stex_import_module_uri:nn { #1 } { #2 }
4181   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
4182   \stex_import_require_module:nnnn
4183     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4184     { \l_stex_import_path_str } { \l_stex_import_name_str }
4185
4186   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
4187   \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
4188
4189   % fields
4190   \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
4191   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4192     \seq_map_inline:cn {c_stex_module_##1_constants}{
4193       \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
4194         ##1 ? ####1
4195       }
4196     }
4197   }
4198
4199   % setup prop
4200   \seq_clear:N \l_tmpa_seq

```



```

4201 \exp_args:Nn \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
4202   name      = \l_stex_current_copymodule_name_str ,
4203   module    = \l_stex_current_module_str ,
4204   from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
4205   includes  = \l_tmpa_seq %,
4206 % fields    = \l_tmpa_seq
4207 }
4208 \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
4209   as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
4210 \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_seq {,
4211 \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
4212
4213 \stex_if_do_html:T {
4214   \begin{stex_annotate_env} {#4} {
4215     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4216   }
4217   \stex_annotate_invisible:nnn{domain}{\l_stex_import_ns_str ?\l_stex_import_name_str}{}
4218 }
4219 }
4220
4221 \cs_new_protected:Nn \stex_copymodule_end:n {
4222   % apply to every field
4223   \def \l_tmpa_cs ##1 ##2 {#1}
4224
4225   \tl_clear:N \__stex_copymodule_module_tl
4226   \tl_clear:N \__stex_copymodule_exec_tl
4227
4228   %\prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
4229   \seq_clear:N \__stex_copymodule_fields_seq
4230
4231   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4232     \seq_map_inline:cn {c_stex_module_##1_constants}{
4233
4234       \tl_clear:N \__stex_copymodule_curr_symbol_tl % <- wrap in current symbol html
4235       \l_tmpa_cs{##1}{####1}
4236
4237       \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
4238         \str_set_eq:Nc \__stex_copymodule_curr_name_str {l__stex_copymodule_copymodule_##1?####1_name_str}
4239         \stex_if_do_html:T {
4240           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4241             \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?####1_name_str}}
4242           }
4243         }
4244       }{
4245         \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str /
4246       }
4247
4248       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
4249       \prop_put:Nnx \l_tmpa_prop { name } \__stex_copymodule_curr_name_str
4250       \prop_put:Nnx \l_tmpa_prop { module } \l_stex_current_module_str
4251
4252       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4253         \stex_if_do_html:T {
4254           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {

```

```

4255     $\stex_annotate_invisible:nnn{definiens}{\exp_after:wN \exp_not:N\csname l__st
4256   }
4257 }
4258 \prop_put:Nnn \l_tmpa_prop { defined } { true }
4259 }
4260
4261 \stex_add_constant_to_current_module:n \__stex_copymodule_curr_name_str
4262 \tl_put_right:Nx \__stex_copymodule_module_tl {
4263   \seq_clear:c {l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_n
4264   \prop_set_from_keyval:cn {
4265     l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_name_str _prop
4266   }{
4267     \prop_to_keyval:N \l_tmpa_prop
4268   }
4269 }
4270
4271 \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
4272   \stex_if_do_html:T {
4273     \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4274       \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1
4275     }
4276   }
4277   \tl_put_right:Nx \__stex_copymodule_module_tl {
4278     \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
4279       \stex_invoke_symbol:n {
4280         \l_stex_current_module_str ? \__stex_copymodule_curr_name_str
4281       }
4282     }
4283   }
4284 }
4285
4286 \seq_put_right:Nx \__stex_copymodule_fields_seq {\l_stex_current_module_str ? \__stex_
4287
4288 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4289   \stex_copy_notations:nn {\l_stex_current_module_str ? \__stex_copymodule_curr_name_s
4290 }
4291
4292 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4293   \stex_if_do_html:TF{
4294     \stex_annotate_invisible:nnn{assignment} {##1?####1} { \exp_after:wN \exp_not:n \e
4295   }{
4296     \exp_after:wN \exp_not:n \exp_after:wN {\__stex_copymodule_curr_symbol_tl}
4297   }
4298 }
4299 }
4300 }
4301
4302
4303 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \__stex_copymodule_fields_seq
4304 \tl_put_left:Nx \__stex_copymodule_module_tl {
4305   \prop_set_from_keyval:cn {
4306     l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _pro
4307 }{
4308   \prop_to_keyval:N \l_stex_current_copymodule_prop

```

```

4309     }
4310   }
4311
4312   \seq_gput_right:cx{c_stex_module_\l_stex_current_module_str _copymodules}{
4313     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4314   }
4315
4316   \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4317   \stex_debug:nn{copymodule}{result:\meaning \__stex_copymodule_module_tl}
4318   \stex_debug:nn{copymodule}{output:\meaning \__stex_copymodule_exec_tl}
4319
4320   \__stex_copymodule_exec_tl
4321   \stex_if_do_html:T {
4322     \end{stex_annotate_env}
4323   }
4324 }
4325
4326 \NewDocumentEnvironment {copymodule} { 0{} m m}{
4327   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ copymodule }
4328   \stex_deactivate_macro:Nn \symdecl {module~environments}
4329   \stex_deactivate_macro:Nn \symdef {module~environments}
4330   \stex_deactivate_macro:Nn \notation {module~environments}
4331   \stex_reactivate_macro:N \assign
4332   \stex_reactivate_macro:N \renamedekl
4333   \stex_reactivate_macro:N \donotcopy
4334   \stex_smsmode_do:
4335 }{
4336   \stex_copymodule_end:n {}
4337 }
4338
4339 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
4340   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ interpretmodule }
4341   \stex_deactivate_macro:Nn \symdecl {module~environments}
4342   \stex_deactivate_macro:Nn \symdef {module~environments}
4343   \stex_deactivate_macro:Nn \notation {module~environments}
4344   \stex_reactivate_macro:N \assign
4345   \stex_reactivate_macro:N \renamedekl
4346   \stex_reactivate_macro:N \donotcopy
4347   \stex_smsmode_do:
4348 }{
4349   \stex_copymodule_end:n {
4350     \tl_if_exist:cF {
4351       l__stex_copymodule_copymodule_##1?##2_def_tl
4352     }{
4353       \str_if_eq:eeF {
4354         \prop_item:cn{
4355           l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
4356         }{ true }{
4357           \msg_error:nxxx{stex}{error/interpretmodule/nodéfiniens}{
4358             ##1?##2
4359           }{\l_stex_current_copymodule_name_str}
4360         }
4361       }
4362     }

```

```

4363 }
4364
4365 \iffalse \begin{stex_annotate_env} \fi
4366 \NewDocumentEnvironment {realization} { 0 } { m } {
4367   \stex_copymodule_start:nnnn { #1 } { #2 } { #2 } { realize }
4368   \stex_deactivate_macro:Nn \symdecl {module~environments}
4369   \stex_deactivate_macro:Nn \symdef {module~environments}
4370   \stex_deactivate_macro:Nn \notation {module~environments}
4371   \stex_reactivate_macro:N \donotcopy
4372   \stex_reactivate_macro:N \assign
4373   \stex_smsmode_do:
4374 } {
4375   \stex_import_module_uri:nn { #1 } { #2 }
4376   \tl_clear:N \__stex_copymodule_exec_tl
4377   \tl_set:Nx \__stex_copymodule_module_tl {
4378     \stex_import_require_module:nnnn
4379     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4380     { \l_stex_import_path_str } { \l_stex_import_name_str }
4381   }
4382
4383   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4384     \seq_map_inline:cn {c_stex_module_##1_constants}{
4385       \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str / #1 }
4386       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?###1_def_tl}{
4387         \stex_if_do_html:T {
4388           \tl_put_right:Nx \__stex_copymodule_exec_tl {
4389             \stex_annotate_invisible:nnn{assignment} {##1?###1} {
4390               $\stex_annotate_invisible:nnn{definien}{}{\exp_after:wN \exp_not:N\csname l__
4391             }
4392           }
4393         }
4394         \tl_put_right:Nx \__stex_copymodule_module_tl {
4395           \prop_put:cnn {l_stex_symdecl_##1?###1_prop}{ defined }{ true }
4396         }
4397       }
4398     }
4399
4400     \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4401
4402     \__stex_copymodule_exec_tl
4403     \stex_if_do_html:T {\end{stex_annotate_env}}
4404   }
4405
4406   \NewDocumentCommand \donotcopy { m } {
4407     \str_clear:N \l_stex_import_name_str
4408     \str_set:Nn \l_tmpa_str { #1 }
4409     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4410     \seq_map_inline:Nn \l_stex_all_modules_seq {
4411       \str_set:Nn \l_tmpb_str { ##1 }
4412       \str_if_eq:eeT { \l_tmpa_str } {
4413         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4414       } {
4415         \seq_map_break:n {
4416           \stex_if_do_html:T {

```

```

4417         \stex_if_smsmode:F {
4418             \stex_annotate_invisible:nnn{donotcopy}{##1}{
4419                 \stex_annotate:nnn{domain}{##1}{}}
4420         }
4421     }
4422 }
4423 \str_set_eq:NN \l_stex_import_name_str \l_tmpb_str
4424 }
4425 }
4426 \seq_map_inline:cn {c_stex_module_###1_copymodules}{
4427     \str_set:Nn \l_tmpb_str { #####1 }
4428     \str_if_eq:eeT { \l_tmpa_str } {
4429         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4430     } {
4431         \seq_map_break:n {\seq_map_break:n {
4432             \stex_if_do_html:T {
4433                 \stex_if_smsmode:F {
4434                     \stex_annotate_invisible:nnn{donotcopy}{#####1}{
4435                         \stex_annotate:nnn{domain}{
4436                             \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4437                         }{}
4438                     }
4439                 }
4440             }
4441             \str_set:Nx \l_stex_import_name_str {
4442                 \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4443             }
4444         }}
4445     }
4446 }
4447 }
4448 \str_if_empty:NTF \l_stex_import_name_str {
4449     % TODO throw error
4450 }{
4451     \stex_collect_imports:n {\l_stex_import_name_str }
4452     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4453         \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
4454         \seq_map_inline:cn {c_stex_module_###1_constants}{
4455             \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? #####1 }
4456             \bool_lazy_any:nT {
4457                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_name_str}}
4458                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_macroname_str}}
4459                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_def_tl}}
4460             }{
4461                 % TODO throw error
4462             }
4463         }
4464     }
4465     \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
4466     \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_name_str }
4467     \prop_put:Nno \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
4468 }
4469 \stex_smsmode_do:
4470 }

```

```

4471
4472 \NewDocumentCommand \assign { m m }{
4473   \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
4474   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
4475   \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_def_tl}{#2}
4476   \stex_smsmode_do:
4477 }
4478
4479 \keys_define:nn { stex / renamedekl } {
4480   name          .str_set_x:N = \l_stex_renamedekl_name_str
4481 }
4482 \cs_new_protected:Nn \__stex_copymodule_renamedekl_args:n {
4483   \str_clear:N \l_stex_renamedekl_name_str
4484   \keys_set:nn { stex / renamedekl } { #1 }
4485 }
4486
4487 \NewDocumentCommand \renamedekl { O{} m m }{
4488   \__stex_copymodule_renamedekl_args:n { #1 }
4489   \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
4490   \stex_debug:nn{renamedekl}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
4491   \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_macroname_str}{#3}
4492   \str_if_empty:NTF \l_stex_renamedekl_name_str {
4493     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4494       \l_stex_get_symbol_uri_str
4495     } }
4496   } {
4497     \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_name_str}{\l_stex
4498       \stex_debug:nn{renamedekl}{@~\l_stex_current_module_str ? \l_stex_renamedekl_name_str}
4499       \prop_set_eq:cc {l_stex_symdecl_
4500         \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4501         _prop
4502       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
4503       \seq_set_eq:cc {l_stex_symdecl_
4504         \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4505         _notations
4506       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
4507       \prop_put:cnx {l_stex_symdecl_
4508         \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4509         _prop
4510       }{ name }{ \l_stex_renamedekl_name_str }
4511       \prop_put:cnx {l_stex_symdecl_
4512         \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4513         _prop
4514       }{ module }{ \l_stex_current_module_str }
4515       \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
4516         \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4517       }
4518       \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4519         \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4520       } }
4521     }
4522   \stex_smsmode_do:
4523 }
4524

```

```

4525 \stex_deactivate_macro:Nn \assign {copymodules}
4526 \stex_deactivate_macro:Nn \renamedekl {copymodules}
4527 \stex_deactivate_macro:Nn \donotcopy {copymodules}
4528
4529

```

## 31.2 The feature environment

structural@feature

```

4530 <@@=stex_features>
4531
4532 \NewDocumentEnvironment{structural_feature_module}{ m m m }{
4533   \stex_if_in_module:F {
4534     \msg_set:nnn{stex}{error/nomodule}{
4535       Structural~Feature~has~to~occur~in~a~module:\\
4536       Feature~#2~of~type~#1\\
4537       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
4538     }
4539     \msg_error:nn{stex}{error/nomodule}
4540   }
4541
4542   \str_set_eq:NN \l_stex_feature_parent_str \l_stex_current_module_str
4543
4544   \stex_module_setup:nn{meta=NONE}{#2 - #1}
4545
4546   \stex_if_do_html:T {
4547     \begin{stex_annotate_env}{ feature:#1 }{\l_stex_feature_parent_str ? #2 - #1}
4548     \stex_annotate_invisible:nnn{header}{\{ #3 }
4549   }
4550 }{
4551   \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
4552   \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
4553   \stex_debug:nn{features}{
4554     Feature: \l_stex_last_feature_str
4555   }
4556   \stex_if_do_html:T {
4557     \end{stex_annotate_env}
4558   }
4559 }

```

## 31.3 Structure

structure

```

4560 <@@=stex_structures>
4561 \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
4562   \prop_if_exist:cF {c_stex_module_ \l_stex_current_module_str _structures}{
4563     \prop_new:c {c_stex_module_ \l_stex_current_module_str _structures}
4564   }
4565   \prop_gput:cxn{c_stex_module_ \l_stex_current_module_str _structures}
4566   {#1}{#2}
4567 }
4568

```

```

4569 \keys_define:nn { stex / features / structure } {
4570   name          .str_set_x:N = \l__stex_structures_name_str ,
4571 }
4572
4573 \cs_new_protected:Nn \__stex_structures_structure_args:n {
4574   \str_clear:N \l__stex_structures_name_str
4575   \keys_set:nn { stex / features / structure } { #1 }
4576 }
4577
4578 \NewDocumentEnvironment{mathstructure}{m O{}}{
4579   \__stex_structures_structure_args:n { #2 }
4580   \str_if_empty:NT \l__stex_structures_name_str {
4581     \str_set:Nx \l__stex_structures_name_str { #1 }
4582   }
4583   \stex_suppress_html:n {
4584     \bool_set_true:N \l_stex_symdecl_make_macro_bool
4585     \exp_args:Nx \stex_symdecl_do:nn {
4586       name = \l__stex_structures_name_str ,
4587       def = {\STEXsymbol{module-type}}{
4588         \stex_term_math_oms:nnnn {
4589           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4590             { ns } ?
4591           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4592             { name } / \l__stex_structures_name_str - structure
4593         }{}{0}{}
4594       }}
4595     }{ #1 }
4596   }
4597   \exp_args:Nnnx
4598   \begin{structural_feature_module}{ structure }
4599     { \l__stex_structures_name_str }{}
4600   \stex_smsmode_do:
4601 }{
4602   \end{structural_feature_module}
4603   \stex_reset_up_to_module:n \l_stex_last_feature_str
4604   \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
4605   \seq_clear:N \l_tmpa_seq
4606   \seq_map_inline:Nn \l_stex_collect_imports_seq {
4607     \seq_map_inline:cn{c_stex_module_##1_constants}{
4608       \seq_put_right:Nn \l_tmpa_seq { ##1 ? ####1 }
4609     }
4610   }
4611   \exp_args:Nnno
4612   \prop_gput:cnn {c_stex_module_ \l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
4613   \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
4614   \stex_add_structure_to_current_module:nn
4615     \l__stex_structures_name_str
4616     \l_stex_last_feature_str
4617
4618   \stex_execute_in_module:x {
4619     \tl_set:cn { #1 }{
4620       \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures_name_str }
4621     }
4622   }

```



```

4623 }
4624
4625 \cs_new:Nn \stex_invoke_structure:nn {
4626   \stex_invoke_symbol:n { #1?#2 }
4627 }
4628
4629 \cs_new_protected:Nn \stex_get_structure:n {
4630   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4631     \tl_set:Nn \l_tmpa_tl { #1 }
4632     \__stex_structures_get_from_cs:
4633   }{
4634     \cs_if_exist:cTF { #1 }{
4635       \cs_set_eq:Nc \l_tmpa_cs { #1 }
4636       \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
4637       \str_if_empty:NTF \l_tmpa_str {
4638         \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs} \stex_invoke_structure:nn {
4639           \__stex_structures_get_from_cs:
4640         }{
4641           \__stex_structures_get_from_string:n { #1 }
4642         }
4643       }{
4644         \__stex_structures_get_from_string:n { #1 }
4645       }
4646     }{
4647       \__stex_structures_get_from_string:n { #1 }
4648     }
4649   }
4650 }
4651
4652 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
4653   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4654     { \tl_tail:N \l_tmpa_tl }
4655   \str_set:Nx \l_tmpa_str {
4656     \exp_after:wN \use_i:nn \l_tmpa_tl
4657   }
4658   \str_set:Nx \l_tmpb_str {
4659     \exp_after:wN \use_ii:nn \l_tmpa_tl
4660   }
4661   \str_set:Nx \l_stex_get_structure_str {
4662     \l_tmpa_str ? \l_tmpb_str
4663   }
4664   \str_set:Nx \l_stex_get_structure_module_str {
4665     \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
4666   }
4667 }
4668
4669 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
4670   \tl_set:Nn \l_tmpa_tl {
4671     \msg_error:nnn{stex}{error/unknownstructure}{#1}
4672   }
4673   \str_set:Nn \l_tmpa_str { #1 }
4674   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4675
4676   \seq_map_inline:Nn \l_stex_all_modules_seq {

```

```

4677 \prop_if_exist:cT {c_stex_module_##1_structures} {
4678 \prop_map_inline:cn {c_stex_module_##1_structures} {
4679 \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
4680 \prop_map_break:n{\seq_map_break:n{
4681 \tl_set:Nn \l_tmpa_tl {
4682 \str_set:Nn \l_stex_get_structure_str {##1?####1}
4683 \str_set:Nn \l_stex_get_structure_module_str {####2}
4684 }
4685 }}
4686 }
4687 }
4688 }
4689 }
4690 \l_tmpa_tl
4691 }

```

**\instantiate**

```

4692
4693 \keys_define:nn { stex / instantiate } {
4694   name          .str_set_x:N = \l__stex_structures_name_str
4695 }
4696 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
4697   \str_clear:N \l__stex_structures_name_str
4698   \keys_set:nn { stex / instantiate } { #1 }
4699 }
4700
4701 \NewDocumentCommand \instantiate {m O{} m m O{}}{
4702   \beginingroup
4703     \stex_get_structure:n {#3}
4704     \__stex_structures_instantiate_args:n { #2 }
4705     \str_if_empty:NT \l__stex_structures_name_str {
4706       \str_set:Nn \l__stex_structures_name_str { #1 }
4707     }
4708     \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4709     \seq_clear:N \l__stex_structures_fields_seq
4710     \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4711     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4712       \seq_map_inline:cn {c_stex_module_##1_constants}{
4713         \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4714       }
4715     }
4716
4717     \tl_if_empty:nF{#5}{
4718       \seq_set_split:Nnn \l_tmpa_seq , {#5}
4719       \prop_clear:N \l_tmpa_prop
4720       \seq_map_inline:Nn \l_tmpa_seq {
4721         \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4722         \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4723           \msg_error:nnn{stex}{error/keyval}{##1}
4724         }
4725         \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
4726         \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4727         \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
4728         \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}

```

```

4729     \exp_args:Nxx \str_if_eq:nnF
4730     {\prop_item:cn{l_stex_symdecl\_l__stex_structures_dom_str _prop}{args}}
4731     {\prop_item:cn{l_stex_symdecl\_l_stex_get_symbol_uri_str _prop}{args}}{
4732     \msg_error:nnxxxx{stex}{error/incompatible}
4733     {l__stex_structures_dom_str}
4734     {\prop_item:cn{l_stex_symdecl\_l__stex_structures_dom_str _prop}{args}}
4735     {\l_stex_get_symbol_uri_str}
4736     {\prop_item:cn{l_stex_symdecl\_l_stex_get_symbol_uri_str _prop}{args}}
4737   }
4738   \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
4739 }
4740 }
4741
4742 \seq_map_inline:Nn \l__stex_structures_fields_seq {
4743   \str_set:Nx \l_tmpa_str {field:l__stex_structures_name_str . \prop_item:cn {l_stex_sy
4744   \stex_debug:nn{instantiate}{Field~\l_tmpa_str :~##1}
4745
4746   \stex_add_constant_to_current_module:n {\l_tmpa_str}
4747   \stex_execute_in_module:x {
4748     \prop_set_from_keyval:cn { l_stex_symdecl_ \l_stex_current_module_str?\l_tmpa_str _p
4749     name   = \l_tmpa_str ,
4750     args   = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4751     arity  = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4752     assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4753   }
4754   \seq_clear:c {l_stex_symdecl\_l_stex_current_module_str?\l_tmpa_str _notations}
4755 }
4756
4757 \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4758   \stex_find_notation:nn{##1}{}
4759   \stex_execute_in_module:x {
4760     \seq_put_right:cn {l_stex_symdecl\_l_stex_current_module_str?\l_tmpa_str _notation
4761   }
4762
4763   \stex_copy_control_sequence_ii:ccN
4764   {stex_notation\_l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_notation_
4765   {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4766   \l_tmpa_tl
4767   \exp_args:No \stex_execute_in_module:n \l_tmpa_tl
4768
4769
4770   \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
4771     \tl_set_eq:Nc \l_tmpa_cs {stex_op_notation_##1\c_hash_str \l_stex_notation_variant
4772     \stex_execute_in_module:x {
4773       \tl_set:cn
4774       {stex_op_notation\_l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_not
4775       { \exp_args:No \exp_not:n \l_tmpa_cs}
4776     }
4777   }
4778
4779 }
4780
4781 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\l_stex_cur
4782 }

```

```

4783
4784 \stex_execute_in_module:x {
4785   \prop_set_from_keyval:cn {l_stex_instance\l_stex_current_module_str?\l__stex_structur
4786     domain = \l_stex_get_structure_module_str ,
4787     \prop_to_keyval:N \l_tmpa_prop
4788   }
4789   \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
4790 }
4791 \stex_debug:nn{instantiate}{
4792   Instance~\l_stex_current_module_str?\l__stex_structures_name_str \
4793   \prop_to_keyval:N \l_tmpa_prop
4794 }
4795 \exp_args:Nxx \stex_symdecl_do:nn {
4796   type={\STEXsymbol{module-type}}{
4797     \stex_term_math_oms:nnnn {
4798       \l_stex_get_structure_module_str
4799     }{}{0}{}
4800   }}
4801 }{\l__stex_structures_name_str}
4802 % {
4803   \str_set:Nx \l_stex_get_symbol_uri_str { \l_stex_current_module_str?\l__stex_structures
4804   \tl_set:Nn \l_stex_notation_after_do_tl { \l__stex_notation_final:}
4805   \stex_notation_do:nnnnn{}{}{}{}{\comp{#4}}
4806 % }
4807 %\exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}
4808 \endgroup
4809 \stex_smsmode_do:\ignorespacesandpars
4810 }
4811
4812 \cs_new_protected:Nn \stex_symbol_or_var:n {
4813   \cs_if_exist:cTF{#1}{
4814     \cs_set_eq:Nc \l_tmpa_tl { #1 }
4815     \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4816     \str_if_empty:NTF \l_tmpa_str {
4817       \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
4818       \stex_invoke_variable:n {
4819         \bool_set_true:N \l_stex_symbol_or_var_bool
4820         \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4821         \tl_set:Nx \l_tmpa_tl {\exp_after:wN \use:n \l_tmpa_tl}
4822         \str_set:Nx \l_stex_get_symbol_uri_str {
4823           \exp_after:wN \use:n \l_tmpa_tl
4824         }
4825       }{
4826         \bool_set_false:N \l_stex_symbol_or_var_bool
4827         \stex_get_symbol:n{#1}
4828       }
4829     }{
4830       \__stex_structures_symbolorvar_from_string:n{ #1 }
4831     }
4832   }{
4833     \__stex_structures_symbolorvar_from_string:n{ #1 }
4834   }
4835 }
4836

```

```

4837 \cs_new_protected:Nn \__stex_structures_symbol_or_var_from_string:n {
4838   \prop_if_exist:cTF {l_stex_variable_#1 _prop}{
4839     \bool_set_true:N \l_stex_symbol_or_var_bool
4840     \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
4841   }{
4842     \bool_set_false:N \l_stex_symbol_or_var_bool
4843     \stex_get_symbol:n{#1}
4844   }
4845 }
4846
4847 \keys_define:nn { stex / varinstantiate } {
4848   name          .str_set_x:N = \l__stex_structures_name_str,
4849   bind           .choices:nn =
4850     {forall,exists}
4851     {\str_set:Nx \l__stex_structures_bind_str {\l_keys_choice_tl}}
4852 }
4853
4854 \cs_new_protected:Nn \__stex_structures_varinstantiate_args:n {
4855   \str_clear:N \l__stex_structures_name_str
4856   \str_clear:N \l__stex_structures_bind_str
4857   \keys_set:nn { stex / varinstantiate } { #1 }
4858 }
4859
4860 \NewDocumentCommand \varinstantiate {m O{} m m O{}}{
4861   \begingroup
4862     \stex_get_structure:n {#3}
4863     \__stex_structures_varinstantiate_args:n { #2 }
4864     \str_if_empty:NT \l__stex_structures_name_str {
4865       \str_set:Nn \l__stex_structures_name_str { #1 }
4866     }
4867     \stex_if_do_html:TF{
4868       \stex_annotate:nnn{varinstance}{\l__stex_structures_name_str}
4869     }{\use:n}
4870     {
4871       \stex_if_do_html:T{
4872         \stex_annotate_invisible:nnn{domain}{\l_stex_get_structure_module_str}{ }
4873       }
4874       \seq_clear:N \l__stex_structures_fields_seq
4875       \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4876       \seq_map_inline:Nn \l_stex_collect_imports_seq {
4877         \seq_map_inline:cn {c_stex_module_##1_constants}{
4878           \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? #####1 }
4879         }
4880       }
4881       \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4882       \prop_clear:N \l_tmpa_prop
4883       \tl_if_empty:nF {#5} {
4884         \seq_set_split:Nnn \l_tmpa_seq , {#5}
4885         \seq_map_inline:Nn \l_tmpa_seq {
4886           \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4887           \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4888             \msg_error:nnn{stex}{error/keyval}{##1}
4889           }
4890           \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_stru

```

```

4891 \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4892 \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol
4893 \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
4894 \stex_if_do_html:T{
4895   \stex_annotate:nnn{assign}{\l__stex_structures_dom_str,
4896     \bool_if:NTF\l_stex_symbol_or_var_bool{var://}{\l_stex_get_symbol_uri_str}{}}
4897 }
4898 \bool_if:NTF \l_stex_symbol_or_var_bool {
4899   \exp_args:Nxx \str_if_eq:nnF
4900     {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4901     {\prop_item:cn{l_stex_variable\l_stex_get_symbol_uri_str _prop}{args}}{
4902     \msg_error:nnxxx{stex}{error/incompatible}
4903     {\l__stex_structures_dom_str}
4904     {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4905     {\l_stex_get_symbol_uri_str}
4906     {\prop_item:cn{l_stex_variable\l_stex_get_symbol_uri_str _prop}{args}}
4907   }
4908   \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n
4909 }{
4910   \exp_args:Nxx \str_if_eq:nnF
4911     {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4912     {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}{
4913     \msg_error:nnxxx{stex}{error/incompatible}
4914     {\l__stex_structures_dom_str}
4915     {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4916     {\l_stex_get_symbol_uri_str}
4917     {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}
4918   }
4919   \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n
4920 }
4921 }
4922 }
4923 \tl_gclear:N \g__stex_structures_aftergroup_tl
4924 \seq_map_inline:Nn \l__stex_structures_fields_seq {
4925   \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdecl
4926 \stex_debug:nn{varinstantiate}{Field~\l_tmpa_str :~##1}
4927 \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4928   \stex_find_notation:nn{##1}{}
4929   \cs_gset_eq:cc{g__stex_structures_tmpa_\l_tmpa_str _cs}
4930     {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4931   \stex_debug:nn{varinstantiate}{Notation:~\cs_meaning:c{g__stex_structures_tmpa_\l_
4932 \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
4933   \cs_gset_eq:cc {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
4934     {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4935   \stex_debug:nn{varinstantiate}{Operator~Notation:~\cs_meaning:c{g__stex_struct
4936 }
4937 }
4938 }
4939 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4940   \prop_set_from_keyval:cn { l_stex_variable_ \l_tmpa_str _prop}{
4941     name = \l_tmpa_str ,
4942     args = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4943     arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4944     assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}

```

```

4945     }
4946     \cs_set_eq:cc {stex_var_notation_\l_tmpa_str_cs}
4947     {g__stex_structures_tmpa_\l_tmpa_str_cs}
4948     \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str_cs}
4949     {g__stex_structures_tmpa_op_\l_tmpa_str_cs}
4950   }
4951   \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_inv
4952 }
4953 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4954   \prop_set_from_keyval:cn {l_stex_varinstance_\l__stex_structures_name_str_prop }{
4955     domain = \l_stex_get_structure_module_str ,
4956     \prop_to_keyval:N \l_tmpa_prop
4957   }
4958   \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
4959   \tl_set:cn {l_stex_varinstance_\l__stex_structures_name_str_op_tl}{
4960     \exp_args:Nnx \exp_not:N \use:nn {
4961       \str_set:Nn \exp_not:N \l_stex_current_symbol_str {var://\l__stex_structures_nam
4962       \_stex_term_omv:nn {var://\l__stex_structures_name_str}{
4963         \exp_not:n{
4964           \_varcomp{#4}
4965         }
4966       }
4967     }{
4968       \exp_not:n{\_stex_reset:N \l_stex_current_symbol_str}
4969     }
4970   }
4971 }
4972 }
4973 \stex_debug:nn{varinstantiate}{\expandafter\detokenize\expandafter{\g__stex_structures_a
4974 \aftergroup\g__stex_structures_aftergroup_tl
4975 \endgroup
4976 \stex_smsmode_do:\ignorespacesandpars
4977 }
4978
4979 \cs_new_protected:Nn \stex_invoke_instance:n {
4980   \peek_charcode_remove:NTF ! {
4981     \stex_invoke_symbol:n{#1}
4982   }{
4983     \_stex_invoke_instance:nn {#1}
4984   }
4985 }
4986
4987
4988 \cs_new_protected:Nn \stex_invoke_varinstance:n {
4989   \peek_charcode_remove:NTF ! {
4990     \exp_args:Nnx \use:nn {
4991       \def\comp{\_varcomp}
4992       \use:c{l_stex_varinstance_#1_op_tl}
4993     }{
4994       \_stex_reset:N \comp
4995     }
4996   }{
4997     \_stex_invoke_varinstance:nn {#1}
4998   }

```

```

4999 }
5000
5001 \cs_new_protected:Nn \_stex_invoke_instance:nn {
5002   \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
5003     \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
5004   }{
5005     \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
5006     \msg_error:nnxxx{stex}{error/unknownfield}{#2}{#1}{
5007       \prop_to_keyval:N \l_tmpa_prop
5008     }
5009   }
5010 }
5011
5012 \cs_new_protected:Nn \_stex_invoke_varinstance:nn {
5013   \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
5014     \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
5015     \l_tmpa_tl
5016   }{
5017     \msg_error:nnnnn{stex}{error/unknownfield}{#2}{#1}{
5018     }
5019   }

```

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

`\stex_invoke_structure:nnn`

```

5020 % #1: URI of the instance
5021 % #2: URI of the instantiated module
5022 \cs_new_protected:Nn \stex_invoke_structure:nnn {
5023   \tl_if_empty:nTF{ #3 }{
5024     \prop_set_eq:Nc \l__stex_structures_structure_prop {
5025       c_stex_feature_ #2 _prop
5026     }
5027     \tl_clear:N \l_tmpa_tl
5028     \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
5029     \seq_map_inline:Nn \l_tmpa_seq {
5030       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
5031       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
5032       \cs_if_exist:cT {
5033         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
5034       }{
5035         \tl_if_empty:NF \l_tmpa_tl {
5036           \tl_put_right:Nn \l_tmpa_tl {,}
5037         }
5038         \tl_put_right:Nx \l_tmpa_tl {
5039           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
5040         }
5041       }
5042     }
5043     \exp_args:No \mathstruct \l_tmpa_tl
5044   }{
5045     \stex_invoke_symbol:n{#1/#3}
5046   }
5047 }

```



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

5048 `\endpackage`

## Chapter 32

# STEX -Statements Implementation

```
5049 <*package>
5050
5051 %%%%%%%%%%% features.dtx %%%%%%%%%%%
5052
5053 <@@=stex_statements>
    Warnings and error messages
5054
\titleemph
5055 \def\titleemph#1{\textbf{#1}}
    (End definition for \titleemph. This function is documented on page ??.)
```

### 32.1 Definitions

#### definiendum

```
5056 \keys_define:nn {stex / definiendum }{
5057   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
5058   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
5059   root     .str_set_x:N    = \l__stex_statements_definiendum_root_str,
5060   gfa      .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
5061 }
5062 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
5063   \str_clear:N \l__stex_statements_definiendum_root_str
5064   \tl_clear:N \l__stex_statements_definiendum_post_tl
5065   \str_clear:N \l__stex_statements_definiendum_gfa_str
5066   \keys_set:nn { stex / definiendum }{ #1 }
5067 }
5068 \NewDocumentCommand \definiendum { O{} m m } {
5069   \__stex_statements_definiendum_args:n { #1 }
5070   \stex_get_symbol:n { #2 }
5071   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5072   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
5073     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```

5074     \tl_set:Nn \l_tmpa_tl { #3 }
5075   } {
5076     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
5077     \tl_set:Nn \l_tmpa_tl {
5078       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
5079     }
5080   }
5081 } {
5082   \tl_set:Nn \l_tmpa_tl { #3 }
5083 }
5084
5085 % TODO root
5086 \stex_html_backend:TF {
5087   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
5088 } {
5089   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
5090 }
5091 }
5092 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

(End definition for definiendum. This function is documented on page 41.)

#### definame

```

5093
5094 \NewDocumentCommand \definame { 0{ } m } {
5095   \__stex_statements_definiendum_args:n { #1 }
5096   % TODO: root
5097   \stex_get_symbol:n { #2 }
5098   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5099   \str_set:Nx \l_tmpa_str {
5100     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5101   }
5102   \str_replace_all:Nnn \l_tmpa_str {-} {~}
5103   \stex_html_backend:TF {
5104     \stex_if_do_html:T {
5105       \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5106         \l_tmpa_str\l__stex_statements_definiendum_post_tl
5107       }
5108     }
5109   } {
5110     \exp_args:Nnx \defemph@uri {
5111       \l_tmpa_str\l__stex_statements_definiendum_post_tl
5112     } { \l_stex_get_symbol_uri_str }
5113   }
5114 }
5115 \stex_deactivate_macro:Nn \definame {definition~environments}
5116
5117 \NewDocumentCommand \Definame { 0{ } m } {
5118   \__stex_statements_definiendum_args:n { #1 }
5119   \stex_get_symbol:n { #2 }
5120   \str_set:Nx \l_tmpa_str {
5121     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5122   }
5123   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}

```

```

5124 \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5125 \stex_html_backend:TF {
5126   \stex_if_do_html:T {
5127     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5128       \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5129     }
5130   }
5131 } {
5132   \exp_args:Nnx \defemph@uri {
5133     \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5134   } { \l_stex_get_symbol_uri_str }
5135 }
5136 }
5137 \stex_deactivate_macro:Nn \Definame {definition-environments}
5138
5139 \NewDocumentCommand \premise { m }{
5140   \noindent\stex_annotate:nnn{ premise }{}{\ignorespaces #1 }
5141 }
5142 \NewDocumentCommand \conclusion { m }{
5143   \noindent\stex_annotate:nnn{ conclusion }{}{\ignorespaces #1 }
5144 }
5145 \NewDocumentCommand \definiens { 0{} m }{
5146   \str_clear:N \l_stex_get_symbol_uri_str
5147   \tl_if_empty:nF {#1} {
5148     \stex_get_symbol:n { #1 }
5149   }
5150   \str_if_empty:NT \l_stex_get_symbol_uri_str {
5151     \int_compare:nNnTF {\clist_count:N \l__stex_statements_sdefinition_for_clist} = 1 {
5152       \str_set:Nx \l_stex_get_symbol_uri_str {\clist_item:Nn \l__stex_statements_sdefinition
5153     }{
5154       % TODO throw error
5155     }
5156   }
5157   \str_if_eq:eeT {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{module}}
5158   {\l_stex_current_module_str}{
5159     \str_if_eq:eeF {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defin
5160   }{true}{
5161     \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5162     \exp_args:Nx \stex_add_to_current_module:n {
5163       \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5164     }
5165   }
5166 }
5167 \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
5168 }
5169
5170 \NewDocumentCommand \varbindforall {m}{
5171   \stex_symbol_or_var:n {#1}
5172   \bool_if:NTF\l_stex_symbol_or_var_bool{
5173     \stex_if_do_html:T {
5174       \stex_annotate_invisible:nnn {bindtype}{forall,\l_stex_get_symbol_uri_str}{}
5175     }
5176   }{
5177     % todo throw error

```

```

5178 }
5179 }
5180
5181 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
5182 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
5183 \stex_deactivate_macro:Nn \definiens {definition~environments}
5184 \stex_deactivate_macro:Nn \varbindforall {definition~or~assertion~environments}
5185

```

(End definition for definame. This function is documented on page [41](#).)

#### sdefinition

```

5186
5187 \keys_define:nn {stex / sdefinition }{
5188   type      .str_set_x:N = \sdefinitiontype,
5189   id        .str_set_x:N = \sdefinitionid,
5190   name      .str_set_x:N = \sdefinitionname,
5191   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5192   title     .tl_set:N    = \sdefinitiontitle
5193 }
5194 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
5195   \str_clear:N \sdefinitiontype
5196   \str_clear:N \sdefinitionid
5197   \str_clear:N \sdefinitionname
5198   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5199   \tl_clear:N \sdefinitiontitle
5200   \keys_set:nn { stex / sdefinition }{ #1 }
5201 }
5202
5203 \NewDocumentEnvironment{sdefinition}{0{}}{
5204   \__stex_statements_sdefinition_args:n{ #1 }
5205   \stex_reactivate_macro:N \definiendum
5206   \stex_reactivate_macro:N \definame
5207   \stex_reactivate_macro:N \Definame
5208   \stex_reactivate_macro:N \premise
5209   \stex_reactivate_macro:N \definiens
5210   \stex_reactivate_macro:N \varbindforall
5211   \stex_if_smsmode:F{
5212     \seq_clear:N \l_tmpb_seq
5213     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5214       \tl_if_empty:nF{ ##1 }{
5215         \stex_get_symbol:n { ##1 }
5216         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5217           \l_stex_get_symbol_uri_str
5218         }
5219       }
5220     }
5221     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5222     \exp_args:Nnnx
5223     \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
5224     \str_if_empty:NF \sdefinitiontype {
5225       \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{ }
5226     }
5227     \str_if_empty:NF \sdefinitionname {

```

```

5228     \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5229   }
5230   \clist_set:No \l_tmpa_clist \sdefinitiontype
5231   \tl_clear:N \l_tmpa_tl
5232   \clist_map_inline:Nn \l_tmpa_clist {
5233     \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
5234       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
5235     }
5236   }
5237   \tl_if_empty:NTF \l_tmpa_tl {
5238     \__stex_statements_sdefinition_start:
5239   }{
5240     \l_tmpa_tl
5241   }
5242 }
5243 \stex_ref_new_doc_target:n \sdefinitionid
5244 \stex_smsmode_do:
5245 ){
5246   \stex_suppress_html:n {
5247     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5248   }
5249   \stex_if_smsmode:F {
5250     \clist_set:No \l_tmpa_clist \sdefinitiontype
5251     \tl_clear:N \l_tmpa_tl
5252     \clist_map_inline:Nn \l_tmpa_clist {
5253       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
5254         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
5255       }
5256     }
5257     \tl_if_empty:NTF \l_tmpa_tl {
5258       \__stex_statements_sdefinition_end:
5259     }{
5260       \l_tmpa_tl
5261     }
5262     \end{stex_annotate_env}
5263   }
5264 }

```

### **\stexpatchdefinition**

```

5265 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
5266   \stex_par:\noindent\titllemph{Definition\tl_if_empty:NF \sdefinitiontitle {
5267     ~(\sdefinitiontitle)
5268   }~}
5269 }
5270 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\stex_par:\medskip}
5271
5272 \newcommand\stexpatchdefinition[3]{} {
5273   \str_set:Nx \l_tmpa_str{ #1 }
5274   \str_if_empty:NTF \l_tmpa_str {
5275     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
5276     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
5277   }{
5278     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
5279     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }

```

```

5280     }
5281 }

```

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

`\inlinedef` inline:

```

5282 \keys_define:nn {stex / inlinedef }{
5283   type      .str_set_x:N = \sdefinitiontype,
5284   id        .str_set_x:N = \sdefinitionid,
5285   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5286   name      .str_set_x:N = \sdefinitionname
5287 }
5288 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
5289   \str_clear:N \sdefinitiontype
5290   \str_clear:N \sdefinitionid
5291   \str_clear:N \sdefinitionname
5292   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5293   \keys_set:nn { stex / inlinedef }{ #1 }
5294 }
5295 \NewDocumentCommand \inlinedef { 0{} m } {
5296   \beginngroup
5297   \__stex_statements_inlinedef_args:n{ #1 }
5298   \stex_reactivate_macro:N \definiendum
5299   \stex_reactivate_macro:N \definame
5300   \stex_reactivate_macro:N \Definame
5301   \stex_reactivate_macro:N \premise
5302   \stex_reactivate_macro:N \definiens
5303   \stex_reactivate_macro:N \varbindforall
5304   \stex_ref_new_doc_target:n \sdefinitionid
5305   \stex_if_smsmode:TF{\stex_suppress_html:n {
5306     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5307   }}{
5308     \seq_clear:N \l_tmpb_seq
5309     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5310       \tl_if_empty:nF{ ##1 }{
5311         \stex_get_symbol:n { ##1 }
5312         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5313           \l_stex_get_symbol_uri_str
5314         }
5315       }
5316     }
5317     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5318     \exp_args:Nnx
5319     \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpb_seq {,}}{
5320       \str_if_empty:NF \sdefinitiontype {
5321         \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5322       }
5323       #2
5324       \str_if_empty:NF \sdefinitionname {
5325         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sdefinitionname}}
5326         \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5327       }
5328     }
5329   }

```

```

5330 \endgroup
5331 \stex_smsmode_do:
5332 }

```

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

## 32.2 Assertions

sassertion

```

5333 \keys_define:nn {stex / sassertion }{
5334   type      .str_set_x:N = \sassertiontype,
5335   id        .str_set_x:N = \sassertionid,
5336   title     .tl_set:N     = \sassertiontitle ,
5337   for       .clist_set:N  = \l__stex_statements_sassertion_for_clist ,
5338   name      .str_set_x:N  = \sassertionname
5339 }
5340
5341 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
5342   \str_clear:N \sassertiontype
5343   \str_clear:N \sassertionid
5344   \str_clear:N \sassertionname
5345   \clist_clear:N \l__stex_statements_sassertion_for_clist
5346   \tl_clear:N \sassertiontitle
5347   \keys_set:nn { stex / sassertion }{ #1 }
5348 }
5349
5350 %\tl_new:N \g__stex_statements_aftergroup_tl
5351
5352 \NewDocumentEnvironment{sassertion}{0}{}{
5353   \__stex_statements_sassertion_args:n{ #1 }
5354   \stex_reactivate_macro:N \premise
5355   \stex_reactivate_macro:N \conclusion
5356   \stex_reactivate_macro:N \varbindforall
5357   \stex_if_smsmode:F {
5358     \seq_clear:N \l_tmpb_seq
5359     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5360       \tl_if_empty:nF{ ##1 }{
5361         \stex_get_symbol:n { ##1 }
5362         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5363           \l_stex_get_symbol_uri_str
5364         }
5365       }
5366     }
5367     \exp_args:Nnnx
5368     \begin{sassertion_env}{assertion}{\seq_use:Nn \l_tmpb_seq {},}}
5369     \str_if_empty:NF \sassertiontype {
5370       \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
5371     }
5372     \str_if_empty:NF \sassertionname {
5373       \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
5374     }
5375     \clist_set:Nn \l_tmpa_clist \sassertiontype
5376     \tl_clear:N \l_tmpa_tl

```



```

5377 \clist_map_inline:Nn \l_tmpa_clist {
5378   \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
5379     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
5380   }
5381 }
5382 \tl_if_empty:NTF \l_tmpa_tl {
5383   \__stex_statements_sassertion_start:
5384 }{
5385   \l_tmpa_tl
5386 }
5387 }
5388 \str_if_empty:NTF \sassertionid {
5389   \str_if_empty:NF \sassertionname {
5390     \stex_ref_new_doc_target:n {}
5391   }
5392 } {
5393   \stex_ref_new_doc_target:n \sassertionid
5394 }
5395 \stex_smsmode_do:
5396 }{
5397   \str_if_empty:NF \sassertionname {
5398     \stex_suppress_html:n{\stex_symdecl_do:nn}{\sassertionname}}
5399     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5400   }
5401   \stex_if_smsmode:F {
5402     \clist_set:No \l_tmpa_clist \sassertiontype
5403     \tl_clear:N \l_tmpa_tl
5404     \clist_map_inline:Nn \l_tmpa_clist {
5405       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
5406         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
5407       }
5408     }
5409     \tl_if_empty:NTF \l_tmpa_tl {
5410       \__stex_statements_sassertion_end:
5411     }{
5412       \l_tmpa_tl
5413     }
5414     \end{stex_annotate_env}
5415   }
5416 }

```

## \stexpatchassertion

```

5417
5418 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
5419   \stex_par:\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
5420     (\sassertiontitle)
5421   }~}
5422 }
5423 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\stex_par:\medskip}
5424
5425 \newcommand\stexpatchassertion[3] [] {
5426   \str_set:Nx \l_tmpa_str{ #1 }
5427   \str_if_empty:NTF \l_tmpa_str {
5428     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }

```

```

5429     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
5430   }{
5431     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
5432     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
5433   }
5434 }

```

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

`\inlineass` inline:

```

5435 \keys_define:nn {stex / inlineass }{
5436   type      .str_set_x:N = \sassertiontype,
5437   id        .str_set_x:N = \sassertionid,
5438   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
5439   name      .str_set_x:N = \sassertionname
5440 }
5441 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
5442   \str_clear:N \sassertiontype
5443   \str_clear:N \sassertionid
5444   \str_clear:N \sassertionname
5445   \clist_clear:N \l__stex_statements_sassertion_for_clist
5446   \keys_set:nn { stex / inlineass }{ #1 }
5447 }
5448 \NewDocumentCommand \inlineass { 0{} m } {
5449   \beginngroup
5450   \stex_reactivate_macro:N \premise
5451   \stex_reactivate_macro:N \conclusion
5452   \stex_reactivate_macro:N \varbindforall
5453   \__stex_statements_inlineass_args:n{ #1 }
5454   \str_if_empty:NTF \sassertionid {
5455     \str_if_empty:NF \sassertionname {
5456       \stex_ref_new_doc_target:n {}
5457     }
5458   } {
5459     \stex_ref_new_doc_target:n \sassertionid
5460   }
5461
5462   \stex_if_smsmode:TF{
5463     \str_if_empty:NF \sassertionname {
5464       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5465       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5466     }
5467   }{
5468     \seq_clear:N \l_tmpb_seq
5469     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5470       \tl_if_empty:nF{ ##1 }{
5471         \stex_get_symbol:n { ##1 }
5472         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5473           \l_stex_get_symbol_uri_str
5474         }
5475       }
5476     }
5477     \exp_args:Nnx
5478     \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpb_seq {,}}{

```

```

5479     \str_if_empty:NF \sassertiontype {
5480       \stex_annotate_invisible:nnn{typestrings}{\sassertiontype}{ }
5481     }
5482     #2
5483     \str_if_empty:NF \sassertionname {
5484       \stex_suppress_html:n{\stex_symdecl_do:nn{ }{\sassertionname}}
5485       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5486       \stex_annotate_invisible:nnn{statementname}{\sassertionname}{ }
5487     }
5488   }
5489 }
5490 \endgroup
5491 \stex_smsmode_do:
5492 }

```

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

## 32.3 Examples

`sexample`

```

5493
5494 \keys_define:nn {stex / sexample }{
5495   type      .str_set_x:N = \exampletype,
5496   id        .str_set_x:N = \sexampleid,
5497   title     .tl_set:N    = \sexampletitle,
5498   name      .str_set_x:N = \sexamplename ,
5499   for       .clist_set:N = \l__stex_statements_sexample_for_clist,
5500 }
5501 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
5502   \str_clear:N \sexampletype
5503   \str_clear:N \sexampleid
5504   \str_clear:N \sexamplename
5505   \tl_clear:N \sexampletitle
5506   \clist_clear:N \l__stex_statements_sexample_for_clist
5507   \keys_set:nn { stex / sexample }{ #1 }
5508 }
5509
5510 \NewDocumentEnvironment{sexample}{0{}}{
5511   \__stex_statements_sexample_args:n{ #1 }
5512   \stex_reactivate_macro:N \premise
5513   \stex_reactivate_macro:N \conclusion
5514   \stex_if_smsmode:F {
5515     \seq_clear:N \l_tmpb_seq
5516     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5517       \tl_if_empty:nF{ ##1 }{
5518         \stex_get_symbol:n { ##1 }
5519         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5520           \l_stex_get_symbol_uri_str
5521         }
5522       }
5523     }
5524     \exp_args:Nnnx
5525     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpb_seq {,}}

```

```

5526 \str_if_empty:NF \sexamplotype {
5527   \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{}
5528 }
5529 \str_if_empty:NF \sexamplename {
5530   \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
5531 }
5532 \clist_set:No \l_tmpa_clist \sexamplotype
5533 \tl_clear:N \l_tmpa_tl
5534 \clist_map_inline:Nn \l_tmpa_clist {
5535   \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
5536     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
5537   }
5538 }
5539 \tl_if_empty:NTF \l_tmpa_tl {
5540   \__stex_statements_sexample_start:
5541 }{
5542   \l_tmpa_tl
5543 }
5544 }
5545 \str_if_empty:NF \sexampleid {
5546   \stex_ref_new_doc_target:n \sexampleid
5547 }
5548 \stex_smsmode_do:
5549 }{
5550   \str_if_empty:NF \sexamplename {
5551     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5552   }
5553   \stex_if_smsmode:F {
5554     \clist_set:No \l_tmpa_clist \sexamplotype
5555     \tl_clear:N \l_tmpa_tl
5556     \clist_map_inline:Nn \l_tmpa_clist {
5557       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
5558         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
5559       }
5560     }
5561     \tl_if_empty:NTF \l_tmpa_tl {
5562       \__stex_statements_sexample_end:
5563     }{
5564       \l_tmpa_tl
5565     }
5566     \end{stex_annotate_env}
5567   }
5568 }

```

**\stexpatchexample**

```

5569
5570 \cs_new_protected:Nn \__stex_statements_sexample_start: {
5571   \stex_par:\noindent\titllemph{Example~\tl_if_empty:NF \sexampltitle {
5572     (\sexampltitle)
5573   }~}
5574 }
5575 \cs_new_protected:Nn \__stex_statements_sexample_end: {\stex_par:\medskip}
5576
5577 \newcommand\stexpatchexample[3]{} {

```

```

5578 \str_set:Nx \l_tmpa_str{ #1 }
5579 \str_if_empty:NTF \l_tmpa_str {
5580   \tl_set:Nn \__stex_statements_sexample_start: { #2 }
5581   \tl_set:Nn \__stex_statements_sexample_end: { #3 }
5582 }{
5583   \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
5584   \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
5585 }
5586 }

```

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

`\inlineex` inline:

```

5587 \keys_define:nn {stex / inlineex }{
5588   type      .str_set_x:N = \sexamplotype,
5589   id        .str_set_x:N = \sexampleid,
5590   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
5591   name      .str_set_x:N = \sexamplename
5592 }
5593 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
5594   \str_clear:N \sexamplotype
5595   \str_clear:N \sexampleid
5596   \str_clear:N \sexamplename
5597   \clist_clear:N \l__stex_statements_sexample_for_clist
5598   \keys_set:nn { stex / inlineex }{ #1 }
5599 }
5600 \NewDocumentCommand \inlineex { 0{} m } {
5601   \begingroup
5602   \stex_reactivate_macro:N \premise
5603   \stex_reactivate_macro:N \conclusion
5604   \__stex_statements_inlineex_args:n{ #1 }
5605   \str_if_empty:NF \sexampleid {
5606     \stex_ref_new_doc_target:n \sexampleid
5607   }
5608   \stex_if_smsmode:TF{
5609     \str_if_empty:NF \sexamplename {
5610       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sexamplename}}
5611   }
5612 }{
5613   \seq_clear:N \l_tmpb_seq
5614   \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5615     \tl_if_empty:nF{ ##1 }{
5616       \stex_get_symbol:n { ##1 }
5617       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5618         \l_stex_get_symbol_uri_str
5619       }
5620     }
5621   }
5622   \exp_args:Nnx
5623   \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpb_seq {,}}{
5624     \str_if_empty:NF \sexamplotype {
5625       \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{ }
5626     }
5627   #2

```

```

5628     \str_if_empty:NF \sexamplename {
5629       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sexamplename}}
5630       \stex_annotate_invisible:nnn{statementname}{\sexamplename}{ }
5631     }
5632   }
5633 }
5634 \endgroup
5635 \stex_smsmode_do:
5636 }

```

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

## 32.4 Logical Paragraphs

`sparagraph`

```

5637 \keys_define:nn { stex / sparagraph } {
5638   id      .str_set:x:N = \sparagraphid ,
5639   title   .tl_set:N    = \l_stex_sparagraph_title_tl ,
5640   type    .str_set:x:N = \sparagraphtype ,
5641   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5642   from    .tl_set:N    = \sparagraphfrom ,
5643   to      .tl_set:N    = \sparagraphto ,
5644   start   .tl_set:N    = \l_stex_sparagraph_start_tl ,
5645   name    .str_set:N    = \sparagraphname ,
5646   imports .tl_set:N    = \l__stex_statements_sparagraph_imports_tl
5647 }
5648
5649 \cs_new_protected:Nn \stex_sparagraph_args:n {
5650   \tl_clear:N \l_stex_sparagraph_title_tl
5651   \tl_clear:N \sparagraphfrom
5652   \tl_clear:N \sparagraphto
5653   \tl_clear:N \l_stex_sparagraph_start_tl
5654   \tl_clear:N \l__stex_statements_sparagraph_imports_tl
5655   \str_clear:N \sparagraphid
5656   \str_clear:N \sparagraphtype
5657   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5658   \str_clear:N \sparagraphname
5659   \keys_set:nn { stex / sparagraph } { #1 }
5660 }
5661 \newif\if@in@omtext\@in@omtextfalse
5662
5663 \NewDocumentEnvironment {sparagraph} { 0{ } } {
5664   \stex_sparagraph_args:n { #1 }
5665   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5666     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
5667   }{
5668     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
5669   }
5670   \@in@omtexttrue
5671   \stex_if_smsmode:F {
5672     \seq_clear:N \l_tmpb_seq
5673     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5674       \tl_if_empty:nF{ ##1 }{

```

```

5675     \stex_get_symbol:n { ##1 }
5676     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5677         \l_stex_get_symbol_uri_str
5678     }
5679 }
5680 }
5681 \exp_args:Nnnx
5682 \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}
5683 \str_if_empty:NF \sparagraphtype {
5684     \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
5685 }
5686 \str_if_empty:NF \sparaagraphfrom {
5687     \stex_annotate_invisible:nnn{from}{\sparaagraphfrom}{}
5688 }
5689 \str_if_empty:NF \sparaagraphto {
5690     \stex_annotate_invisible:nnn{to}{\sparaagraphto}{}
5691 }
5692 \str_if_empty:NF \sparaagraphname {
5693     \stex_annotate_invisible:nnn{statementname}{\sparaagraphname}{}
5694 }
5695 \clist_set:No \l_tmpa_clist \sparaagraphtype
5696 \tl_clear:N \l_tmpa_tl
5697 \clist_map_inline:Nn \sparaagraphtype {
5698     \tl_if_exist:cT {__stex_statements_sparaagraph_##1_start:}{
5699         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparaagraph_##1_start:}}
5700     }
5701 }
5702 \stex_csl_to_imports:No \usemodule \l__stex_statements_sparaagraph_imports_tl
5703 \tl_if_empty:NTF \l_tmpa_tl {
5704     \__stex_statements_sparaagraph_start:
5705 }{
5706     \l_tmpa_tl
5707 }
5708 }
5709 \clist_set:No \l_tmpa_clist \sparaagraphtype
5710 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5711 {
5712     \stex_reactivate_macro:N \definiendum
5713     \stex_reactivate_macro:N \definame
5714     \stex_reactivate_macro:N \Definame
5715     \stex_reactivate_macro:N \premise
5716     \stex_reactivate_macro:N \definiens
5717 }
5718 \str_if_empty:NTF \sparaagraphid {
5719     \str_if_empty:NTF \sparaagraphname {
5720         \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5721             \stex_ref_new_doc_target:n {}
5722         }
5723     } {
5724         \stex_ref_new_doc_target:n {}
5725     }
5726 } {
5727     \stex_ref_new_doc_target:n \sparaagraphid
5728 }

```

```

5729 \exp_args:NNx
5730 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5731   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5732     \tl_if_empty:nF{ ##1 }{
5733       \stex_get_symbol:n { ##1 }
5734       \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5735     }
5736   }
5737 }
5738 \stex_smsmode_do:
5739 \ignorespacesandpars
5740 }{
5741   \str_if_empty:NF \sparagraphname {
5742     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5743     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5744   }
5745   \stex_if_smsmode:F {
5746     \clist_set:No \l_tmpa_clist \sparagraphtype
5747     \tl_clear:N \l_tmpa_tl
5748     \clist_map_inline:Nn \l_tmpa_clist {
5749       \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
5750         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
5751       }
5752     }
5753     \tl_if_empty:NTF \l_tmpa_tl {
5754       \__stex_statements_sparagraph_end:
5755     }{
5756       \l_tmpa_tl
5757     }
5758     \end{stex_annotate_env}
5759   }
5760 }

```

## **\stexpatchparagraph**

```

5761
5762 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
5763   \stex_par:\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5764     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
5765       \titleemph{\l_stex_sparagraph_title_tl}:~
5766     }
5767   }{
5768     \titleemph{\l_stex_sparagraph_start_tl}~
5769   }
5770 }
5771 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\stex_par:\medskip}
5772
5773 \newcommand\stexpatchparagraph[3] [] {
5774   \str_set:Nx \l_tmpa_str{ #1 }
5775   \str_if_empty:NTF \l_tmpa_str {
5776     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
5777     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
5778   }{
5779     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2 }
5780     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }

```



```

5781     }
5782 }
5783
5784 \keys_define:nn { stex / inlinepara } {
5785   id      .str_set:N = \sparagraphid ,
5786   type    .str_set:N = \sparagraphtype ,
5787   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5788   from    .tl_set:N   = \sparagraphfrom ,
5789   to      .tl_set:N   = \sparagraphto ,
5790   name    .str_set:N   = \sparagraphname
5791 }
5792 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
5793   \tl_clear:N \sparagraphfrom
5794   \tl_clear:N \sparagraphto
5795   \str_clear:N \sparagraphid
5796   \str_clear:N \sparagraphtype
5797   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5798   \str_clear:N \sparagraphname
5799   \keys_set:nn { stex / inlinepara }{ #1 }
5800 }
5801 \NewDocumentCommand \inlinepara { 0{} m } {
5802   \begingroup
5803   \__stex_statements_inlinepara_args:n{ #1 }
5804   \clist_set:Nn \l_tmpa_clist \sparagraphtype
5805   \str_if_empty:NTF \sparagraphid {
5806     \str_if_empty:NTF \sparagraphname {
5807       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5808         \stex_ref_new_doc_target:n {}
5809       }
5810     } {
5811       \stex_ref_new_doc_target:n {}
5812     }
5813   } {
5814     \stex_ref_new_doc_target:n \sparagraphid
5815   }
5816   \stex_if_smsmode:TF{
5817     \str_if_empty:NF \sparagraphname {
5818       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
5819     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5820   }
5821 }{
5822   \seq_clear:N \l_tmpb_seq
5823   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5824     \tl_if_empty:nF{ ##1 }{
5825       \stex_get_symbol:n { ##1 }
5826       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5827         \l_stex_get_symbol_uri_str
5828       }
5829     }
5830   }
5831   \exp_args:Nnx
5832   \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}{
5833     \str_if_empty:NF \sparagraphtype {
5834       \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{

```

```

5835     }
5836     \str_if_empty:NF \sparagraphfrom {
5837       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5838     }
5839     \str_if_empty:NF \sparagraphto {
5840       \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5841     }
5842     \str_if_empty:NF \sparagraphname {
5843       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5844       \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
5845       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5846     }
5847     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5848       \clist_map_inline:Nn \l_tmpb_seq {
5849         \stex_ref_new_sym_target:n {##1}
5850       }
5851     }
5852     #2
5853   }
5854 }
5855 \endgroup
5856 \stex_smsmode_do:
5857 }
5858

```

(End definition for `\stexpatchparagraph`. This function is documented on page [47](#).)

```

5859 </package>

```

## Chapter 33

# The Implementation

```
5860 <*package>
5861 <@@=stex_sproof>
5862
5863 %%%%%%%%%%    sproof.dtx    %%%%%%%%%%
5864
```

### 33.1 Proofs

We first define some keys for the proof environment.

```
5865 \keys_define:nn { stex / spf } {
5866   id          .str_set_x:N = \spfid,
5867   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,
5868   from        .tl_set:N    = \l__stex_sproof_spf_from_tl ,
5869   proofend    .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
5870   type        .str_set_x:N = \spftype,
5871   title       .tl_set:N    = \spftitle,
5872   continues   .tl_set:N    = \l__stex_sproof_spf_continues_tl,
5873   functions   .tl_set:N    = \l__stex_sproof_spf_functions_tl,
5874   method      .tl_set:N    = \l__stex_sproof_spf_method_tl
5875 }
5876 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
5877   \str_clear:N \spfid
5878   \tl_clear:N \l__stex_sproof_spf_for_tl
5879   \tl_clear:N \l__stex_sproof_spf_from_tl
5880   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
5881   \str_clear:N \spftype
5882   \tl_clear:N \spftitle
5883   \tl_clear:N \l__stex_sproof_spf_continues_tl
5884   \tl_clear:N \l__stex_sproof_spf_functions_tl
5885   \tl_clear:N \l__stex_sproof_spf_method_tl
5886   \bool_set_false:N \l__stex_sproof_inc_counter_bool
5887   \keys_set:nn { stex / spf }{ #1 }
5888 }
```

\c\_\_stex\_sproof\_flow\_str

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

```
5889 \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.

```

5890 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
5891 \cs_new_protected:Npn \sproofnumber {
5892   \int_set:Nn \l_tmpa_int {1}
5893   \bool_while_do:nn {
5894     \int_compare_p:nNn {
5895       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5896     } > 0
5897   }{
5898     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
5899     \int_incr:N \l_tmpa_int
5900   }
5901 }
5902 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
5903   \int_set:Nn \l_tmpa_int {1}
5904   \bool_while_do:nn {
5905     \int_compare_p:nNn {
5906       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5907     } > 0
5908   }{
5909     \int_incr:N \l_tmpa_int
5910   }
5911   \int_compare:nNnF \l_tmpa_int = 1 {
5912     \int_decr:N \l_tmpa_int
5913   }
5914   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
5915     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1
5916   }
5917 }
5918
5919 \cs_new_protected:Npn \__stex_sproof_add_counter: {
5920   \int_set:Nn \l_tmpa_int {1}
5921   \bool_while_do:nn {
5922     \int_compare_p:nNn {
5923       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5924     } > 0
5925   }{
5926     \int_incr:N \l_tmpa_int
5927   }
5928   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 1 }
5929 }
5930
5931 \cs_new_protected:Npn \__stex_sproof_remove_counter: {
5932   \int_set:Nn \l_tmpa_int {1}
5933   \bool_while_do:nn {

```

```

5934 \int_compare_p:nNn {
5935 \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5936 } > 0
5937 }{
5938 \int_incr:N \l_tmpa_int
5939 }
5940 \int_decr:N \l_tmpa_int
5941 \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 0 }
5942 }

```

**\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

```

5943 \def\sproof@box{
5944 \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
5945 }
5946 \def\sproofend{
5947 \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
5948 \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
5949 }
5950 }

```

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

**spf@\*@kw**

```

5951 \def\spf@proofsketch@kw{Proof~Sketch}
5952 \def\spf@proof@kw{Proof}
5953 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

5954 \AddToHook{begindocument}{
5955 \ltx@ifpackageloaded{babel}{
5956 \makeatletter
5957 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5958 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5959 \input{sproof-ngerman.ldf}
5960 }
5961 \clist_if_in:NnT \l_tmpa_clist {finnish}{
5962 \input{sproof-finnish.ldf}
5963 }
5964 \clist_if_in:NnT \l_tmpa_clist {french}{
5965 \input{sproof-french.ldf}
5966 }
5967 \clist_if_in:NnT \l_tmpa_clist {russian}{
5968 \input{sproof-russian.ldf}
5969 }
5970 \makeatother
5971 }{}
5972 }

```

**spfsketch**

```

5973 \newcommand\spsketch[2][]{
5974 \begin{group}
5975 \let \premise \stex_proof_premise:

```

```

5976 \__stex_sproof_spf_args:n{#1}
5977 \stex_if_smsmode:TF {
5978   \str_if_empty:NF \spfid {
5979     \stex_ref_new_doc_target:n \spfid
5980   }
5981 }{
5982   \seq_clear:N \l_tmpa_seq
5983   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5984     \tl_if_empty:nF{ ##1 }{
5985       \stex_get_symbol:n { ##1 }
5986       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5987         \l_stex_get_symbol_uri_str
5988       }
5989     }
5990   }
5991   \exp_args:Nnx
5992   \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
5993     \str_if_empty:NF \spftype {
5994       \stex_annotate_invisible:nnn{type}{\spftype}{
5995     }
5996     \clist_set:No \l_tmpa_clist \spftype
5997     \tl_set:Nn \l_tmpa_tl {
5998       \titleemph{
5999         \tl_if_empty:NTF \spftitle {
6000           \spf@proofsketch@kw
6001         }{
6002           \spftitle
6003         }
6004       }::~
6005     }
6006     \clist_map_inline:Nn \l_tmpa_clist {
6007       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6008         \tl_clear:N \l_tmpa_tl
6009       }
6010     }
6011     \str_if_empty:NF \spfid {
6012       \stex_ref_new_doc_target:n \spfid
6013     }
6014     \l_tmpa_tl #2 \sproofend
6015   }
6016 }
6017 \endgroup
6018 \stex_smsmode_do:
6019 }
6020

```

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

**spfeq** This is very similar to `\spfsketch`, but uses a computation array<sup>1415</sup>

```

6021 \newenvironment{spfeq}[2][ ]{
6022   \__stex_sproof_spf_args:n{#1}

```

<sup>14</sup>EDNOTE: This should really be more like a tabular with an ensuremath in it. or invoke text on the last column

<sup>15</sup>EDNOTE: document above

```

6023 \let \premise \stex_proof_premise:
6024 \stex_if_smsmode:TF {
6025   \str_if_empty:NF \spfid {
6026     \stex_ref_new_doc_target:n \spfid
6027   }
6028 }{
6029   \seq_clear:N \l_tmpa_seq
6030   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6031     \tl_if_empty:nF{ ##1 }{
6032       \stex_get_symbol:n { ##1 }
6033       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6034         \l_stex_get_symbol_uri_str
6035       }
6036     }
6037   }
6038   \exp_args:Nnnx
6039   \begin{stex_annotate_env}{spfeq}{\seq_use:Nn \l_tmpa_seq {,}}
6040   \str_if_empty:NF \spftype {
6041     \stex_annotate_invisible:nnn{type}{\spftype}{}
6042   }
6043
6044   \clist_set:No \l_tmpa_clist \spftype
6045   \tl_clear:N \l_tmpa_tl
6046   \clist_map_inline:Nn \l_tmpa_clist {
6047     \tl_if_exist:cT {__stex_sproof_spfeq_##1_start:}{
6048       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_start:}}
6049     }
6050     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6051       \tl_set:Nn \l_tmpa_tl {\use:n{}}
6052     }
6053   }
6054   \tl_if_empty:NTF \l_tmpa_tl {
6055     \__stex_sproof_spfeq_start:
6056   }{
6057     \l_tmpa_tl
6058   }{~#2}
6059   \str_if_empty:NF \spfid {
6060     \stex_ref_new_doc_target:n \spfid
6061   }
6062   \begin{displaymath}\begin{array}{rc1l}
6063   }
6064   \stex_smsmode_do:
6065 }{
6066   \stex_if_smsmode:F {
6067     \end{array}\end{displaymath}
6068     \clist_set:No \l_tmpa_clist \spftype
6069     \tl_clear:N \l_tmpa_tl
6070     \clist_map_inline:Nn \l_tmpa_clist {
6071       \tl_if_exist:cT {__stex_sproof_spfeq_##1_end:}{
6072         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_end:}}
6073       }
6074     }
6075     \tl_if_empty:NTF \l_tmpa_tl {
6076       \__stex_sproof_spfeq_end:

```

```

6077     }{
6078       \l_tmpa_tl
6079     }
6080     \end{stex_annotate_env}
6081   }
6082 }
6083
6084 \cs_new_protected:Nn \__stex_sproof_spfeq_start: {
6085   \titleemph{
6086     \tl_if_empty:NTF \spftitle {
6087       \spf@proof@kw
6088     }{
6089       \spftitle
6090     }
6091   }:
6092 }
6093 \cs_new_protected:Nn \__stex_sproof_spfeq_end: {\sproofend}
6094
6095 \newcommand\stexpatchspfeq[3] [] {
6096   \str_set:Nx \l_tmpa_str{ #1 }
6097   \str_if_empty:NTF \l_tmpa_str {
6098     \tl_set:Nn \__stex_sproof_spfeq_start: { #2 }
6099     \tl_set:Nn \__stex_sproof_spfeq_end: { #3 }
6100   }{
6101     \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_start:\endcsname{ #2 }
6102     \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_end:\endcsname{ #3 }
6103   }
6104 }
6105

```

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

**sproof** In this environment, we initialize the proof depth counter `\count10` to 10, and set up the description environment that will take the proof steps. At the end of the proof, we position the proof end into the last line.

```

6106 \newenvironment{sproof}[2] []{
6107   \let \premise \stex_proof_premise:
6108   \intarray_gzero:N \l__stex_sproof_counter_intarray
6109   \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
6110   \__stex_sproof_spf_args:n{#1}
6111   \stex_if_smsmode:TF {
6112     \str_if_empty:NF \spfid {
6113       \stex_ref_new_doc_target:n \spfid
6114     }
6115   }{
6116     \seq_clear:N \l_tmpa_seq
6117     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6118       \tl_if_empty:nF{ ##1 }{
6119         \stex_get_symbol:n { ##1 }
6120         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6121           \l_stex_get_symbol_uri_str
6122         }
6123       }
6124     }

```



```

6125 \exp_args:Nnnx
6126 \begin{stex_annotate_env}{sproof}{\seq_use:Nn \l_tmpa_seq {,}}
6127 \str_if_empty:NF \spftype {
6128   \stex_annotate_invisible:nnn{type}{\spftype}{}}
6129 }
6130
6131 \clist_set:No \l_tmpa_clist \spftype
6132 \tl_clear:N \l_tmpa_tl
6133 \clist_map_inline:Nn \l_tmpa_clist {
6134   \tl_if_exist:cT {__stex_sproof_sproof_##1_start:}{
6135     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_start:}}
6136   }
6137   \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6138     \tl_set:Nn \l_tmpa_tl {\use:n{}}
6139   }
6140 }
6141 \tl_if_empty:NTF \l_tmpa_tl {
6142   \__stex_sproof_sproof_start:
6143 }{
6144   \l_tmpa_tl
6145 }{~#2}
6146 \str_if_empty:NF \spfid {
6147   \stex_ref_new_doc_target:n \spfid
6148 }
6149 \begin{description}
6150 }
6151 \stex_smsmode_do:
6152 }{
6153   \stex_if_smsmode:F{
6154     \end{description}
6155     \clist_set:No \l_tmpa_clist \spftype
6156     \tl_clear:N \l_tmpa_tl
6157     \clist_map_inline:Nn \l_tmpa_clist {
6158       \tl_if_exist:cT {__stex_sproof_sproof_##1_end:}{
6159         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_end:}}
6160       }
6161     }
6162     \tl_if_empty:NTF \l_tmpa_tl {
6163       \__stex_sproof_sproof_end:
6164     }{
6165       \l_tmpa_tl
6166     }
6167     \end{stex_annotate_env}
6168   }
6169 }
6170
6171 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
6172   \par\noindent\titleemph{
6173     \tl_if_empty:NTF \spftype {
6174       \spf@proof@kw
6175     }{
6176       \spftype
6177     }
6178   }::

```

```

6179 }
6180 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
6181
6182 \newcommand\stexpatchproof[3] [] {
6183   \str_set:Nx \l_tmpa_str{ #1 }
6184   \str_if_empty:NTF \l_tmpa_str {
6185     \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
6186     \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
6187   }{
6188     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
6189     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
6190   }
6191 }

```

**\spfidea**

```

6192 \newcommand\spfidea[2] []{
6193   \__stex_sproof_spf_args:n{#1}
6194   \titleemph{
6195     \tl_if_empty:NTF \spftype {Proof-Idea}{
6196       \spftype
6197     }:
6198   }~#2
6199   \sproofend
6200 }

```

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

The next two environments (proof steps) and comments, are mostly semantical, they take KeyVal arguments that specify their semantic role. In draft mode, they read these values and show them. If the surrounding proof had `display=flow`, then no new `\item` is generated, otherwise it is. In any case, the proof step number (at the current level) is incremented.

**spfstep**

```

6201 \newenvironment{spfstep}[1] []{
6202   \__stex_sproof_spf_args:n{#1}
6203   \stex_if_smsmode:TF {
6204     \str_if_empty:NF \spfid {
6205       \stex_ref_new_doc_target:n \spfid
6206     }
6207   }{
6208     \@in@omtexttrue
6209     \seq_clear:N \l_tmpa_seq
6210     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6211       \tl_if_empty:nF{ ##1 }{
6212         \stex_get_symbol:n { ##1 }
6213         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6214           \l_stex_get_symbol_uri_str
6215         }
6216       }
6217     }
6218     \exp_args:Nnnx
6219     \begin{stex_annotate_env}{spfstep}{\seq_use:Nn \l_tmpa_seq {,}}
6220     \str_if_empty:NF \spftype {
6221       \stex_annotate_invisible:nnn{type}{\spftype}{ }

```

```

6222   }
6223   \clist_set:No \l_tmpa_clist \spftype
6224   \tl_set:Nn \l_tmpa_tl {
6225     \item[\sproofnumber]
6226     \bool_set_true:N \l__stex_sproof_inc_counter_bool
6227   }
6228   \clist_map_inline:Nn \l_tmpa_clist {
6229     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6230       \tl_clear:N \l_tmpa_tl
6231     }
6232   }
6233   \l_tmpa_tl
6234   \tl_if_empty:NF \spftitle {
6235     {(\titleemph{\spftitle})\enspace}
6236   }
6237   \str_if_empty:NF \spfid {
6238     \stex_ref_new_doc_target:n \spfid
6239   }
6240   }
6241   \stex_smsmode_do:
6242   \ignorespacesandpars
6243 }{
6244   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6245     \__stex_sproof_inc_counter:
6246   }
6247   \stex_if_smsmode:F {
6248     \end{stex_annotate_env}
6249   }
6250 }

```

**spfcomment**

```

6251 \newenvironment{spfcomment}[1][]{
6252   \__stex_sproof_spf_args:n{#1}
6253   \clist_set:No \l_tmpa_clist \spftype
6254   \tl_set:Nn \l_tmpa_tl {
6255     \item[\sproofnumber]
6256     \bool_set_true:N \l__stex_sproof_inc_counter_bool
6257   }
6258   \clist_map_inline:Nn \l_tmpa_clist {
6259     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6260       \tl_clear:N \l_tmpa_tl
6261     }
6262   }
6263   \l_tmpa_tl
6264 }{
6265   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6266     \__stex_sproof_inc_counter:
6267   }
6268 }

```

The next two environments also take a `KeyVal` argument, but also a regular one, which contains a start text. Both environments start a new numbered proof level.

**subproof** In the `subproof` environment, a new (lower-level) `proproof` environment is started.

```

6269 \newenvironment{subproof}[2][]{
6270   \__stex_sproof_spf_args:n{#1}
6271   \stex_if_smsmode:TF{
6272     \str_if_empty:NF \spfid {
6273       \stex_ref_new_doc_target:n \spfid
6274     }
6275   }{
6276     \seq_clear:N \l_tmpa_seq
6277     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6278       \tl_if_empty:nF{ ##1 }{
6279         \stex_get_symbol:n { ##1 }
6280         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6281           \l_stex_get_symbol_uri_str
6282         }
6283       }
6284     }
6285     \exp_args:Nnnx
6286     \begin{stex_annotate_env}{subproof}{\seq_use:Nn \l_tmpa_seq {,}}
6287     \str_if_empty:NF \spftype {
6288       \stex_annotate_invisible:nnn{type}{\spftype}{ }
6289     }
6290
6291     \clist_set:No \l_tmpa_clist \spftype
6292     \tl_set:Nn \l_tmpa_tl {
6293       \item[\sproofnumber]
6294       \bool_set_true:N \l__stex_sproof_inc_counter_bool
6295     }
6296     \clist_map_inline:Nn \l_tmpa_clist {
6297       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6298         \tl_clear:N \l_tmpa_tl
6299       }
6300     }
6301     \l_tmpa_tl
6302     \tl_if_empty:NF \spftitle {
6303       {(\titleemph{\spftitle})\enspace}
6304     }
6305     {~#2}
6306     \str_if_empty:NF \spfid {
6307       \stex_ref_new_doc_target:n \spfid
6308     }
6309   }
6310   \__stex_sproof_add_counter:
6311   \stex_smsmode_do:
6312 }{
6313   \__stex_sproof_remove_counter:
6314   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6315     \__stex_sproof_inc_counter:
6316   }
6317   \stex_if_smsmode:F{
6318     \end{stex_annotate_env}
6319   }
6320 }

```

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

```

6321 \newenvironment{spfcases}[2] [] {
6322   \tl_if_empty:nTF{#1}{
6323     \begin{subproof}[method=by-cases]{#2}
6324   }{
6325     \begin{subproof}[#1,method=by-cases]{#2}
6326   }
6327 }{
6328   \end{subproof}
6329 }

```

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

```

6330 \newenvironment{spfcase}[2] [] {
6331   \__stex_sproof_spf_args:n{#1}
6332   \stex_if_smsmode:TF {
6333     \str_if_empty:NF \spfid {
6334       \stex_ref_new_doc_target:n \spfid
6335     }
6336   }{
6337     \seq_clear:N \l_tmpa_seq
6338     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6339       \tl_if_empty:nF{ ##1 }{
6340         \stex_get_symbol:n { ##1 }
6341         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6342           \l_stex_get_symbol_uri_str
6343         }
6344       }
6345     }
6346     \exp_args:Nnnx
6347     \begin{stex_annotate_env}{spfcase}{\seq_use:Nn \l_tmpa_seq {,}}
6348     \str_if_empty:NF \spftype {
6349       \stex_annotate_invisible:nnn{type}{\spftype}{ }
6350     }
6351     \clist_set:No \l_tmpa_clist \spftype
6352     \tl_set:Nn \l_tmpa_tl {
6353       \item[\sproofnumber]
6354       \bool_set_true:N \l__stex_sproof_inc_counter_bool
6355     }
6356     \clist_map_inline:Nn \l_tmpa_clist {
6357       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6358         \tl_clear:N \l_tmpa_tl
6359       }
6360     }
6361     \l_tmpa_tl
6362     \tl_if_empty:nF{#2}{
6363       \titleemph{#2}:~
6364     }
6365   }
6366   \__stex_sproof_add_counter:
6367   \stex_smsmode_do:
6368 }{
6369   \__stex_sproof_remove_counter:
6370   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6371     \__stex_sproof_inc_counter:

```

```

6372 }
6373 \stex_if_smsmode:F{
6374   \clist_set:No \l_tmpa_clist \spftype
6375   \tl_set:Nn \l_tmpa_tl{\sproofend}
6376   \clist_map_inline:Nn \l_tmpa_clist {
6377     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6378       \tl_clear:N \l_tmpa_tl
6379     }
6380   }
6381   \l_tmpa_tl
6382   \end{stex_annotate_env}
6383 }
6384 }

```

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

```

6385 \newcommand\spfcasesketch[3] [] {
6386   \begin{spfcase}[#1]{#2}#3\end{spfcase}
6387 }

```

## 33.2 Justifications

We define the actions that are undertaken, when the keys for justifications are encountered. Here this is very simple, we just define an internal macro with the value, so that we can use it later.

```

6388 \keys_define:nn { stex / just }{
6389   id      .str_set_x:N = \l__stex_sproof_just_id_str,
6390   method  .tl_set:N   = \l__stex_sproof_just_method_tl,
6391   premises .tl_set:N   = \l__stex_sproof_just_premises_tl,
6392   args    .tl_set:N   = \l__stex_sproof_just_args_tl
6393 }

```

The next three environments and macros are purely semantic, so we ignore the keyval arguments for now and only display the content.<sup>16</sup>

**\spfjust**

```

6394 \newcommand\spfjust[1] [] {}

```

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

**\premise**

```

6395 \newcommand\stex_proof_promise:[2] [] {#2}

```

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

**\justarg** the **\justarg** macro is purely semantic, so we ignore the keyval arguments for now and only display the content.

```

6396 \newcommand\justarg[2] [] {#2}
6397 \</package>

```

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

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

---

<sup>16</sup>EdNOTE: need to do something about the premise in draft mode.

## Chapter 34

# STEX -Others Implementation

```
6398 <*package>
6399
6400 %%%%%%%%%% others.dtx %%%%%%%%%%
6401
6402 <@@=stex_others>
        Warnings and error messages
6403 % None

\MSC Math subject classifier

6404 \NewDocumentCommand \MSC {m} {
6405 % TODO
6406 }

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

6407 \@ifpackageloaded{tikzinput}{
6408 \RequirePackage{stex-tikzinput}
6409 }{}
6410
6411 \bool_if:NT \c_stex_persist_mode_bool {
6412 \input{\jobname.sms}
6413 \prop_if_exist:NT\c_stex_mathhub_main_manifest_prop{
6414 \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
6415 \l_tmpa_str
6416 \prop_set_eq:cN { c_stex_mathhub\_l_tmpa_str_manifest_prop }
6417 \c_stex_mathhub_main_manifest_prop
6418 \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
6419 }
6420 }
6421 </package>
```

## Chapter 35

# STEX -Metatheory Implementation

```
6422 <*package>
6423 <@@=stex_modules>
6424
6425 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
6426
6427 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX/meta}
6428 \begingroup
6429 \stex_module_setup:nn{
6430   ns=\c_stex_metatheory_ns_str,
6431   meta=NONE
6432 }{Metatheory}
6433 \stex_reactivate_macro:N \symdecl
6434 \stex_reactivate_macro:N \notation
6435 \stex_reactivate_macro:N \symdef
6436 \ExplSyntaxOff
6437 \csname stex_suppress_html:n\endcsname{
6438   % is-a (a:A, a \in A, a is an A, etc.)
6439   \symdecl{isa}[args=ai]
6440   \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
6441   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
6442   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
6443
6444   % bind (\forall, \Pi, \lambda etc.)
6445   \symdecl{bind}[args=Bi,assoc=pre]
6446   \notation{bind}[depfun,prec=nobrackets,op={(\cdot)\to\cdot}]{\comp( #1 \comp{)}\;\to\;}{#2}
6447   \notation{bind}[forall]{\comp\forall #1.\;#2}{##1 \comp, ##2}
6448   \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6449
6450   % implicit bind
6451   \symdecl{implicitbind}[args=Bi,assoc=pre]
6452   \notation{implicitbind}[braces,prec=nobrackets,op={(\cdot)\to\cdot}]{\comp\{ #1 \comp{\} \}
6453   \notation{implicitbind}[depfun,prec=nobrackets]{\comp( #1 \comp{)}\;\to_I\;}{#2}{##1 \comp,
6454   \notation{implicitbind}[Pi]{\comp\prod^I_{#1}#2}{##1\comp,##2}
6455
6456   % dummy variable
```



```

6457 \symdecl{dummyvar}
6458 \notation{dummyvar}[underscore]{\comp\_}
6459 \notation{dummyvar}[dot]{\comp\cdot}
6460 \notation{dummyvar}[dash]{\comp{\rm --}}
6461
6462 %fromto (function space, Hom-set, implication etc.)
6463 \symdecl{fromto}[args=ai]
6464 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
6465 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
6466
6467 % mapto (lambda etc.)
6468 \symdecl{mapto}[args=Bi]
6469 %\notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
6470 %\notation{mapto}[lambda]{\comp\lambda #1 \comp.; #2}{#1 \comp, #2}
6471 %\notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.; #2}{#1 \comp, #2}
6472
6473 % function/operator application
6474 \symdecl{apply}[args=ia]
6475 \notation{apply}[prec=0;0x\infprec,parens]{#1 \comp( #2 \comp)}{##1 \comp, ##2}
6476 \notation{apply}[prec=0;0x\infprec,lambda]{#1 \; #2 }{##1 \; ; ##2}
6477
6478 % collection of propositions/booleans/truth values
6479 \symdecl{prop}[name=proposition]
6480 \notation{prop}[prop]{\comp{\rm prop}}
6481 \notation{prop}[BOOL]{\comp{\rm BOOL}}
6482
6483 \symdecl{judgmentholds}[args=1]
6484 \notation{judgmentholds}[vdash,op=\vdash]{\comp\vdash\; ; #1}
6485
6486 % sequences
6487 \symdecl{seqtype}[args=1]
6488 \notation{seqtype}[kleene]{#1^{\comp\ast}}
6489
6490 \symdecl{seqexpr}[args=a]
6491 \notation{seqexpr}[angle,prec=nobrackets]{\comp\langle #1\comp\rangle}{##1\comp,##2}
6492
6493 \symdef{seqmap}[args=abi,setlike]{\comp\{#3 \comp| #2\comp\in \dobrackets{#1} \comp\}}{##1\comp,##2}
6494 \symdef{seqprepend}[args=ia]{#1 \comp{::} #2}{##1 \comp, ##2}
6495 \symdef{seqappend}[args=ai]{#1 \comp{::} #2}{##1 \comp, ##2}
6496 \symdef{seqfoldleft}[args=iabbi]{ \comp{foldl}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\comp}}{##1\comp,##2}
6497 \symdef{seqfoldright}[args=iabbi,op=foldr]{ \comp{foldr}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\comp}}{##1\comp,##2}
6498 \symdef{seqhead}[args=a]{\comp{head}\dobrackets{#1}}{##1 \comp, ##2}
6499 \symdef{seqtail}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6500 \symdef{seqlast}[args=a]{\comp{last}\dobrackets{#1}}{##1 \comp, ##2}
6501 \symdef{seqinit}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6502
6503 \symdef{sequence-index}[args=2,li,prec=nobrackets]{\comp{#1}_{#2}}{##1 \comp, ##2}
6504 \notation{sequence-index}[ui,prec=nobrackets]{\comp{#1}^{#2}}{##1 \comp, ##2}
6505
6506 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{,\ellipses}}{##1\comp,##2}
6507 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{,\ellipses,}#2}{##1\comp,##2}
6508 \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{,\ellipses,}#2\comp{,\ellipses,}#3}{##1\comp,##2,##3}
6509
6510 % letin ("let", local definitions, variable substitution)

```

```

6511 \symdecl{letin}[args=bii]
6512 \notation{letin}[let]{\comp{{\rm let}}\;#1\comp{=}#2\;\comp{{\rm in}}\;#3}
6513 \notation{letin}[subst]{#3 \comp[ #1 \comp/ #2 \comp]}
6514 \notation{letin}[frac]{#3 \comp[ \frac{#2}{#1} \comp]}
6515
6516 % structures
6517 \symdecl*{module-type}[args=1]
6518 \notation{module-type}{\comp{\mathtt{MOD}}} #1}
6519 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
6520 \notation{mathstruct}[angle,prec=nobrackets]{\comp\langle #1 \comp\rangle{##1 \comp, ##2}}
6521
6522 % objects
6523 \symdecl{object}
6524 \notation{object}{\comp{\mathtt{OBJECT}}}
6525
6526 }
6527
6528 % The following are abbreviations in the sTeX corpus that are left over from earlier
6529 % developments. They will eventually be phased out.
6530
6531 \ExplSyntaxOn
6532 \stex_add_to_current_module:n{
6533   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
6534   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
6535   \def\livar{\csname sequence-index\endcsname[li]}
6536   \def\uivar{\csname sequence-index\endcsname[ui]}
6537   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
6538   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
6539 }
6540 \__stex_modules_end_module:
6541 \endgroup
6542 </package>

```

## Chapter 36

# Tikzinput Implementation

```
6543 <@@=tikzinput>
6544 <*package>
6545
6546 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
6547
6548 \ProvidesExplPackage{tikzinput}{2022/02/26}{3.0.1}{tikzinput package}
6549 \RequirePackage{l3keys2e}
6550
6551 \keys_define:nn { tikzinput } {
6552   image .bool_set:N = \c_tikzinput_image_bool,
6553   image .default:n = false ,
6554   unknown .code:n = {}
6555 }
6556
6557 \ProcessKeysOptions { tikzinput }
6558
6559 \bool_if:NTF \c_tikzinput_image_bool {
6560   \RequirePackage{graphicx}
6561
6562   \providecommand\usetikzlibrary[]{}
6563   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
6564 }{
6565   \RequirePackage{tikz}
6566   \RequirePackage{standalone}
6567
6568   \newcommand \tikzinput [2] [] {
6569     \setkeys{Gin}{#1}
6570     \ifx \Gin@ewidth \Gin@exclamation
6571       \ifx \Gin@eheight \Gin@exclamation
6572         \input { #2 }
6573       \else
6574         \resizebox{!}{ \Gin@eheight }{
6575           \input { #2 }
6576         }
6577       \fi
6578     \else
6579       \ifx \Gin@eheight \Gin@exclamation
6580         \resizebox{ \Gin@ewidth }{!}{
```

```

6581         \input { #2 }
6582     }
6583     \else
6584         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
6585             \input { #2 }
6586         }
6587     \fi
6588 \fi
6589 }
6590 }
6591
6592 \newcommand \ctikzinput [2] [] {
6593     \begin{center}
6594         \tikzinput [#1] {#2}
6595     \end{center}
6596 }
6597
6598 \@ifpackageloaded{stex}{
6599     \RequirePackage{stex-tikzinput}
6600 }{}
6601
6602 </package>
6603 <*stex>
6604 \ProvidesExplPackage{stex-tikzinput}{2022/02/26}{3.0.1}{stex-tikzinput}
6605 \RequirePackage{stex}
6606 \RequirePackage{tikzinput}
6607
6608 \newcommand\mhtikzinput[2] []{%
6609     \def\Gin@mhrepos{ }\setkeys{Gin}{#1}%
6610     \stex_in_repository:nn\Gin@mhrepos{
6611         \tikzinput[#1]{\mhp\path{##1}{#2}}
6612     }
6613 }
6614 \newcommand\cmhtikzinput[2] [] {\begin{center}\mhtikzinput[#1]{#2}\end{center}}
6615
6616 \cs_new_protected:Nn \__tikzinput_usetikzlibrary:nn {
6617     \pgfkeys@spdef\pgf@temp{#1}
6618     \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
6619     \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgf@temp
6620     \expandafter\edef\csname tikz@library@#1@atcode\endcsname{\the\catcode'\@}
6621     \expandafter\edef\csname tikz@library@#1@barcode\endcsname{\the\catcode'\|}
6622     \expandafter\edef\csname tikz@library@#1@dollarcode\endcsname{\the\catcode'\$}
6623     \catcode'\@=11
6624     \catcode'\|=12
6625     \catcode'\$=3
6626     \pgfutil@InputIfFileExists{#2}{-}{-}
6627     \catcode'\@=\csname tikz@library@#1@atcode\endcsname
6628     \catcode'\|=\csname tikz@library@#1@barcode\endcsname
6629     \catcode'\$=\csname tikz@library@#1@dollarcode\endcsname
6630 }
6631
6632
6633 \newcommand\libusetikzlibrary[1]{

```

```

6634 \prop_if_exist:NF \l_stex_current_repository_prop {
6635   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6636 }
6637 \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
6638   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6639 }
6640 \seq_clear:N \l__tikzinput_libinput_files_seq
6641 \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
6642 \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
6643
6644 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
6645   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / tikzlibrary
6646   \IfFileExists{ \l_tmpa_str }{
6647     \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6648   }{}
6649   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
6650   \seq_put_right:No \l_tmpa_seq \l_tmpa_str
6651 }
6652
6653 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / tikzlibrary #1 .code.t
6654 \IfFileExists{ \l_tmpa_str }{
6655   \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6656 }{}
6657
6658 \seq_if_empty:NTF \l__tikzinput_libinput_files_seq {
6659   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .code.t
6660 }{
6661   \int_compare:nNnTF {\seq_count:N \l__tikzinput_libinput_files_seq} = 1 {
6662     \seq_map_inline:Nn \l__tikzinput_libinput_files_seq {
6663       \__tikzinput_usetikzlibrary:nn{#1}{ ##1 }
6664     }
6665   }{
6666     \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .co
6667   }
6668 }
6669 }
6670 </stex>

```

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

## Chapter 37

# document-structure.sty Implementation

```
6671 <*package>
6672 <@@=document_structure>
6673 \ProvidesExplPackage{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure}
6674 \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).

```
6675
6676 \keys_define:nn{ document-structure }{
6677   class      .str_set_x:N = \c_document_structure_class_str,
6678   topsect    .str_set_x:N = \c_document_structure_topsect_str,,
6679   unknown    .code:n      = {
6680     \PassOptionsToClass{\CurrentOption}{stex}
6681     \PassOptionsToClass{\CurrentOption}{tikzinput}
6682   }
6683   % showignores .bool_set:N = \c_document_structure_showignores_bool,
6684 }
6685 \ProcessKeysOptions{ document-structure }
6686 \str_if_empty:NT \c_document_structure_class_str {
6687   \str_set:Nn \c_document_structure_class_str {article}
6688 }
6689 \str_if_empty:NT \c_document_structure_topsect_str {
6690   \str_set:Nn \c_document_structure_topsect_str {section}
6691 }
```

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

```
6692 \RequirePackage{xspace}
6693 \RequirePackage{comment}
6694 \RequirePackage{stex}
6695 \AddToHook{begindocument}{}
```

```

6696 \ltx@ifpackageloaded{babel}{
6697   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6698   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6699     \makeatletter\input{document-structure-ngerman.ldf}\makeatother
6700   }
6701 }{}
6702 }

```

`\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.

```

6703 \int_new:N \l_document_structure_section_level_int
6704 \str_case:NnF \c_document_structure_topsect_str {
6705   {part}}{
6706     \int_set:Nn \l_document_structure_section_level_int {0}
6707   }
6708   {chapter}{
6709     \int_set:Nn \l_document_structure_section_level_int {1}
6710   }
6711 }{
6712   \str_case:NnF \c_document_structure_class_str {
6713     {book}{
6714       \int_set:Nn \l_document_structure_section_level_int {0}
6715     }
6716     {report}{
6717       \int_set:Nn \l_document_structure_section_level_int {0}
6718     }
6719   }{
6720     \int_set:Nn \l_document_structure_section_level_int {2}
6721   }
6722 }

```

## 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>17</sup>

EdN:17

```

6723 \def\current@section@level{document}%
6724 \newcommand\currentsectionlevel{\lowercase\expandafter\current@section@level\xspace}%
6725 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%

```

(End definition for `\currentsectionlevel`. This function is documented on page 52.)

`\skipfragment`

```

6726 \cs_new_protected:Npn \skipfragment {

```

---

<sup>17</sup>EdNOTE: MK: we may have to experiment with the more powerful uppercasing macro from `mfirstuc.sty` once we internationalize.

```

6727 \ifcase\l_document_structure_section_level_int
6728 \or\stepcounter{part}
6729 \or\stepcounter{chapter}
6730 \or\stepcounter{section}
6731 \or\stepcounter{subsection}
6732 \or\stepcounter{subsubsection}
6733 \or\stepcounter{paragraph}
6734 \or\stepcounter{subparagraph}
6735 \fi
6736 }

```

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

**blindfragment**

```

6737 \newcommand\at@begin@blindsfragment[1]{
6738 \newenvironment{blindfragment}
6739 {
6740 \int_incr:N\l_document_structure_section_level_int
6741 \at@begin@blindsfragment\l_document_structure_section_level_int
6742 }{}

```

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

```

6743 \newcommand\sfragment@nonum[2]{
6744 \ifx\hyper@anchor\@undefined\else\phantomsection\fi
6745 \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
6746 }

```

(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.

```

6747 \newcommand\sfragment@num[2]{
6748 \tl_if_empty:NTF \l__document_structure_sfragment_short_tl {
6749 \@nameuse{#1}{#2}
6750 }{
6751 \cs_if_exist:NTF\rdfmata@sectioning{
6752 \@nameuse{rdfmata@#1@old}[\l__document_structure_sfragment_short_tl]{#2}
6753 }{
6754 \@nameuse{#1}[\l__document_structure_sfragment_short_tl]{#2}
6755 }
6756 }
6757 %\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\sfragment@id
6758 }

```

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

**sfragment**

```

6759 \keys_define:nn { document-structure / sfragment }{
6760 id .str_set_x:N = \l__document_structure_sfragment_id_str,
6761 date .str_set_x:N = \l__document_structure_sfragment_date_str,

```



```

6762 creators      .clist_set:N = \l__document_structure_sfragment_creators_clist,
6763 contributors  .clist_set:N = \l__document_structure_sfragment_contributors_clist,
6764 srccite       .tl_set:N     = \l__document_structure_sfragment_srccite_tl,
6765 type          .tl_set:N     = \l__document_structure_sfragment_type_tl,
6766 short        .tl_set:N     = \l__document_structure_sfragment_short_tl,
6767 display       .tl_set:N     = \l__document_structure_sfragment_display_tl,
6768 intro        .tl_set:N     = \l__document_structure_sfragment_intro_tl,
6769 imports      .tl_set:N     = \l__document_structure_sfragment_imports_tl,
6770 loadmodules   .bool_set:N   = \l__document_structure_sfragment_loadmodules_bool
6771 }
6772 \cs_new_protected:Nn \l__document_structure_sfragment_args:n {
6773   \str_clear:N \l__document_structure_sfragment_id_str
6774   \str_clear:N \l__document_structure_sfragment_date_str
6775   \clist_clear:N \l__document_structure_sfragment_creators_clist
6776   \clist_clear:N \l__document_structure_sfragment_contributors_clist
6777   \tl_clear:N \l__document_structure_sfragment_srccite_tl
6778   \tl_clear:N \l__document_structure_sfragment_type_tl
6779   \tl_clear:N \l__document_structure_sfragment_short_tl
6780   \tl_clear:N \l__document_structure_sfragment_display_tl
6781   \tl_clear:N \l__document_structure_sfragment_imports_tl
6782   \tl_clear:N \l__document_structure_sfragment_intro_tl
6783   \bool_set_false:N \l__document_structure_sfragment_loadmodules_bool
6784   \keys_set:nn { document-structure / sfragment } { #1 }
6785 }

```

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.

```

6786 \newif@if@mainmatter\@mainmattertrue
6787 \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.

```

6788 \keys_define:nn { document-structure / sectioning }{
6789   name      .str_set_x:N = \l__document_structure_sect_name_str   ,
6790   ref       .str_set_x:N = \l__document_structure_sect_ref_str    ,
6791   clear     .bool_set:N  = \l__document_structure_sect_clear_bool ,
6792   clear     .default:n   = {true}                                ,
6793   num       .bool_set:N  = \l__document_structure_sect_num_bool   ,
6794   num       .default:n   = {true}
6795 }
6796 \cs_new_protected:Nn \l__document_structure_sect_args:n {
6797   \str_clear:N \l__document_structure_sect_name_str
6798   \str_clear:N \l__document_structure_sect_ref_str
6799   \bool_set_false:N \l__document_structure_sect_clear_bool
6800   \bool_set_false:N \l__document_structure_sect_num_bool
6801   \keys_set:nn { document-structure / sectioning } { #1 }
6802 }
6803 \newcommand\omdoc@sectioning[3][]{
6804   \l__document_structure_sect_args:n {#1 }
6805   \let\omdoc@sect@name\l__document_structure_sect_name_str
6806   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
6807   \if@mainmatter% numbering not overridden by frontmatter, etc.
6808     \bool_if:NTF \l__document_structure_sect_num_bool {

```

```

6809     \sfragment@num{#2}{#3}
6810   }{
6811     \sfragment@nonum{#2}{#3}
6812   }
6813   \def\current@section@level{\omdoc@sect@name}
6814 \else
6815   \sfragment@nonum{#2}{#3}
6816 \fi
6817 }% 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.

```

6818 \newcommand\sfragment@redefine@addtocontents[1]{%
6819 %\edef\__document_structureimport{#1}%
6820 %\@for\@I:=\__document_structureimport\do{%
6821 %\edef\@path{\csname module@\@I @path\endcsname}%
6822 %\@ifundefined{tf@toc}\relax%
6823 %    {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
6824 %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
6825 %\def\addcontentsline##1##2##3{%
6826 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6827 %\else% hyperref.sty not loaded
6828 %\def\addcontentsline##1##2##3{%
6829 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6830 %\fi
6831 }% 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.

```

6832 \newenvironment{sfragment}[2] []% keys, title
6833 {
6834   \__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.

```

6835   \stex_csl_to_imports:No \usemodule \l__document_structure_sfragment_imports_tl
6836
6837   \bool_if:NT \l__document_structure_sfragment_loadmodules_bool {
6838     \sfragment@redefine@addtocontents{
6839       %\@ifundefined{module@id}\used@modules%
6840       %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}
6841     }
6842   }

```

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

```

6843
6844   \stex_document_title:n { #2 }
6845
6846   \int_incr:N\l__document_structure_section_level_int
6847   \ifcase\l__document_structure_section_level_int
6848     \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
6849     \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}

```

```

6850 \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
6851 \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
6852 \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
6853 \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
6854 \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
6855 \fi
6856 \at@begin@sfragment[#1]\l__document_structure_section_level_int{#2}
6857 \str_if_empty:NF \l__document_structure_sfragment_id_str {
6858   \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
6859 }
6860 }% for customization
6861 {}

```

and finally, we localize the sections

```

6862 \newcommand\omdoc@part@kw{Part}
6863 \newcommand\omdoc@chapter@kw{Chapter}
6864 \newcommand\omdoc@section@kw{Section}
6865 \newcommand\omdoc@subsection@kw{Subsection}
6866 \newcommand\omdoc@subsubsection@kw{Subsubsection}
6867 \newcommand\omdoc@paragraph@kw{paragraph}
6868 \newcommand\omdoc@subparagraph@kw{subparagraph}

```

## 37.3 Front and Backmatter

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

`\printindex`

```

6869 \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).

```

6870 \cs_if_exist:NTF\frontmatter{
6871   \let\__document_structure_orig_frontmatter\frontmatter
6872   \let\frontmatter\relax
6873 }{
6874   \tl_set:Nn\__document_structure_orig_frontmatter{
6875     \clearpage
6876     \@mainmatterfalse
6877     \pagenumbering{roman}
6878   }
6879 }
6880 \cs_if_exist:NTF\backmatter{
6881   \let\__document_structure_orig_backmatter\backmatter
6882   \let\backmatter\relax
6883 }{
6884   \tl_set:Nn\__document_structure_orig_backmatter{
6885     \clearpage
6886     \@mainmatterfalse

```

```

6887 \pagenumbering{roman}
6888 }
6889 }

```

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

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

```

6890 \newenvironment{frontmatter}{
6891   \_document_structure_orig_frontmatter
6892 }{
6893   \cs_if_exist:NTF\mainmatter{
6894     \mainmatter
6895   }{
6896     \clearpage
6897     \@mainmattertrue
6898     \pagenumbering{arabic}
6899   }
6900 }

```

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

```

6901 \newenvironment{backmatter}{
6902   \_document_structure_orig_backmatter
6903 }{
6904   \cs_if_exist:NTF\mainmatter{
6905     \mainmatter
6906   }{
6907     \clearpage
6908     \@mainmattertrue
6909     \pagenumbering{arabic}
6910   }
6911 }

```

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

```

6912 \@mainmattertrue\pagenumbering{arabic}

```

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

```

6913 \def \c__document_structure_document_str{document}
6914 \newcommand\afterprematurestop{}
6915 \def\prematurestop@endsfragment{
6916   \unless\ifx\@currenvir\c__document_structure_document_str
6917     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
6918       \expandafter\prematurestop@endsfragment
6919     }
6920   }
6921 \providecommand\prematurestop{
6922   \message{Stopping~sTeX~processing~prematurely}
6923   \prematurestop@endsfragment
6924   \afterprematurestop
6925   \end{document}
6926 }

```

(End definition for `\prematurestop`. This function is documented on page 52.)

## 37.4 Global Variables

**\setSGvar** set a global variable

```
6927 \RequirePackage{etoolbox}
6928 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}
```

*(End definition for \setSGvar. This function is documented on page 52.)*

**\useSGvar** use a global variable

```
6929 \newrobustcmd\useSGvar[1]{%
6930   \@ifundefined{sTeX@Gvar@#1}
6931   {\PackageError{document-structure}
6932    {The sTeX Global variable #1 is undefined}
6933    {set it with \protect\setSGvar}}
6934   \@nameuse{sTeX@Gvar@#1}}
```

*(End definition for \useSGvar. This function is documented on page 52.)*

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

```
6935 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
6936   \@ifundefined{sTeX@Gvar@#1}
6937   {\PackageError{document-structure}
6938    {The sTeX Global variable #1 is undefined}
6939    {set it with \protect\setSGvar}}
6940   {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}
```

*(End definition for \ifSGvar. This function is documented on page 52.)*

## Chapter 38

# NotesSlides – Implementation

### 38.1 Class and Package Options

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

```
6941 \*cls)
6942 \@@=notesslides)
6943 \ProvidesExplClass{notesslides}{2022/02/28}{3.1.0}{notesslides Class}
6944 \RequirePackage{13keys2e}
6945
6946 \keys_define:nn{notesslides / cls}{
6947   class .str_set_x:N = \c__notesslides_class_str,
6948   notes .bool_set:N = \c__notesslides_notes_bool ,
6949   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
6950   docopt .str_set_x:N = \c__notesslides_docopt_str,
6951   unknown .code:n = {
6952     \PassOptionsToPackage{\CurrentOption}{document-structure}
6953     \PassOptionsToClass{\CurrentOption}{beamer}
6954     \PassOptionsToPackage{\CurrentOption}{notesslides}
6955     \PassOptionsToPackage{\CurrentOption}{stex}
6956   }
6957 }
6958 \ProcessKeysOptions{ notesslides / cls }
6959
6960 \str_if_empty:NF \c__notesslides_class_str {
6961   \PassOptionsToPackage{class=\c__notesslides_class_str}{document-structure}
6962 }
6963
6964 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{book}{
6965   \PassOptionsToPackage{defaultttopsect=part}{notesslides}
6966 }
6967 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{report}{
6968   \PassOptionsToPackage{defaultttopsect=part}{notesslides}
6969 }
6970
6971 \RequirePackage{stex}
```

```

6972 \stex_html_backend:T {
6973   \bool_set_true:N\c__notesslides_notes_bool
6974 }
6975
6976 \bool_if:NTF \c__notesslides_notes_bool {
6977   \PassOptionsToPackage{notes=true}{notesslides}
6978 }{
6979   \PassOptionsToPackage{notes=false}{notesslides}
6980 }
6981 \</cls>

```

now we do the same for the notesslides package.

```

6982 \*package>
6983 \ProvidesExplPackage{notesslides}{2022/02/28}{3.1.0}{notesslides Package}
6984 \RequirePackage{13keys2e}
6985
6986 \keys_define:nn{notesslides / pkg}{
6987   topsect          .str_set_x:N = \c__notesslides_topsect_str,
6988   defaulttopsect   .str_set_x:N = \c__notesslides_defaulttopsec_str,
6989   notes            .bool_set:N = \c__notesslides_notes_bool ,
6990   slides           .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
6991   sectocframes     .bool_set:N = \c__notesslides_sectocframes_bool ,
6992   frameimages      .bool_set:N = \c__notesslides_frameimages_bool ,
6993   fiboxed          .bool_set:N = \c__notesslides_fiboxed_bool ,
6994   noproblems       .bool_set:N = \c__notesslides_noproblems_bool,
6995   unknown          .code:n      = {
6996     \PassOptionsToClass{\CurrentOption}{stex}
6997     \PassOptionsToClass{\CurrentOption}{tikzinput}
6998   }
6999 }
7000 \ProcessKeysOptions{ notesslides / pkg }
7001
7002 \RequirePackage{stex}
7003 \stex_html_backend:T {
7004   \bool_set_true:N\c__notesslides_notes_bool
7005 }
7006
7007 \newif\ifnotes
7008 \bool_if:NTF \c__notesslides_notes_bool {
7009   \notesttrue
7010 }{
7011   \notesfalse
7012 }
7013

```

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

```

7014 \str_if_empty:NTF \c__notesslides_topsect_str {
7015   \str_set_eq:NN \__notesslidestopsect \c__notesslides_defaulttopsec_str
7016 }{
7017   \str_set_eq:NN \__notesslidestopsect \c__notesslides_topsect_str
7018 }
7019 \PassOptionsToPackage{topsect=\__notesslidestopsect}{document-structure}
7020 \</package>

```

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

```

7021 <*cls>
7022 \bool_if:NTF \c__notesslides_notes_bool {
7023   \str_if_empty:NT \c__notesslides_class_str {
7024     \str_set:Nn \c__notesslides_class_str {article}
7025   }
7026   \exp_after:wN\LoadClass\exp_after:wN[\c__notesslides_docostr]
7027     {\c__notesslides_class_str}
7028 }{
7029   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
7030   \newcounter{Item}
7031   \newcounter{paragraph}
7032   \newcounter{subparagraph}
7033   \newcounter{Hfootnote}
7034 }
7035 \RequirePackage{document-structure}

```

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

```

7036 \RequirePackage{notesslides}
7037 </cls>

```

In `notes` mode, we also have to make the `beamer`-specific things available to `article` via the `beamerarticle` package. We use options to avoid loading theorem-like environments, since we want to use our own from the `TeX` packages. The first batch of packages we want are loaded on `notesslides.sty`. These are the general ones, we will load the `TeX`-specific ones after we have done some work (e.g. defined the counters `m*`). Only the `stex`-logo package is already needed now for the default theme.

```

7038 <*package>
7039 \bool_if:NT \c__notesslides_notes_bool {
7040   \RequirePackage{a4wide}
7041   \RequirePackage{marginnote}
7042   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
7043   \RequirePackage{mdframed}
7044   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
7045   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
7046 }
7047 \RequirePackage{stex-tikzinput}
7048 \RequirePackage{etoolbox}
7049 \RequirePackage{amssymb}
7050 \RequirePackage{amsmath}
7051 \RequirePackage{comment}
7052 \RequirePackage{textcomp}
7053 \RequirePackage{url}
7054 \RequirePackage{graphicx}
7055 \RequirePackage{pgf}

```

## 38.2 Notes and Slides

For the lecture notes cases, we also provide the `\usetheme` macro that would otherwise come from the `beamer` class. While the latter loads `beamertheme<theme>.sty`, the



notes version loads `beamernotestheme(theme).sty`.<sup>18</sup>

```

7056 \bool_if:NT \c__notesslides_notes_bool {
7057   \renewcommand\usetheme[2] [] {\usepackage[#1]{beamernotestheme#2}}
7058 }
7059
7060
7061 \NewDocumentCommand \libusetheme {O{} m} {
7062   \bool_if:NTF \c__notesslides_notes_bool {
7063     \libusepackage[#1]{beamernotestheme#2}
7064   }{
7065     \libusepackage[#1]{beamertheme#2}
7066   }
7067 }

```

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

```

7068 \newcounter{slide}
7069 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
7070 \newlength{\slideheight}\setlength{\slideheight}{9cm}

```

**note** The `note` environment is used to leave out text in the `slides` mode. It does not have a counterpart in OMDoc. So for course notes, we define the `note` environment to be a no-operation otherwise we declare the `note` environment as a comment via the `comment` package.

```

7071 \bool_if:NTF \c__notesslides_notes_bool {
7072   \renewenvironment{note}{\ignorespaces}{}
7073 }{
7074   \excludecomment{note}
7075 }

```

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.

```

7076 \bool_if:NT \c__notesslides_notes_bool {
7077   \newlength{\slideframewidth}
7078   \setlength{\slideframewidth}{1.5pt}

```

**frame** We first define the keys.

```

7079 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
7080   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
7081     \bool_set_true:N #1
7082   }{
7083     \bool_set_false:N #1
7084   }
7085 }
7086 \keys_define:nn{notesslides / frame}{
7087   label .str_set_x:N = \l__notesslides_frame_label_str,
7088   allowframebreaks .code:n = {
7089     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
7090   },
7091   allowdisplaybreaks .code:n = {

```

---

<sup>18</sup>EdNOTE: MK: This is not ideal, but I am not sure that I want to be able to provide the full theme functionality there.

```

7092     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_allowdisplaybreaks\_bool { #1 }
7093 },
7094 fragile .code:n = {
7095     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_fragile\_bool { #1 }
7096 },
7097 shrink .code:n = {
7098     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_shrink\_bool { #1 }
7099 },
7100 squeeze .code:n = {
7101     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_squeeze\_bool { #1 }
7102 },
7103 t .code:n = {
7104     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_t\_bool { #1 }
7105 },
7106 unknown .code:n = {}
7107 }
7108 \cs\_new\_protected:Nn \_notesslides\_frame\_args:n {
7109     \str\_clear:N \l\_notesslides\_frame\_label\_str
7110     \bool\_set\_true:N \l\_notesslides\_frame\_allowframebreaks\_bool
7111     \bool\_set\_true:N \l\_notesslides\_frame\_allowdisplaybreaks\_bool
7112     \bool\_set\_true:N \l\_notesslides\_frame\_fragile\_bool
7113     \bool\_set\_true:N \l\_notesslides\_frame\_shrink\_bool
7114     \bool\_set\_true:N \l\_notesslides\_frame\_squeeze\_bool
7115     \bool\_set\_true:N \l\_notesslides\_frame\_t\_bool
7116     \keys\_set:nn { notesslides / frame }{ #1 }
7117 }

```

We define the environment, read them, and construct the slide number and label.

```

7118 \renewenvironment{frame}[1][]{
7119     \_notesslides\_frame\_args:n{#1}
7120     \sffamily
7121     \stepcounter{slide}
7122     \def\@currentlabel{\theslide}
7123     \str\_if\_empty:NF \l\_notesslides\_frame\_label\_str {
7124         \label{\l\_notesslides\_frame\_label\_str}
7125     }

```

We redefine the `itemize` environment so that it looks more like the one in `beamer`.

```

7126 \def\itemize@level{outer}
7127 \def\itemize@outer{outer}
7128 \def\itemize@inner{inner}
7129 \renewcommand\newpage{\addtocounter{framenum}{1}}
7130 %\newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
7131 \renewenvironment{itemize}{
7132     \ifx\itemize@level\itemize@outer
7133         \def\itemize@label{\$ \rhd \$}
7134     \fi
7135     \ifx\itemize@level\itemize@inner
7136         \def\itemize@label{\$ \scriptstyle \rhd \$}
7137     \fi
7138     \begin{list}
7139     {\itemize@label}
7140     {\setlength{\labelsep}{.3em}
7141      \setlength{\labelwidth}{.5em}
7142      \setlength{\leftmargin}{1.5em}

```

```

7143     }
7144     \edef\itemize@level{\itemize@inner}
7145   }{
7146     \end{list}
7147   }

```

We create the box with the `mdframed` environment from the `equinymous` package.

```

7148     \stex_html_backend:TF {
7149       \begin{stex_annotate_env}{frame}{}\vbox\bgroup
7150         \mdf@patchamsthm
7151       }{
7152         \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwid
7153       }
7154     }{
7155       \stex_html_backend:TF {
7156         \miko@slidelabel\egroup\end{stex_annotate_env}
7157       }\medskip\miko@slidelabel\end{mdframed}}
7158     }

```

Now, we need to redefine the `frametitle` (we are still in course notes mode).

`\frametitle`

```

7159   \renewcommand{\frametitle}[1]{
7160     \stex_document_title:n { #1 }
7161     {\Large\bf\sf\color{blue}{#1}}\medskip
7162   }
7163 }

```

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

EdN:19

`\pause`

```

19
7164 \bool_if:NT \c__notesslides_notes_bool {
7165   \newcommand\pause{}
7166 }

```

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

`nparagraph`

```

7167 \bool_if:NTF \c__notesslides_notes_bool {
7168   \newenvironment{nparagraph}[1] []{\begin{sparagraph}[#1]}\end{sparagraph}}
7169 }{
7170   \excludacomment{nparagraph}
7171 }

```

`nfragment`

```

7172 \bool_if:NTF \c__notesslides_notes_bool {
7173   \newenvironment{nfragment}[2] []{\begin{sfragment}[#1]{#2}}\end{sfragment}}
7174 }{
7175   \excludacomment{nfragment}
7176 }

```

---

<sup>19</sup>EdNOTE: MK: fake it in notes mode for now

ndefinition

```
7177 \bool_if:NTF \c__notesslides_notes_bool {
7178   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}\end{sdefinition}}
7179 }{
7180   \excludecomment{ndefinition}
7181 }
```

nassertion

```
7182 \bool_if:NTF \c__notesslides_notes_bool {
7183   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]\end{sassertion}}
7184 }{
7185   \excludecomment{nassertion}
7186 }
```

nsproof

```
7187 \bool_if:NTF \c__notesslides_notes_bool {
7188   \newenvironment{nproof}[2] [] {\begin{sproof}[#1]{#2}\end{sproof}}
7189 }{
7190   \excludecomment{nproof}
7191 }
```

nexample

```
7192 \bool_if:NTF \c__notesslides_notes_bool {
7193   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]\end{sexample}}
7194 }{
7195   \excludecomment{nexample}
7196 }
```

\inputref@\*skip We customize the hooks for in \inputref.

```
7197 \def\inputref@preskip{\smallskip}
7198 \def\inputref@postskip{\medskip}
```

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

**\inputref\***

```
7199 \let\orig@inputref\inputref
7200 \def\inputref{\@ifstar\ninputref\orig@inputref}
7201 \newcommand\ninputref[2] [] {
7202   \bool_if:NT \c__notesslides_notes_bool {
7203     \orig@inputref[#1]{#2}
7204   }
7205 }
```

(End definition for \inputref\*. This function is documented on page 54.)

## 38.3 Header and Footer Lines

Now, we set up the infrastructure for the footer line of the slides, we use boxes for the logos, so that they are only loaded once, that considerably speeds up processing.

**\setslidelo** The default logo is the  $\TeX$  logo. Customization can be done by `\setslidelo{<logo name>}`.

```

7206 \newlength{\slideloheight}
7207
7208 \bool_if:NTF \c_notesslides_notes_bool {
7209   \setlength{\slideloheight}{.4cm}
7210 }{
7211   \setlength{\slideloheight}{1cm}
7212 }
7213 \newsavebox{\slidelo}
7214 \sbox{\slidelo}{\TeX}
7215 \newrobustcmd{\setslidelo}[1]{
7216   \sbox{\slidelo}{\includegraphics[height=\slideloheight]{#1}}
7217 }

```

(End definition for `\setslidelo`. This function is documented on page 54.)

**\setsource** `\source` stores the writer's name. By default it is *Michael Kohlhase* since he is the main user and designer of this package. `\setsource{<name>}` can change the writer's name.

```

7218 \def\source{Michael Kohlhase}% customize locally
7219 \newrobustcmd{\setsource}[1]{\def\source{#1}}

```

(End definition for `\setsource`. This function is documented on page 54.)

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

```

7220 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
7221 \newsavebox{\cclogo}
7222 \sbox{\cclogo}{\includegraphics[height=\slideloheight]{stex-cc_somerights}}
7223 \newif\ifcchref\cchreffalse
7224 \AtBeginDocument{
7225   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
7226 }
7227 \def\licensing{
7228   \ifcchref
7229     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
7230   \else
7231     {\usebox{\cclogo}}
7232   \fi
7233 }
7234 \newrobustcmd{\setlicensing}[2][]{
7235   \def\@url{#1}
7236   \sbox{\cclogo}{\includegraphics[height=\slideloheight]{#2}}
7237   \ifx\@url\@empty
7238     \def\licensing{{\usebox{\cclogo}}}
7239   \else
7240     \def\licensing{
7241       \ifcchref
7242         \href{#1}{\usebox{\cclogo}}
7243       \else
7244         {\usebox{\cclogo}}
7245       \fi

```

```

7246     }
7247     \fi
7248 }

```

(End definition for `\setlicensing`. This function is documented on page 54.)

EdN:20

`\slidelabel` Now, we set up the slide label for the article mode.<sup>20</sup>

```

7249 \newrobustcmd\miko@slidelabel{
7250   \vbox to \slidelogoheight{
7251     \vss\hbox to \slidewidth
7252     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
7253   }
7254 }

```

(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.

```

7255 \def\Gin@mhrepos{}
7256 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7257 \define@key{Gin}{label}{\def\currentlabel{\arabic{slide}}\label{#1}}
7258 \newrobustcmd\frameimage[2][]{
7259   \stepcounter{slide}
7260   \bool_if:NT \c__notesslides_frameimages_bool {
7261     \def\Gin@ewidth{}\setkeys{Gin}{#1}
7262     \bool_if:NF \c__notesslides_notes_bool { \vfill }
7263     \begin{center}
7264       \bool_if:NTF \c__notesslides_fiboxed_bool {
7265         \fbox{
7266           \ifx\Gin@ewidth\@empty
7267             \ifx\Gin@mhrepos\@empty
7268               \mhgraphics[width=\slidewidth,#1]{#2}
7269             \else
7270               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7271             \fi
7272           \else% Gin@ewidth empty
7273             \ifx\Gin@mhrepos\@empty
7274               \mhgraphics[#1]{#2}
7275             \else
7276               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7277             \fi
7278           \fi% Gin@ewidth empty
7279         }
7280       }{
7281         \ifx\Gin@ewidth\@empty
7282           \ifx\Gin@mhrepos\@empty
7283             \mhgraphics[width=\slidewidth,#1]{#2}
7284           \else
7285             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7286           \fi

```

---

<sup>20</sup>EdNOTE: see that we can use the themes for the slides some day. This is all fake.

```

7287         \ifx\Gin@mhrepos\@empty
7288             \mhgraphics[#1]{#2}
7289         \else
7290             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7291         \fi
7292     \fi% Gin@ewidth empty
7293 }
7294 \end{center}
7295 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
7296 \bool_if:NF \c__notesslides_notes_bool { \vfill }
7297 }
7298 } % ifmks@sty@frameimages

```

(End definition for `\frameimage`. This function is documented on page 55.)

## 38.5 Colors and Highlighting

We first specify sans serif fonts as the default.

```

7299 \sffamily

```

Now, we set up an infrastructure for highlighting phrases in slides. Note that we use content-oriented macros for highlighting rather than directly using color markup. The first thing to do is to adapt the green so that it is dark enough for most beamers

```

7300 \AddToHook{begindocument}{
7301     \definecolor{green}{rgb}{0,.5,0}
7302     \definecolor{purple}{cmyk}{.3,1,0,.17}
7303 }

```

We customize the `\defemph`, `\symrefemph`, `\compemph`, and `\titleemph` macros with colors. Furthermore we customize the `\__omtextlec` macro for the appearance of line end comments in `\lec`.

```

7304 % \def\STpresent#1{\textcolor{blue}{#1}}
7305 \def\defemph#1{\textcolor{magenta}{#1}}
7306 \def\symrefemph#1{\textcolor{cyan}{#1}}
7307 \def\compemph#1{\textcolor{blue}{#1}}
7308 \def\titleemph#1{\textcolor{blue}{#1}}
7309 \def\__omtext_lec#1{\textcolor{green}{#1}}

```

I like to use the dangerous bend symbol for warnings, so we provide it here.

`\textwarning` as the macro can be used quite often we put it into a box register, so that it is only loaded once.

```

7310 \pgfdeclareimage[width=.8em]{miko@small@dbend}{stex-dangerous-bend}
7311 \def\smalltextwarning{
7312     \pgfuseimage{miko@small@dbend}
7313     \xspace
7314 }
7315 \pgfdeclareimage[width=1.2em]{miko@dbend}{stex-dangerous-bend}
7316 \newrobustcmd\textwarning{
7317     \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
7318     \xspace
7319 }
7320 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{stex-dangerous-bend}

```

```

7321 \newrobustcmd\bigtextwarning{
7322   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
7323   \xspace
7324 }
(End definition for \textwarning. This function is documented on page 55.)
7325 \newrobustcmd\putgraphicsat[3]{
7326   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
7327 }
7328 \newrobustcmd\putat[2]{
7329   \begin{picture}(0,0)\put(#1){#2}\end{picture}
7330 }

```

## 38.6 Sectioning

If the `sectocframes` option is set, then we make section frames. We first define counters for `part` and `chapter`, which `beamer.cls` does not have and we make the `section` counter which it does dependent on `chapter`.

```

7331 \stex_html_backend:F {
7332   \bool_if:NT \c__notesslides_sectocframes_bool {
7333     \str_if_eq:VnTF \__notesslidesstopsect{part}{
7334       \newcounter{chapter}\counterwithin*{section}{chapter}
7335     }{
7336       \str_if_eq:VnT \__notesslidesstopsect{chapter}{
7337         \newcounter{chapter}\counterwithin*{section}{chapter}
7338       }
7339     }
7340   }
7341 }

```

`\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
7342 \def\part@prefix{}
7343 \@ifpackageloaded{document-structure}{\{
7344   \str_case:VnF \__notesslidesstopsect {
7345     {part}{
7346       \int_set:Nn \l_document_structure_section_level_int {0}
7347       \def\thesection{\arabic{chapter}.\arabic{section}}
7348       \def\part@prefix{\arabic{chapter}.}
7349     }
7350     {chapter}{
7351       \int_set:Nn \l_document_structure_section_level_int {1}
7352       \def\thesection{\arabic{chapter}.\arabic{section}}
7353       \def\part@prefix{\arabic{chapter}.}
7354     }
7355   }{
7356     \int_set:Nn \l_document_structure_section_level_int {2}
7357     \def\part@prefix{}
7358   }
7359 }
7360
7361 \bool_if:NF \c__notesslides_notes_bool { % only in slides

```



(End definition for \section@level. This function is documented on page ??.)

The new counters are used in the `sfragment` environment that choses the L<sup>A</sup>T<sub>E</sub>X sectioning macros according to \section@level.

`sfragment`

```

7362 \renewenvironment{sfragment}[2][]{
7363   \_document\_structure\_sfragment\_args:n { #1 }
7364   \int\_incr:N \l\_document\_structure\_section\_level\_int
7365   \bool\_if:NT \c\_notesslides\_sectocframes\_bool {
7366     \stepcounter{slide}
7367     \begin{frame}[noframenumbering]
7368     \vfill\Large\centering
7369     \red{
7370       \ifcase\l\_document\_structure\_section\_level\_int\or
7371         \stepcounter{part}
7372         \def\_notesslideslabel{\omdoc@part@kw}~\Roman{part}}
7373         \def\currentsectionlevel{\omdoc@part@kw}
7374       \or
7375         \stepcounter{chapter}
7376         \def\_notesslideslabel{\omdoc@chapter@kw}~\arabic{chapter}}
7377         \def\currentsectionlevel{\omdoc@chapter@kw}
7378       \or
7379         \stepcounter{section}
7380         \def\_notesslideslabel{\part@prefix\arabic{section}}
7381         \def\currentsectionlevel{\omdoc@section@kw}
7382       \or
7383         \stepcounter{subsection}
7384         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
7385         \def\currentsectionlevel{\omdoc@subsection@kw}
7386       \or
7387         \stepcounter{subsubsection}
7388         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
7389         \def\currentsectionlevel{\omdoc@subsubsection@kw}
7390       \or
7391         \stepcounter{paragraph}
7392         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
7393         \def\currentsectionlevel{\omdoc@paragraph@kw}
7394       \else
7395         \def\_notesslideslabel{}
7396         \def\currentsectionlevel{\omdoc@paragraph@kw}
7397       \fi% end ifcase
7398       \_notesslideslabel%\sref@label@id\_notesslideslabel
7399       \quad #2%
7400     }%
7401     \vfill%
7402     \end{frame}%
7403   }
7404   \str\_if\_empty:NF \l\_document\_structure\_sfragment\_id\_str {
7405     \stex\_ref\_new\_doc\_target:n\l\_document\_structure\_sfragment\_id\_str
7406   }
7407 }{}
7408 }

```

We set up a `beamer` template for theorems like `ams` style, but without a block environment.

```

7409 \def\inserttheorembodyfont{\normalfont}
7410 %\bool_if:NF \c__notesslides_notes_bool {
7411 % \defbeamertemplate{theorem begin}{miko}
7412 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
7413 % \ifx\inserttheoremaddition\empty\else\ (\inserttheoremaddition)\fi%
7414 % \inserttheorempunctuation\inserttheorembodyfont\xspace}
7415 % \defbeamertemplate{theorem end}{miko}{}
```

and we set it as the default one.

```

7416 % \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.

```

7417 % \expandafter\def\csname Parent2\endcsname{}
7418 %}
7419
7420 \AddToHook{begindocument}{ % this does not work for some reasons
7421 \setbeamertemplate{theorems}[ams style]
7422 }
7423 \bool_if:NT \c__notesslides_notes_bool {
7424 \renewenvironment{columns}[1][]{%
7425 \par\noindent%
7426 \begin{minipage}%
7427 \slidewidth\centering\leavevmode%
7428 }{%
7429 \end{minipage}\par\noindent%
7430 }%
7431 \newsavebox\columnbox%
7432 \renewenvironment<>{column}[2][]{%
7433 \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
7434 }{%
7435 \end{minipage}\end{lrbox}\usebox\columnbox%
7436 }%
7437 }
7438 \bool_if:NTF \c__notesslides_noproblems_bool {
7439 \newenvironment{problems}{}{}
7440 }{
7441 \excludacomment{problems}
7442 }
```

## 38.7 Excursions

**\excursion** The excursion macros are very simple, we define a new internal macro `\excursionref` and use it in `\excursion`, which is just an `\inputref` that checks if the new macro is defined before formatting the file in the argument.

```

7443 \gdef\printexcursions{}
7444 \newcommand\excursionref[2]{% label, text
7445 \bool_if:NT \c__notesslides_notes_bool {
7446 \begin{sparagraph}[title=Excursion]
7447 #2 \sref[fallback=the appendix]{#1}.
7448 \end{sparagraph}}
```

```

7449 }
7450 }
7451 \newcommand\activate@excursion[2][ ]{
7452   \gappto\printexcursions{\inputref{#1}{#2}}
7453 }
7454 \newcommand\excursion[4][ ]{% repos, label, path, text
7455   \bool_if:NT \c__notesslides_notes_bool {
7456     \activate@excursion{#1}{#3}\excursionref{#2}{#4}
7457   }
7458 }

```

(End definition for \excursion. This function is documented on page 55.)

### \excursiongroup

```

7459 \keys_define:nn{notesslides / excursiongroup }{
7460   id          .str_set_x:N = \l__notesslides_excursion_id_str,
7461   intro       .tl_set:N    = \l__notesslides_excursion_intro_tl,
7462   mhrepos     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
7463 }
7464 \cs_new_protected:Nn \__notesslides_excursion_args:n {
7465   \tl_clear:N \l__notesslides_excursion_intro_tl
7466   \str_clear:N \l__notesslides_excursion_id_str
7467   \str_clear:N \l__notesslides_excursion_mhrepos_str
7468   \keys_set:nn {notesslides / excursiongroup }{ #1 }
7469 }
7470 \newcommand\excursiongroup[1][ ]{
7471   \__notesslides_excursion_args:n{ #1 }
7472   \ifdefempty\printexcursions{}% only if there are excursions
7473   {\begin{note}
7474     \begin{sfragment}[#1]{Excursions}%
7475     \ifdefempty\l__notesslides_excursion_intro_tl{\{
7476       \inputref[\l__notesslides_excursion_mhrepos_str]{
7477         \l__notesslides_excursion_intro_tl
7478       }
7479     }
7480     \printexcursions%
7481     \end{sfragment}
7482   }\end{note}}
7483 }
7484 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{\fi
7485 \</package>

```

(End definition for \excursiongroup. This function is documented on page 56.)

## Chapter 39

# The Implementation

### 39.1 Package Options

The first step is to declare (a few) package options that handle whether certain information is printed or not. They all come with their own conditionals that are set by the options.

```
7486 <*package>
7487 <@@=problems>
7488 \ProvidesExplPackage{problem}{2022/02/26}{3.0.1}{Semantic Markup for Problems}
7489 \RequirePackage{l3keys2e,stex}
7490
7491 \keys_define:nn { problem / pkg }{
7492   notes      .default:n    = { true },
7493   notes      .bool_set:N   = \c__problems_notes_bool,
7494   gnotes     .default:n    = { true },
7495   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
7496   hints      .default:n    = { true },
7497   hints      .bool_set:N   = \c__problems_hints_bool,
7498   solutions  .default:n    = { true },
7499   solutions  .bool_set:N   = \c__problems_solutions_bool,
7500   pts        .default:n    = { true },
7501   pts        .bool_set:N   = \c__problems_pts_bool,
7502   min        .default:n    = { true },
7503   min        .bool_set:N   = \c__problems_min_bool,
7504   boxed      .default:n    = { true },
7505   boxed      .bool_set:N   = \c__problems_boxed_bool,
7506   unknown    .code:n       = {}
7507 }
7508 \newif\ifsolutions
7509
7510 \ProcessKeysOptions{ problem / pkg }
7511 \bool_if:NTF \c__problems_solutions_bool {
7512   \solutionstrue
7513 }{
7514   \solutionsfalse
7515 }
```

Then we make sure that the necessary packages are loaded (in the right versions).

```
7516 \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.

```
7517 \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.

```
7518 \def\prob@problem@kw{Problem}
7519 \def\prob@solution@kw{Solution}
7520 \def\prob@hint@kw{Hint}
7521 \def\prob@note@kw{Note}
7522 \def\prob@gnote@kw{Grading}
7523 \def\prob@pt@kw{pt}
7524 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
7525 \AddToHook{begindocument}{
7526   \ltx@ifpackageloaded{babel}{
7527     \makeatletter
7528     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7529     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7530       \input{problem-ngerman.ldf}
7531     }
7532     \clist_if_in:NnT \l_tmpa_clist {finnish}{
7533       \input{problem-finnish.ldf}
7534     }
7535     \clist_if_in:NnT \l_tmpa_clist {french}{
7536       \input{problem-french.ldf}
7537     }
7538     \clist_if_in:NnT \l_tmpa_clist {russian}{
7539       \input{problem-russian.ldf}
7540     }
7541     \makeatother
7542   }{}
7543 }
```

## 39.2 Problems and Solutions

We now prepare the KeyVal support for problems. The key macros just set appropriate internal macros.

```
7544 \keys_define:nn{ problem / problem }{
7545   id      .str_set_x:N = \l__problems_prob_id_str,
7546   pts     .tl_set:N    = \l__problems_prob_pts_tl,
7547   min     .tl_set:N    = \l__problems_prob_min_tl,
7548   title   .tl_set:N    = \l__problems_prob_title_tl,
7549   type    .tl_set:N    = \l__problems_prob_type_tl,
7550   imports .tl_set:N    = \l__problems_prob_imports_tl,
7551   name    .str_set_x:N = \l__problems_prob_name_str,
7552   refnum  .int_set:N   = \l__problems_prob_refnum_int
```

```

7553 }
7554 \cs_new_protected:Nn \__problems_prob_args:n {
7555   \str_clear:N \l__problems_prob_id_str
7556   \str_clear:N \l__problems_prob_name_str
7557   \tl_clear:N \l__problems_prob_pts_tl
7558   \tl_clear:N \l__problems_prob_min_tl
7559   \tl_clear:N \l__problems_prob_title_tl
7560   \tl_clear:N \l__problems_prob_type_tl
7561   \tl_clear:N \l__problems_prob_imports_tl
7562   \int_zero_new:N \l__problems_prob_refnum_int
7563   \keys_set:nn { problem / problem }{ #1 }
7564   \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
7565     \let\l__problems_prob_refnum_int\undefined
7566   }
7567 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

7568 \newcounter{problem}[section]
7569 \newcommand\numberproblemsin[1]{\@addtoreset{problem}{#1}}

```

(End definition for `\numberproblemsin`. This function is documented on page ??.)

`\prob@label` We provide the macro `\prob@label` to redefine later to get context involved.

```

7570 \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

```

7571 \newcommand\prob@number{
7572   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
7573     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
7574   }{
7575     \int_if_exist:NTF \l__problems_prob_refnum_int {
7576       \prob@label{\int_use:N \l__problems_prob_refnum_int }
7577     }{
7578       \prob@label\theproblem
7579     }
7580   }
7581 }

```

(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.

```

7582 \newcommand\prob@title[3]{%
7583   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
7584     #2 \l__problems_inclprob_title_tl #3
7585   }{
7586     \tl_if_exist:NTF \l__problems_prob_title_tl {
7587       #2 \l__problems_prob_title_tl #3
7588     }{
7589       #1

```

```

7590     }
7591   }
7592 }

```

(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.

```

7593 \def\prob@heading{
7594   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
7595   %\sref@label@id{\prob@problem@kw~\prob@number}{~}
7596 }

```

(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`

```

7597 \newenvironment{sproblem}[1][{}]{
7598   \__problems_prob_args:n{#1}%\sref@target%
7599   \@in@omtexttrue% we are in a statement (for inline definitions)
7600   \stepcounter{problem}\record@problem
7601   \def\current@section@level{\prob@problem@kw}
7602
7603   \str_if_empty:NT \l__problems_prob_name_str {
7604     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
7605     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
7606     \seq_get_left:NN \l_tmpa_seq \l__problems_prob_name_str
7607   }
7608
7609   \stex_if_do_html:T{
7610     \tl_if_empty:NF \l__problems_prob_title_tl {
7611       \exp_args:No \stex_document_title:n \l__problems_prob_title_tl
7612     }
7613   }
7614
7615   \exp_args:Nno\stex_module_setup:nn{type=problem}\l__problems_prob_name_str
7616
7617   \stex_reactivate_macro:N \STEXexport
7618   \stex_reactivate_macro:N \importmodule
7619   \stex_reactivate_macro:N \symdecl
7620   \stex_reactivate_macro:N \notation
7621   \stex_reactivate_macro:N \symdef
7622
7623   \stex_if_do_html:T{
7624     \begin{stex_annotate_env} {problem} {
7625       \l_stex_module_ns_str ? \l_stex_module_name_str
7626     }
7627
7628     \stex_annotate_invisible:nnn{header}{} {
7629       \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}

```

```

7630     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
7631     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
7632         \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
7633     }
7634 }
7635 }
7636
7637 \stex_csl_to_imports:No \importmodule \l__problems_prob_imports_tl
7638
7639
7640 \tl_if_exist:NTF \l__problems_inclprob_type_tl {
7641     \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
7642 }{
7643     \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
7644 }
7645 \str_if_exist:NTF \l__problems_inclprob_id_str {
7646     \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
7647 }{
7648     \str_set_eq:NN \sproblemid \l__problems_prob_id_str
7649 }
7650
7651
7652 \stex_if_smsmode:F {
7653     \clist_set:No \l_tmpa_clist \sproblemtype
7654     \tl_clear:N \l_tmpa_tl
7655     \clist_map_inline:Nn \l_tmpa_clist {
7656         \tl_if_exist:cT {__problems_sproblem_##1_start:}{
7657             \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_start:}}
7658         }
7659     }
7660     \tl_if_empty:NTF \l_tmpa_tl {
7661         \__problems_sproblem_start:
7662     }{
7663         \l_tmpa_tl
7664     }
7665 }
7666 \stex_ref_new_doc_target:n \sproblemid
7667 \stex_smsmode_do:
7668 }{
7669     \__stex_modules_end_module:
7670     \stex_if_smsmode:F{
7671         \clist_set:No \l_tmpa_clist \sproblemtype
7672         \tl_clear:N \l_tmpa_tl
7673         \clist_map_inline:Nn \l_tmpa_clist {
7674             \tl_if_exist:cT {__problems_sproblem_##1_end:}{
7675                 \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_end:}}
7676             }
7677         }
7678         \tl_if_empty:NTF \l_tmpa_tl {
7679             \__problems_sproblem_end:
7680         }{
7681             \l_tmpa_tl
7682         }
7683     }

```



```

7684 \stex_if_do_html:T{
7685   \end{stex_annotate_env}
7686 }
7687
7688 \smallskip
7689 }
7690
7691 \seq_put_right:Nx\g_stex_smsmode_allowedenvs_seq{\tl_to_str:n{sproblem}}
7692
7693
7694
7695 \cs_new_protected:Nn \__problems_sproblem_start: {
7696   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
7697 }
7698 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
7699
7700 \newcommand\stexpatchproblem[3][] {
7701   \str_set:Nx \l_tmpa_str{ #1 }
7702   \str_if_empty:NTF \l_tmpa_str {
7703     \tl_set:Nn \__problems_sproblem_start: { #2 }
7704     \tl_set:Nn \__problems_sproblem_end: { #3 }
7705   }{
7706     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
7707     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
7708   }
7709 }
7710
7711
7712 \bool_if:NT \c__problems_boxed_bool {
7713   \surroundwithmdframed{problem}
7714 }

```

**\record@problem** This macro records information about the problems in the \*.aux file.

```

7715 \def\record@problem{
7716   \protected@write\@auxout{}
7717   {
7718     \string\@problem{\prob@number}
7719     {
7720       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7721         \l__problems_inclprob_pts_tl
7722       }{
7723         \l__problems_prob_pts_tl
7724       }
7725     }%
7726     {
7727       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7728         \l__problems_inclprob_min_tl
7729       }{
7730         \l__problems_prob_min_tl
7731       }
7732     }
7733   }
7734 }

```

(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).

```
7735 \def\@problem#1#2#3{}
```

(End definition for `\@problem`. This function is documented on page ??.)

`solution` The `solution` environment is similar to the `problem` environment, only that it is independent of the boxed mode. It also has it's own keys that we need to define first.

```
7736 \keys_define:nn { problem / solution }{
7737   id          .str_set_x:N = \l__problems_solution_id_str ,
7738   for         .tl_set:N   = \l__problems_solution_for_tl ,
7739   height      .dim_set:N  = \l__problems_solution_height_dim ,
7740   creators    .clist_set:N = \l__problems_solution_creators_clist ,
7741   contributors .clist_set:N = \l__problems_solution_contributors_clist ,
7742   srccite     .tl_set:N   = \l__problems_solution_srccite_tl
7743 }
7744 \cs_new_protected:Nn \__problems_solution_args:n {
7745   \str_clear:N \l__problems_solution_id_str
7746   \tl_clear:N \l__problems_solution_for_tl
7747   \tl_clear:N \l__problems_solution_srccite_tl
7748   \clist_clear:N \l__problems_solution_creators_clist
7749   \clist_clear:N \l__problems_solution_contributors_clist
7750   \dim_zero:N \l__problems_solution_height_dim
7751   \keys_set:nn { problem / solution }{ #1 }
7752 }
```

the next step is to define a helper macro that does what is needed to start a solution.

```
7753 \newcommand\@startsolution[1][]{
7754   \__problems_solution_args:n { #1 }
7755   \@in@omtexttrue% we are in a statement.
7756   \bool_if:NF \c__problems_boxed_bool { \hrule }
7757   \smallskip\noindent
7758   {\textbf{\prob@solution@kw :}\enspace}
7759   \begin{small}
7760   \def\current@section@level{\prob@solution@kw}
7761   \ignorespacesandpars
7762 }
```

`\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.

```
7763 \box_new:N \l__problems_solution_box
7764 \newenvironment{solution}[1][]{
7765   \stex_html_backend:TF{
7766     \stex_if_do_html:T{
7767       \begin{stex_annotate_env}{solution}{}}
7768     }
7769   }{
7770     \setbox\l__problems_solution_box\vbox\bgroup
7771     \par\smallskip\hrule\smallskip
7772     \noindent\textbf{Solution:}~
7773   }
7774 }{
7775   \stex_html_backend:TF{
```

```

7776 \stex_if_do_html:T{
7777 \end{stex_annotate_env}
7778 }
7779 }{
7780 \smallskip\hrule
7781 \egroup
7782 \bool_if:NT \c__problems_solutions_bool {
7783 \box\l__problems_solution_box
7784 }
7785 }
7786 }
7787
7788 \newcommand\startsolutions{
7789 \bool_set_true:N \c__problems_solutions_bool
7790 % \specialcomment{solution}{\@startsolution}{
7791 % \bool_if:NF \c__problems_boxed_bool {
7792 % \hrule\medskip
7793 % }
7794 % \end{small}}%
7795 % }
7796 % \bool_if:NT \c__problems_boxed_bool {
7797 % \surroundwithmdframed{solution}
7798 % }
7799 }

```

(End definition for \startsolutions. This function is documented on page 57.)

## **\stopsolutions**

```

7800 \newcommand\stopsolutions{\bool_set_false:N \c__problems_solutions_bool}%\excludecomment{sol

```

(End definition for \stopsolutions. This function is documented on page 57.)

so it only remains to start/stop solutions depending on what option was specified.

```

7801 \ifsolutions
7802 \startsolutions
7803 \else
7804 \stopsolutions
7805 \fi

```

## **exnote**

```

7806 \bool_if:NTF \c__problems_notes_bool {
7807 \newenvironment{exnote}[1][{}]{
7808 \par\smallskip\hrule\smallskip
7809 \noindent\textbf{\prob@note@kw :~ }\small
7810 }{
7811 \smallskip\hrule
7812 }
7813 }{
7814 \excludecomment{exnote}
7815 }

```

## **hint**

```

7816 \bool_if:NTF \c__problems_notes_bool {
7817 \newenvironment{hint}[1][{}]{
7818 \par\smallskip\hrule\smallskip

```

```

7819 \noindent\textbf{\prob@hint@kw :~ }\small
7820 }{
7821 \smallskip\hrule
7822 }
7823 \newenvironment{exhint}[1][]{
7824 \par\smallskip\hrule\smallskip
7825 \noindent\textbf{\prob@hint@kw :~ }\small
7826 }{
7827 \smallskip\hrule
7828 }
7829 }{
7830 \excludecomment{hint}
7831 \excludecomment{exhint}
7832 }

```

gnote

```

7833 \bool_if:NTF \c__problems_notes_bool {
7834 \newenvironment{gnote}[1][]{
7835 \par\smallskip\hrule\smallskip
7836 \noindent\textbf{\prob@gnote@kw :~ }\small
7837 }{
7838 \smallskip\hrule
7839 }
7840 }{
7841 \excludecomment{gnote}
7842 }

```

### 39.3 Multiple Choice Blocks

EdN:21

mcb 21

```

7843 \newenvironment{mcb}{
7844 \begin{enumerate}
7845 }{
7846 \end{enumerate}
7847 }

```

we define the keys for the mcb macro

```

7848 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
7849 \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
7850 \bool_set_true:N #1
7851 }{
7852 \bool_set_false:N #1
7853 }
7854 }
7855 \keys_define:nn { problem / mcb }{
7856 id .str_set:x:N = \l__problems_mcc_id_str ,
7857 feedback .tl_set:N = \l__problems_mcc_feedback_tl ,
7858 T .default:n = { false } ,
7859 T .bool_set:N = \l__problems_mcc_t_bool ,
7860 F .default:n = { false } ,
7861 F .bool_set:N = \l__problems_mcc_f_bool ,

```

---

<sup>21</sup>EdNOTE: MK: maybe import something better here from a dedicated MC package

```

7862 Ttext      .tl_set:N      = \l__problems_mcc_Ttext_str ,
7863 Ftext      .tl_set:N      = \l__problems_mcc_Ftext_str
7864 }
7865 \cs_new_protected:Nn \l__problems_mcc_args:n {
7866   \str_clear:N \l__problems_mcc_id_str
7867   \tl_clear:N \l__problems_mcc_feedback_tl
7868   \bool_set_false:N \l__problems_mcc_t_bool
7869   \bool_set_false:N \l__problems_mcc_f_bool
7870   \tl_clear:N \l__problems_mcc_Ttext_tl
7871   \tl_clear:N \l__problems_mcc_Ftext_tl
7872   \str_clear:N \l__problems_mcc_id_str
7873   \keys_set:nn { problem / mcc }{ #1 }
7874 }

```

**\mcc**

```

7875 \def\mccTrueText{\textbf{(true)~}}
7876 \def\mccFalseText{\textbf{(false)~}}
7877 \newcommand\mcc[2][]{}
7878   \l__problems_mcc_args:n{ #1 }
7879   \item[{$\Box$}] #2
7880   \ifsolutions
7881     \\\
7882     \bool_if:NT \l__problems_mcc_t_bool {
7883       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccTrueText\l__problems_mcc_Ttext_tl
7884     }
7885     \bool_if:NT \l__problems_mcc_f_bool {
7886       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccFalseText\l__problems_mcc_Ftext_tl
7887     }
7888     \tl_if_empty:NF \l__problems_mcc_feedback_tl {
7889       \emph{(\l__problems_mcc_feedback_tl)}
7890     }
7891   \fi
7892 } %solutions

```

(End definition for \mcc. This function is documented on page 58.)

## 39.4 Including Problems

**\includeproblem** The \includeproblem command is essentially a glorified \input statement, it sets some internal macros first that overwrite the local points. Importantly, it resets the `inclprob` keys after the input.

```

7893
7894 \keys_define:nn{ problem / inclproblem }{
7895   id      .str_set_x:N      = \l__problems_inclprob_id_str,
7896   pts     .tl_set:N         = \l__problems_inclprob_pts_tl,
7897   min     .tl_set:N         = \l__problems_inclprob_min_tl,
7898   title   .tl_set:N         = \l__problems_inclprob_title_tl,
7899   refnum  .int_set:N        = \l__problems_inclprob_refnum_int,
7900   type    .tl_set:N         = \l__problems_inclprob_type_tl,
7901   mhrepos .str_set_x:N      = \l__problems_inclprob_mhrepos_str
7902 }
7903 \cs_new_protected:Nn \__problems_inclprob_args:n {
7904   \str_clear:N \l__problems_prob_id_str

```

```

7905 \tl_clear:N \l__problems_inclprob_pts_tl
7906 \tl_clear:N \l__problems_inclprob_min_tl
7907 \tl_clear:N \l__problems_inclprob_title_tl
7908 \tl_clear:N \l__problems_inclprob_type_tl
7909 \int_zero_new:N \l__problems_inclprob_refnum_int
7910 \str_clear:N \l__problems_inclprob_mhrepos_str
7911 \keys_set:nn { problem / inclproblem }{ #1 }
7912 \tl_if_empty:NT \l__problems_inclprob_pts_tl {
7913   \let\l__problems_inclprob_pts_tl\undefined
7914 }
7915 \tl_if_empty:NT \l__problems_inclprob_min_tl {
7916   \let\l__problems_inclprob_min_tl\undefined
7917 }
7918 \tl_if_empty:NT \l__problems_inclprob_title_tl {
7919   \let\l__problems_inclprob_title_tl\undefined
7920 }
7921 \tl_if_empty:NT \l__problems_inclprob_type_tl {
7922   \let\l__problems_inclprob_type_tl\undefined
7923 }
7924 \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
7925   \let\l__problems_inclprob_refnum_int\undefined
7926 }
7927 }
7928
7929 \cs_new_protected:Nn \__problems_inclprob_clear: {
7930   \let\l__problems_inclprob_id_str\undefined
7931   \let\l__problems_inclprob_pts_tl\undefined
7932   \let\l__problems_inclprob_min_tl\undefined
7933   \let\l__problems_inclprob_title_tl\undefined
7934   \let\l__problems_inclprob_type_tl\undefined
7935   \let\l__problems_inclprob_refnum_int\undefined
7936   \let\l__problems_inclprob_mhrepos_str\undefined
7937 }
7938 \__problems_inclprob_clear:
7939
7940 \newcommand\includeproblem[2][ ]{
7941   \__problems_inclprob_args:n{ #1 }
7942   \exp_args:No \stex_in_repository:nn\l__problems_inclprob_mhrepos_str{
7943     \stex_html_backend:TF {
7944       \str_clear:N \l_tmpa_str
7945       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
7946         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
7947       }
7948       \stex_annotate_invisible:nnn{includeproblem}{
7949         \l_tmpa_str / #2
7950       }{}
7951     }{
7952       \begingroup
7953         \inputreftrue
7954         \tl_if_empty:nTF{ ##1 }{
7955           \input{#2}
7956         }{
7957           \input{ \c_stex_mathhub_str / ##1 / source / #2 }
7958         }

```

```

7959     \endgroup
7960   }
7961 }
7962 \__problems_inclprob_clear:
7963 }

```

(End definition for `\includeproblem`. This function is documented on page 59.)

## 39.5 Reporting Metadata

For messages it is OK to have them in English as the whole documentation is, and we can therefore assume authors can deal with it.

```

7964 \AddToHook{enddocument}{
7965   \bool_if:NT \c__problems_pts_bool {
7966     \message{Total:~\arabic{pts}~points}
7967   }
7968   \bool_if:NT \c__problems_min_bool {
7969     \message{Total:~\arabic{min}~minutes}
7970   }
7971 }

```

The margin pars are reader-visible, so we need to translate

```

7972 \def\pts#1{
7973   \bool_if:NT \c__problems_pts_bool {
7974     \marginpar{#1~\prob@pt@kw}
7975   }
7976 }
7977 \def\min#1{
7978   \bool_if:NT \c__problems_min_bool {
7979     \marginpar{#1~\prob@min@kw}
7980   }
7981 }

```

`\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.

```

7982 \newcounter{pts}
7983 \def\show@pts{
7984   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7985     \bool_if:NT \c__problems_pts_bool {
7986       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
7987       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
7988     }
7989   }{
7990     \tl_if_exist:NT \l__problems_prob_pts_tl {
7991       \bool_if:NT \c__problems_pts_bool {
7992         \tl_if_empty:NTF \l__problems_prob_pts_tl{
7993           \tl_set:Nn \l__problems_prob_pts_tl {0}
7994         }
7995         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
7996         \addtocounter{pts}{\l__problems_prob_pts_tl}
7997       }
7998     }

```

```

7999   }
8000 }

```

(End definition for \show@pts. This function is documented on page ??.)  
and now the same for the minutes

\show@min

```

8001 \newcounter{min}
8002 \def\show@min{
8003   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
8004     \bool_if:NT \c__problems_min_bool {
8005       \marginpar{\l__problems_inclprob_pts_tl\ min}
8006       \addtocounter{min}{\l__problems_inclprob_min_tl}
8007     }
8008   }{
8009     \tl_if_exist:NT \l__problems_prob_min_tl {
8010       \bool_if:NT \c__problems_min_bool {
8011         \tl_if_empty:NT\l__problems_prob_min_tl{
8012           \tl_set:Nn \l__problems_prob_min_tl {0}
8013         }
8014         \marginpar{\l__problems_prob_min_tl\ min}
8015         \addtocounter{min}{\l__problems_prob_min_tl}
8016       }
8017     }
8018   }
8019 }
8020 \</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.

```
8021 \*package>
8022 \ProvidesExplPackage{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
8023 \RequirePackage{13keys2e}
8024
8025 \newif\iftest\testfalse
8026 \DeclareOption{test}{\testtrue}
8027 \newif\ifmultiple\multiplefalse
8028 \DeclareOption{multiple}{\multipletrue}
8029 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
8030 \ProcessOptions
```

Then we make sure that the necessary packages are loaded (in the right versions).

```
8031 \RequirePackage{keyval}[1997/11/10]
8032 \RequirePackage{problem}
```

`\hwexam@*@kw` For multilinguality, we define internal macros for keywords that can be specialized in `*.ldf` files.

```
8033 \newcommand\hwexam@assignment@kw{Assignment}
8034 \newcommand\hwexam@given@kw{Given}
8035 \newcommand\hwexam@due@kw{Due}
8036 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
8037 blank~for~extra~space}
8038 \def\hwexam@minutes@kw{minutes}
8039 \newcommand\correction@probs@kw{prob.}
8040 \newcommand\correction@pts@kw{total}
8041 \newcommand\correction@reached@kw{reached}
8042 \newcommand\correction@sum@kw{Sum}
8043 \newcommand\correction@grade@kw{grade}
8044 \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

```

8045 \AddToHook{begindocument}{
8046 \ltx@ifpackageloaded{babel}{
8047 \makeatletter
8048 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
8049 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
8050 \input{hwexam-ngerman.ldf}
8051 }
8052 \clist_if_in:NnT \l_tmpa_clist {finnish}{
8053 \input{hwexam-finnish.ldf}
8054 }
8055 \clist_if_in:NnT \l_tmpa_clist {french}{
8056 \input{hwexam-french.ldf}
8057 }
8058 \clist_if_in:NnT \l_tmpa_clist {russian}{
8059 \input{hwexam-russian.ldf}
8060 }
8061 \makeatother
8062 }{}
8063 }
8064

```

## 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.

```

8065 \newcounter{assignment}
8066 %\numberproblemsin{assignment}

We will prepare the keyval support for the assignment environment.

8067 \keys_define:nn { hwexam / assignment } {
8068 id .str_set:N = \l_@@_assign_id_str,
8069 number .int_set:N = \l_@@_assign_number_int,
8070 title .tl_set:N = \l_@@_assign_title_tl,
8071 type .tl_set:N = \l_@@_assign_type_tl,
8072 given .tl_set:N = \l_@@_assign_given_tl,
8073 due .tl_set:N = \l_@@_assign_due_tl,
8074 loadmodules .code:n = {
8075 \bool_set_true:N \l_@@_assign_loadmodules_bool
8076 }
8077 }
8078 \cs_new_protected:Nn \_@@_assignment_args:n {
8079 \str_clear:N \l_@@_assign_id_str
8080 \int_set:Nn \l_@@_assign_number_int {-1}
8081 \tl_clear:N \l_@@_assign_title_tl
8082 \tl_clear:N \l_@@_assign_type_tl
8083 \tl_clear:N \l_@@_assign_given_tl
8084 \tl_clear:N \l_@@_assign_due_tl
8085 \bool_set_false:N \l_@@_assign_loadmodules_bool
8086 \keys_set:nn { hwexam / assignment }{ #1 }
8087 }

```

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.

```

8088 \newcommand\given@due[2]{
8089 \bool_lazy_all:nF {
8090 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
8091 { \tl_if_empty_p:V \l_@@_assign_given_tl }
8092 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
8093 { \tl_if_empty_p:V \l_@@_assign_due_tl }
8094 }{ #1 }
8095
8096 \tl_if_empty:NTF \l_@@_inclasssign_given_tl {
8097 \tl_if_empty:NF \l_@@_assign_given_tl {
8098 \hwexam@given@kw\xspace\l_@@_assign_given_tl
8099 }
8100 }{
8101 \hwexam@given@kw\xspace\l_@@_inclasssign_given_tl
8102 }
8103
8104 \bool_lazy_or:nnF {
8105 \bool_lazy_and_p:nn {
8106 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
8107 }{
8108 \tl_if_empty_p:V \l_@@_assign_due_tl
8109 }
8110 }{
8111 \bool_lazy_and_p:nn {
8112 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
8113 }{
8114 \tl_if_empty_p:V \l_@@_assign_due_tl
8115 }
8116 }{ ,~ }
8117
8118 \tl_if_empty:NTF \l_@@_inclasssign_due_tl {
8119 \tl_if_empty:NF \l_@@_assign_due_tl {
8120 \hwexam@due@kw\xspace \l_@@_assign_due_tl
8121 }
8122 }{
8123 \hwexam@due@kw\xspace \l_@@_inclasssign_due_tl
8124 }
8125
8126 \bool_lazy_all:nF {
8127 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
8128 { \tl_if_empty_p:V \l_@@_assign_given_tl }
8129 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
8130 { \tl_if_empty_p:V \l_@@_assign_due_tl }
8131 }{ #2 }
8132 }

```

`\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.

```

8133 \newcommand\assignment@title[3]{
8134 \tl_if_empty:NTF \l_@@_inclasssign_title_tl {
8135 \tl_if_empty:NTF \l_@@_assign_title_tl {
8136 #1
8137 }{
8138 #2\l_@@_assign_title_tl#3
8139 }
8140 }{
8141 #2\l_@@_inclasssign_title_tl#3
8142 }
8143 }

```

(End definition for \assignment@title. This function is documented on page ??.)

**\assignment@number** Like \assignment@title only for the number, and no around part.

```

8144 \newcommand\assignment@number{
8145 \int_compare:nNnTF \l_@@_inclasssign_number_int = {-1} {
8146 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8147 \arabic{assignment}
8148 } {
8149 \int_use:N \l_@@_assign_number_int
8150 }
8151 }{
8152 \int_use:N \l_@@_inclasssign_number_int
8153 }
8154 }

```

(End definition for \assignment@number. This function is documented on page ??.)

With them, we can define the central **assignment** environment. This has two forms (separated by \ifmultiple) in one we make a title block for an assignment sheet, and in the other we make a section heading and add it to the table of contents. We first define an assignment counter

**assignment** For the assignment environment we delegate the work to the @assignment environment that depends on whether multiple option is given.

```

8155 \newenvironment{assignment}[1][]{
8156 \_@@_assignment_args:n { #1 }
8157 %\sref@target
8158 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8159 \global\stepcounter{assignment}
8160 }{
8161 \global\setcounter{assignment}{\int_use:N\l_@@_assign_number_int}
8162 }
8163 \setcounter{problem}{0}
8164 \renewcommand\prob@label[1]{\assignment@number.##1}
8165 \def\current@section@level{\document@hwexamtype}
8166 %\sref@label{id{\document@hwexamtype \thesection}
8167 \begin{@assignment}
8168 }{
8169 \end{@assignment}
8170 }

```

In the multi-assignment case we just use the omdoc environment for suitable sectioning.

```

8171 \def\ass@title{
8172 {\protect\document@hwexamtype}\arabic{assignment}
8173 \assignment@title{\;(\;)\;}\; -- \given@due{}}
8174 }
8175 \ifmultiple
8176 \newenvironment{@assignment}{
8177 \bool_if:NTF \l_@@_assign_loadmodules_bool {
8178 \begin{sfragment}[loadmodules]{\ass@title}
8179 }{
8180 \begin{sfragment}{\ass@title}
8181 }
8182 }{
8183 \end{sfragment}
8184 }

```

for the single-page case we make a title block from the same components.

```

8185 \else
8186 \newenvironment{@assignment}{
8187 \begin{center}\bf
8188 \Large@title\strut\
8189 \document@hwexamtype\arabic{assignment}\assignment@title{\;(\;)\;}\;{\\}
8190 \large\given@due{--\;}\;{\\;--}
8191 \end{center}
8192 }{}
8193 \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.

```

8194 \keys_define:nn { hwexam / inclassignment } {
8195 %id .str_set_x:N = \l_@@_assign_id_str,
8196 number .int_set:N = \l_@@_inclassign_number_int,
8197 title .tl_set:N = \l_@@_inclassign_title_tl,
8198 type .tl_set:N = \l_@@_inclassign_type_tl,
8199 given .tl_set:N = \l_@@_inclassign_given_tl,
8200 due .tl_set:N = \l_@@_inclassign_due_tl,
8201 mhrepos .str_set_x:N = \l_@@_inclassign_mhrepos_str
8202 }
8203 \cs_new_protected:Nn \l_@@_inclassignment_args:n {
8204 \int_set:Nn \l_@@_inclassign_number_int {-1}
8205 \tl_clear:N \l_@@_inclassign_title_tl
8206 \tl_clear:N \l_@@_inclassign_type_tl
8207 \tl_clear:N \l_@@_inclassign_given_tl
8208 \tl_clear:N \l_@@_inclassign_due_tl
8209 \str_clear:N \l_@@_inclassign_mhrepos_str
8210 \keys_set:nn { hwexam / inclassignment }{ #1 }
8211 }
8212 \l_@@_inclassignment_args:n {}
8213
8214 \newcommand\inputassignment[2][{}]{

```

```

8215 \_@@_inclassassignment_args:n { #1 }
8216 \str_if_empty:NTF \l_@@_inclasssign_mhrepos_str {
8217 \input{#2}
8218 }{
8219 \stex_in_repository:nn{\l_@@_inclasssign_mhrepos_str}{
8220 \input{\mhpath{\l_@@_inclasssign_mhrepos_str}{#2}}
8221 }
8222 }
8223 \_@@_inclassassignment_args:n {}
8224 }
8225 \newcommand\includeassignment[2][]{
8226 \newpage
8227 \inputassignment[#1]{#2}
8228 }

```

(End definition for \in\*assignment. This function is documented on page ??.)

## 40.4 Typesetting Exams

\quizheading

```

8229 \ExplSyntaxOff
8230 \newcommand\quizheading[1]{%
8231 \def\@tas{#1}%
8232 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
8233 \ifx\@tas\@empty\else%
8234 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
8235 \fi%
8236 }
8237 \ExplSyntaxOn

```

(End definition for \quizheading. This function is documented on page ??.)

\testheading

```

8238
8239 \def\hwexamheader{\input{hwexam-default.header}}
8240
8241 \def\hwexamminutes{
8242 \tl_if_empty:NTF \testheading@duration {
8243 {\testheading@min}~\hwexam@minutes@kw
8244 }{
8245 \testheading@duration
8246 }
8247 }
8248
8249 \keys_define:nn { hwexam / testheading } {
8250 min .tl_set:N = \testheading@min,
8251 duration .tl_set:N = \testheading@duration,
8252 reqpts .tl_set:N = \testheading@reqpts,
8253 tools .tl_set:N = \testheading@tools
8254 }
8255 \cs_new_protected:Nn \_@@_testheading_args:n {
8256 \tl_clear:N \testheading@min
8257 \tl_clear:N \testheading@duration

```

```

8258 \tl_clear:N \testheading@reqpts
8259 \tl_clear:N \testheading@tools
8260 \keys_set:nn { hwexam / testheading }{ #1 }
8261 }
8262 \newenvironment{testheading}[1][ ]{
8263 \_@@_testheading_args:n{ #1 }
8264 \newcount\check@time\check@time=\testheading@min
8265 \advance\check@time by -\theassignment@totalmin
8266 \newif\if@bonuspoints
8267 \tl_if_empty:NTF \testheading@reqpts {
8268 \@bonuspointsfalse
8269 }{
8270 \newcount\bonus@pts
8271 \bonus@pts=\theassignment@totalpts
8272 \advance\bonus@pts by -\testheading@reqpts
8273 \edef\bonus@pts{\the\bonus@pts}
8274 \@bonuspointstrue
8275 }
8276 \edef\check@time{\the\check@time}
8277
8278 \makeatletter\hwexamheader\makeatother
8279 }{
8280 \newpage
8281 }

```

(End definition for \testheading. This function is documented on page ??.)

\testspace

```

8282 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

(End definition for \testspace. This function is documented on page ??.)

\testnewpage

```

8283 \newcommand\testnewpage{\iftest\newpage\fi}

```

(End definition for \testnewpage. This function is documented on page ??.)

\testemptypage

```

8284 \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.

```

8285 <@@=problems>
8286 \renewcommand\@problem[3]{
8287 \stepcounter{assignment@probs}
8288 \def\__problemspts{#2}
8289 \ifx\__problemspts\@empty\else
8290 \addtocounter{assignment@totalpts}{#2}
8291 \fi
8292 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
8293 \xdef\correction@probs{\correction@probs & #1}%
8294 \xdef\correction@pts{\correction@pts & #2}
8295 \xdef\correction@reached{\correction@reached &}

```

```

8296 }
8297 <@@=hwexam>

```

(End definition for \@problem. This function is documented on page ??.)

`\correction@table` This macro generates the correction table

```

8298 \newcounter{assignment@probs}
8299 \newcounter{assignment@totalpts}
8300 \newcounter{assignment@totalmin}
8301 \def\correction@probs{\correction@probs@kw}
8302 \def\correction@pts{\correction@pts@kw}
8303 \def\correction@reached{\correction@reached@kw}
8304 \stepcounter{assignment@probs}
8305 \newcommand\correction@table{
8306 \resizebox{\textwidth}{!}{%
8307 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
8308 &\multicolumn{\theassignment@probs}{c|}|%|
8309 {\footnotesize\correction@forgrading@kw} &\\ \hline
8310 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
8311 \correction@pts & \theassignment@totalpts & \\ \hline
8312 \correction@reached & & \[.7cm]\hline
8313 \end{tabular}}
8314 </package>

```

(End definition for \correction@table. This function is documented on page ??.)

## 40.5 Leftovers

at some point, we may want to reactivate the logos font, then we use

here we define the logos that characterize the assignment

```

\font\bierfont=../assignments/bierglas
\font\denkerfont=../assignments/denker
\font\uhrfont=../assignments/uhr
\font\warnschildfont=../assignments/achtung

\newcommand\bierglas{{\bierfont\char65}}
\newcommand\denker{{\denkerfont\char65}}
\newcommand\uhr{{\uhrfont\char65}}
\newcommand\warnschild{{\warnschildfont\char 65}}
\newcommand\hardA{\warnschild}
\newcommand\longA{\uhr}
\newcommand\thinkA{\denker}
\newcommand\discussA{\bierglas}

```



# Chapter 41

## References

- [Bus+04] Stephen Buswell et al. *The Open Math Standard, Version 2.0*. Tech. rep. The OpenMath Society, 2004. URL: <http://www.openmath.org/standard/om20>.
- [CR99] David Carlisle and Sebastian Rathz. *The graphicx package*. Part of the T<sub>E</sub>X distribution. The Comprehensive T<sub>E</sub>X Archive Network. 1999. URL: <https://www.tug.org/texlive/devsrc/Master/texmf-dist/doc/latex/graphics/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>22</sup>EDNOTE: we need an un-numbered version sfragment\*

- [SIa] *sLaTeX/sTeX-IDE*. URL: <https://github.com/slatex/sTeX-IDE> (visited on 04/22/2022).
- [SIb] *sLaTeX/stexls-vscode-plugin*. URL: <https://github.com/slatex/stexls-vscode-plugin> (visited on 04/22/2022).
- [SLS] *sLaTeX/stexls*. URL: <https://github.com/slatex/stexls> (visited on 04/22/2022).
- [ST] *sTeX – An Infrastructure for Semantic Preloading of LaTeX Documents*. URL: <https://ctan.org/pkg/stex> (visited on 04/22/2022).
- [sTeX] *sTeX: A semantic Extension of TeX/LaTeX*. URL: <https://github.com/sLaTeX/sTeX> (visited on 05/11/2020).
- [Tana] Till Tantau. *beamer – A LaTeX class for producing presentations and slides*. URL: <http://ctan.org/pkg/beamer> (visited on 01/07/2014).
- [Tanb] Till Tantau. *User Guide to the Beamer Class*. URL: <http://ctan.org/macros/latex/contrib/beamer/doc/beameruserguide.pdf>.
- [TL] *TeX Live*. URL: <http://www.tug.org/texlive/> (visited on 12/11/2012).