

# The sTeX3 Package Collection \*

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

2022-05-24

## Abstract

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

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

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

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

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

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

---

\*Version 3.1 (last revised 2022-05-24)

# Contents

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

<b>5</b>	<b>sTeX Statements</b>	<b>41</b>
5.1	Definitions, Theorems, Examples, Paragraphs . . . . .	41
<b>6</b>	<b>Highlighting and Presentation Customizations</b>	<b>48</b>
<b>7</b>	<b>Additional Packages</b>	<b>50</b>
7.1	Tikzinput: Treating TIKZ code as images . . . . .	50
7.2	Modular Document Structuring . . . . .	51
7.3	Slides and Course Notes . . . . .	53
7.4	Representing Problems and Solutions . . . . .	57
7.5	Homeworks, Quizzes and Exams . . . . .	60
<b>II</b>	<b>Documentation</b>	<b>63</b>
<b>8</b>	<b>sTeX-Basics</b>	<b>64</b>
8.1	Macros and Environments . . . . .	64
8.1.1	HTML Annotations . . . . .	64
8.1.2	Babel Languages . . . . .	65
8.1.3	Auxiliary Methods . . . . .	65
<b>9</b>	<b>sTeX-MathHub</b>	<b>66</b>
9.1	Macros and Environments . . . . .	66
9.1.1	Files, Paths, URIs . . . . .	66
9.1.2	MathHub Archives . . . . .	67
9.1.3	Using Content in Archives . . . . .	68
<b>10</b>	<b>sTeX-References</b>	<b>69</b>
10.1	Macros and Environments . . . . .	69
10.1.1	Setting Reference Targets . . . . .	69
10.1.2	Using References . . . . .	70
<b>11</b>	<b>sTeX-Modules</b>	<b>71</b>
11.1	Macros and Environments . . . . .	71
11.1.1	The <code>smodule</code> environment . . . . .	73
<b>12</b>	<b>sTeX-Module Inheritance</b>	<b>75</b>
12.1	Macros and Environments . . . . .	75
12.1.1	SMS Mode . . . . .	75
12.1.2	Imports and Inheritance . . . . .	76
<b>13</b>	<b>sTeX-Symbols</b>	<b>78</b>
13.1	Macros and Environments . . . . .	78
<b>14</b>	<b>sTeX-Terms</b>	<b>80</b>
14.1	Macros and Environments . . . . .	80
<b>15</b>	<b>sTeX-Structural Features</b>	<b>82</b>
15.1	Macros and Environments . . . . .	82
15.1.1	Structures . . . . .	82

<b>16</b>	<b><code>sTeX</code>-Statements</b>	<b>83</b>
16.1	Macros and Environments . . . . .	83
<b>17</b>	<b><code>sTeX</code>-Proofs: Structural Markup for Proofs</b>	<b>84</b>
<b>18</b>	<b><code>sTeX</code>-Metatheory</b>	<b>85</b>
18.1	Symbols . . . . .	85
<b>III</b>	<b>Extensions</b>	<b>86</b>
<b>19</b>	<b><code>Tikzinput</code>: Treating <code>TIKZ</code> code as images</b>	<b>87</b>
19.1	Macros and Environments . . . . .	87
<b>20</b>	<b><code>document-structure</code>: Semantic Markup for Open Mathematical Documents in <code>LaTeX</code></b>	<b>88</b>
<b>21</b>	<b><code>NotesSlides</code> – Slides and Course Notes</b>	<b>89</b>
<b>22</b>	<b><code>problem.sty</code>: An Infrastructure for formatting Problems</b>	<b>90</b>
<b>23</b>	<b><code>hwexam.sty/cls</code>: An Infrastructure for formatting Assignments and Exams</b>	<b>91</b>
<b>IV</b>	<b>Implementation</b>	<b>92</b>
<b>24</b>	<b><code>sTeX</code>-Basics Implementation</b>	<b>93</b>
24.1	The <code>sTeXDocument</code> Class . . . . .	93
24.2	Preliminaries . . . . .	94
24.3	Messages and logging . . . . .	94
24.4	HTML Annotations . . . . .	95
24.5	Babel Languages . . . . .	97
24.6	Persistence . . . . .	98
24.7	Auxiliary Methods . . . . .	99
<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 . . . . .	152
29.3	Variables . . . . .	160
<b>30</b>	<b>sTeX-Terms Implementation</b>	<b>168</b>
30.1	Symbol Invocations . . . . .	168
30.2	Terms . . . . .	175
30.3	Notation Components . . . . .	179
30.4	Variables . . . . .	181
30.5	Sequences . . . . .	184
<b>31</b>	<b>sTeX-Structural Features Implementation</b>	<b>185</b>
31.1	Imports with modification . . . . .	186
31.2	The feature environment . . . . .	194
31.3	Structure . . . . .	194
<b>32</b>	<b>sTeX-Statements Implementation</b>	<b>205</b>
32.1	Definitions . . . . .	205
32.2	Assertions . . . . .	211
32.3	Examples . . . . .	214
32.4	Logical Paragraphs . . . . .	217
<b>33</b>	<b>The Implementation</b>	<b>222</b>
33.1	Proofs . . . . .	222
33.2	Justifications . . . . .	233
<b>34</b>	<b>sTeX-Others Implementation</b>	<b>234</b>
<b>35</b>	<b>sTeX-Metatheory Implementation</b>	<b>236</b>
<b>36</b>	<b>Tikzinput Implementation</b>	<b>239</b>
<b>37</b>	<b>document-structure.sty Implementation</b>	<b>242</b>
37.1	Package Options . . . . .	242
37.2	Document Structure . . . . .	243
37.3	Front and Backmatter . . . . .	247
37.4	Global Variables . . . . .	249
<b>38</b>	<b>NotesSlides – Implementation</b>	<b>250</b>
38.1	Class and Package Options . . . . .	250
38.2	Notes and Slides . . . . .	252
38.3	Header and Footer Lines . . . . .	256
38.4	Frame Images . . . . .	258
38.5	Colors and Highlighting . . . . .	259
38.6	Sectioning . . . . .	260
38.7	Excursions . . . . .	263

<b>39 The Implementation</b>	<b>265</b>
39.1 Package Options . . . . .	265
39.2 Problems and Solutions . . . . .	266
39.3 Multiple Choice Blocks . . . . .	273
39.4 Including Problems . . . . .	274
39.5 Reporting Metadata . . . . .	276
<b>40 Implementation: The hwexam Package</b>	<b>278</b>
40.1 Package Options . . . . .	278
40.2 Assignments . . . . .	279
40.3 Including Assignments . . . . .	282
40.4 Typesetting Exams . . . . .	283
40.5 Leftovers . . . . .	285
<b>41 References</b>	<b>286</b>

# Part I

## Manual



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



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

# Chapter 1

## What is sTeX?

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

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

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

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

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

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



# Chapter 2

## Quickstart

### 2.1 Setup

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

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

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

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

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

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

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

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

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

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

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

### 2.1.3 sTeX Archives (Manual Setup)

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

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

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

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

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

### 2.1.4 The sTeX IDE

We are currently working on an sTeX IDE as an sTeX plugin for VScode; see [\[S1a\]](#). 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 [\[SLS; S1b\]](#).

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

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

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

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

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

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

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

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

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

```

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

```

Compiling this document with `pdflatex` should yield the output

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

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

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

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

### Remark 2.2.1:

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

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

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

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

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

---

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

---

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

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

---

```
\usemodule
```

---

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

---

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

---

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

---

```
\comp
```

---

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

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

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

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

---

`\symname`

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

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

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

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

---

`\symref`

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

---

`\define`  
`\definiendum`

The `\define{geometricSeries} ...`

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

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

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

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

### 2.2.1 OMDoc/xhtml Conversion

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

#### TODO VSCode Plugin

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

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

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

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

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

```

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

```

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

#### Remark 2.2.2:

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

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

## 2.2.2 Mmt/OMDoc Conversion

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

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

## Chapter 3

# Creating sTeX Content

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

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

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

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

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

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

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

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

### 3.1 How Knowledge is Organized in sTeX

sTeX content is organized on multiple levels:

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



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

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

## 3.2 $\text{\texttt{STeX}}$ Archives

### 3.2.1 The Local MathHub-Directory

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

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

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

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

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

Each such archive needs two subdirectories:

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

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

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

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

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

### 3.2.3 MANIFEST.MF-Files

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

```
id: smglom/calculus
source-base: http://mathhub.info/smgglom/calculus
narration-base: http://mathhub.info/smgglom/calculus
dependencies: smglom/arithmetics,smglom/sets,smglom/topology,
              smglom/mv,smglom/linear-algebra,smglom/algebra
responsible: Michael.Kohlhase@FAU.de
title: Elementary Calculus
```

teaser: Terminology for the mathematical study of change. description: desc.html
---

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

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

**source-base** or

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

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

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

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

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

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

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

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

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

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

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

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

---

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

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

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

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

---

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

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

**Remark 3.2.1:**

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

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

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

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

## 3.3 Module, Symbol and Notation Declarations

### 3.3.1 The `smodule`-Environment

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

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

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

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

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

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

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

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

#### Example 1

Input:

```

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

```

Output:

Hello World

#### `\stexpatchmodule`

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

For example:

#### Example 2

Input:

```

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

```

Output:

```

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

```

### 3.3.2 Declaring New Symbols and Notations

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

`\symdecl`

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

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

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

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

#### Example 3

Input:

```

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

```

Output:

```

Given a foo, we can...

```

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

#### Example 4

Input:

```

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

```

Output:

```

this is a symbol taking two arguments.

```

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

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

---

`\notation`

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

#### Example 5

Input:

```

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

```

Output:

```

First: a; Second: b

```

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

---

`\comp`

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

We can introduce a new notation `highlight` for `\binarysymbol` that fixes this flaw, which we can subsequently use with `\binarysymbol[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  $\text{\TeX}$  groups, making it ultimately almost impossible to determine them automatically while also remaining compatible with arbitrary highlighting customizations (such as tooltips, hyperlinks, colors) that users might employ, and that are ultimately invoked by `\comp`.



Note that it is required that

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

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

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

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

If you want an argument to a semantic macro to be a purely syntactic parameter, then you are likely somewhat confused with respect to the distinction between the precise *syntax* and *semantics* of the symbol you are trying to declare (which happens quite often even to experienced  $\text{\TeX}$  users), and might want to give those another thought - quite likely, the macro you aim to implement does not actually represent a semantically meaningful mathematical concept, and you will want to use `\def` and similar native  $\text{\LaTeX}$  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}[hl,args=2]
2   {\comp{\text{1.: }}#1\comp{\text{; 2.: }}#2}
3 $\newbinarysymbol{a}{b}$

```

Output:

```

1.: a; 2.: b

```

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

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

#### \setnotation

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

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

#### \textsymdecl

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

### Operator Notations

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

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

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

#### Example 8

Input:

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

Output:

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

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

### 3.3.3 Argument Modes

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

#### Mode-b Arguments

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

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

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

#### Example 9

Input:

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

Output:

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

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

## Mode-a Arguments

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

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

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

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

### Example 10

Input:

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

Output:

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

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

### Example 11

Input:

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

Output:

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

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

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

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

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

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

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

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

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

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

## Mode-B Arguments

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

### Example 12

Input:

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

Output:

$\forall x, y, z. P$

### 3.3.4 Type and Definiens Components

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

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

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

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

#### Example 13

Input:

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

Output:

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

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

#### Example 14

Input:

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

Output:

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

### 3.3.5 Precedences and Automated Bracketing

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

#### Example 15

Input:

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

Output:

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

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

#### Example 16

Input:

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

Output:

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

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

#### Example 17

Input:

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

Output:

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

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

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

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

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

### Example 18

Input:

```

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

```

Output:

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

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

---

`\infprec`  
`\neginfprec`

---

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

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

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

In the example above:

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



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

### 3.3.6 Variables

All symbol and notation declarations require a module with which they are associated, hence the commands  $\text{\textbackslashsymdecl}$ ,  $\text{\textbackslashnotation}$ ,  $\text{\textbackslashsymdef}$  etc. are disabled outside of  $\text{\textsf{s}}\text{\textsf{module}}$ -environments.

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

---

$\text{\textbackslashsvar}$

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

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

---

$\text{\textbackslashvardef}$

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

#### Example 19

Input:

```

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

```



Output:

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

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

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

### 3.3.7 Variable Sequences

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

`\varseq`

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

This is best shown by example:

#### Example 20

Input:

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

Output:

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

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

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

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

#### Example 21

Input:

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

Output:

$a_1 + \dots + a_n$

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

#### Example 22

Input:

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

Output:

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

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

#### Example 23

Input:

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

Output:

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

## 3.4 Module Inheritance and Structures

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

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

### 3.4.1 Multilinguality and Translations

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

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

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

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

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

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

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

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

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

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

### 3.4.2 Simple Inheritance and Namespaces

---

`\importmodule`  
`\usemodule`

---

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

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

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



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

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

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

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

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



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

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

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

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

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



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

---

`\STEXexport`

---

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



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

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

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

### 3.4.3 The `mathstructure` Environment

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

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

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

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

How this works is again best demonstrated by example:

### Example 24

Input:

```

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

Output:

A **monoid** is...

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

### Example 25

Input:

```

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

Output:

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

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

### Example 26

Input:

```

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

```

Output:

```

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

```

### \instantiate

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

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

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

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

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

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

### \varinstantiate

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

This allows us to do things like:

#### Example 27

Input:

```

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

```

Output:

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

.

and

### Example 28

Input:

```

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

```

Output:

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

.

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

### 3.4.4 The copymodule Environment

TODO: explain

Given modules:

### Example 29

Input:

```

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

```

Output:



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

### Example 30

Input:

```

1 \begin{smodule}{ring}
2   \begin{copymodule}{group}{addition}
3     \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: $\rtimes a\{rplus c\{rtimes de\}}$
19 \end{smodule}

```

Output:

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

TODO: explain donotclone

### 3.4.5 The interpretmodule Environment

TODO: explain

### Example 31

Input:

```

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

```

Output:

---

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

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

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

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

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

## Chapter 4

# Using $\text{\TeX}$ Symbols

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

### 4.1 `\symref` and its variants

---

`\symref`  
`\symname`

---

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

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

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

#### Example 32

Input:

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

Output:

A natural number is...

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

`\Symname`

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

### Example 33

Input:

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

Output:

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



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

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

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

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

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

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

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

### Example 34

Input:

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

Output:

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

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

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



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

## \arg

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

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

### Example 35

Input:

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

Output:

Addition is...

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

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

### Example 36

Input:

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

Output:

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

adding  $k$  yields...

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

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

### Example 37

Input:

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

Output:

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

## 4.3 Referencing Symbols and Statements

TODO: references documentation

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

## Chapter 5

# sTeX Statements

### 5.1 Definitions, Theorems, Examples, Paragraphs

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

The corresponding environments for that are:

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

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

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

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

#### Example 38

Input:

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

Output:

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

---

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

---

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

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

---

`\definiens`

---

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

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

### Example 39

Input:

---

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



```

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

```

Output:

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

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

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

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

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

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

```

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

```

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

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

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>spfcases</code> environments that mark up the cases one by one.
<code>spfcases</code>	The content of a <code>spfcases</code> environment are a sequence of case proofs marked up in the <code>spfcases</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>spfcases</code> environment is the same as that of a <code>proof</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>spfcases</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>5</sup>

```

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

```

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

## Chapter 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>6</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>6</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>7</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>7</sup>EDNOTE: MK: We need a substitute for the “Note that here the `display=flow` on the `sfragment` environment prevents numbering as is traditional for prefaces.”

---

`\currentsectionlevel`  
`\CurrentSectionLevel`

---

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

---

`\prematurestop`  
`\afterprematurestop`

---

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

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

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

---

`\setSGvar`  
`\useSGvar`

---

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

---

`\ifSGvar`

---

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

## 7.3 Slides and Course Notes

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

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

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

---

slides  
notes  
sectocframes  
frameimages  
fiboxed

---

The notesslides class takes a variety of class options:

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

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



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

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

```

1 \ifnotes\maketitle\else
2 \frame[noframenummering]\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   \end{solution}
15   \begin{gnote}
16     if they do not give the justification deduct 5 pts
17   \end{gnote}
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=Noooooooooooo,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>8</sup>without solutions (that is what the students see during the exam/quiz)<sup>8</sup>

---

<sup>8</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.

<hr/> <code>solutions</code> <code>notes</code> <code>hints</code> <code>gnotes</code> <code>pts</code> <code>min</code> <hr/>	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>assignment</code> <code>number</code>  <code>title</code> <code>type</code> <code>given</code> <code>due</code> <code>multiple</code>  <code>test</code>	<p>This package supplies the <code>assignment</code> environment that groups problems into assignment sheets. It takes an optional KeyVal argument with the keys <code>number</code> (for the assignment number; if none is given, 1 is assumed as the default or — in multi-assignment documents — the ordinal of the <code>assignment</code> environment), <code>title</code> (for the assignment title; this is referenced in the title of the assignment sheet), <code>type</code> (for the assignment type; e.g. “quiz”, or “homework”), <code>given</code> (for the date the assignment was given), and <code>due</code> (for the date the assignment is due).</p> <p>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.).</p> <p>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 L<sup>A</sup>T<sub>E</sub>X source.</p>
<code>\testspace</code> <code>\testnewpage</code> <code>\testemptypage</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>testheading</code> <code>duration</code> <code>min</code> <code>reqpts</code>	<p>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.</p> <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)

2022-05-24

**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:9

9

---

### `\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>9</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-</code> mode, inserting a <i>reference</i> to the file instead. Both also set <code>\ifinputref</code> to true.
<hr/> <hr/> <code>\addmhbibresource</code>	<code>\inputref[&lt;archive-ID&gt;]{&lt;filename&gt;}</code>  Adds a <code>.bib</code> -file <code>&lt;filename&gt;</code> in archive <code>&lt;archive-ID&gt;</code> (relative to the top-directory of the archive!).
<hr/> <hr/> <code>\libinput</code>	<code>\libinput{&lt;filename&gt;}</code>  Inputs <code>&lt;filename&gt;.tex</code> from the <code>lib</code> folders in the current archive and the <code>meta-inf-</code> archive of the current archive group(s) (if existent) in descending order. Throws an error if no file by that name exists in any of the relevant <code>lib</code> -folders.
<hr/> <hr/> <code>\libusepackage</code>	<code>\libusepackage[&lt;args&gt;]{&lt;filename&gt;}</code>  Like <code>\libinput</code> , but looks for <code>.sty</code> -files and calls <code>\usepackage[&lt;meta{args}&gt;]{&lt;Arg{filename}&gt;}</code> instead of <code>\input</code> . Throws an error, if none or more than one suitable package file is found.
<hr/> <hr/> <code>\mhgraphics</code> <code>\cmhgraphics</code>	<i>If</i> the <code>graphicx</code> package is loaded, these macros are defined at <code>\begin{document}</code> . <code>\mhgraphics</code> takes the same arguments as <code>\includegraphics</code> , with the additional optional key <code>mhrepos</code> . It then resolves the file path in <code>\mhgraphics[mhrepos=Foo/Bar]{foo/bar.png}</code> relative to the <code>source-</code> folder of the <code>Foo/Bar</code> -archive. <code>\cmhgraphics</code> additional wraps the image in a <code>center</code> -environment.
<hr/> <hr/> <code>\lstinputmhlisting</code> <code>\clstinputmhlisting</code>	Like <code>\mhgraphics</code> , but only defined if the <code>listings</code> -package is loaded, and with <code>\lstinputlisting</code> instead of <code>\includegraphics</code> .

# Chapter 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

---

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

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

---

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

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

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

---

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

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

# Chapter 11

## STEX-Modules

This sub package contains code related to Modules

### 11.1 Macros and Environments

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

---

---

`\c_stex_module_<URI>_prop`

A property list with the following fields:

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

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

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

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

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

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

**meta** the metatheory of the module.

---

---

`\c_stex_module_<URI>_code`

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

---

---

`\c_stex_module_<URI>_constants`

The names of all constants declared in the module

---

---

`\c_stex_module_<URI>_constants`

The full URIs of all modules imported in this module

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

---

`\stex_modules_current_namespace:`

---

Computes the current namespace as follows:

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

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

### 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` 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 {<archive-ID>} {<module-path>}`**

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

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

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

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

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

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

2. Otherwise:

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

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

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

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

---

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

---

stores the result in these four variables.

---

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

---

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

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

# Chapter 13

## STEX-Symbols

Code related to symbol declarations and notations

### 13.1 Macros and Environments

---

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

---

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

- **name**: An (OMDOC) name. By default equal to  $\langle\mathrm{macroname}\rangle$ .
- **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.  $\backslash\mathrm{symdecl}\{\mathrm{plus}\}[\mathrm{args}=\mathrm{ii}]$  allows for  $\backslash\mathrm{plus}\{2\}\{2\}$ .
  - a** an *associative* argument; i.e. a sequence of arbitrarily many arguments provided as a comma-separated list, e.g.  $\backslash\mathrm{symdecl}\{\mathrm{plus}\}[\mathrm{args}=\mathrm{a}]$  allows for  $\backslash\mathrm{plus}\{2,2,2\}$ .
  - 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.  $\backslash\mathrm{symdecl}\{\mathrm{forall}\}[\mathrm{args}=\mathrm{bi}]$  allows for  $\backslash\mathrm{forall}\{x\in\mathrm{Nat}\}\{x\geq 0\}$ .

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

# Chapter 14

## STEX-Terms

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

### 14.1 Macros and Environments

---

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

---

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

---

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

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

---

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

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

---

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

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

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

## Chapter 15

# TeX-Structural Features

Code related to structural features

### 15.1 Macros and Environments

#### 15.1.1 Structures

`mathstructure` TODO



## Chapter 16

# sTeX-Statements

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

### 16.1 Macros and Environments

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

## Chapter 17

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

## Chapter 18

# 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*~This~is~sTeX~version~3.1.0~*^^J}
57
58 %\RequirePackage{morewrites}
59 %\RequirePackage{amsmath}
60
61 Package options:
62 \keys_define:nn { stex } {
63   debug      .clist_set:N = \c_stex_debug_clist ,
64   lang       .clist_set:N = \c_stex_languages_clist ,
65   mathhub    .tl_set_x:N  = \mathhub ,
66   usesms     .bool_set:N  = \c_stex_persist_mode_bool ,
67   writesms   .bool_set:N  = \c_stex_persist_write_mode_bool ,
68   image      .bool_set:N  = \c_tikzinput_image_bool ,
69   unknown    .code:n      = {}
70 }
71 \ProcessKeysOptions { stex }

```

**\stex** The sTeX logo:

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

(End definition for \stex and \sTeX. These functions are documented on page 64.)

## 24.3 Messages and logging

```

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

```

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

```

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

```

(End definition for \stex\_debug:nn. This function is documented on page 64.)

Redirecting messages:

```

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

```

## 24.4 HTML Annotations

```

107 <@@=stex_annotate>

```

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

```

108 \tl_new:N \l_stex_html_arg_tl

```

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

`\stex_html_checkempty:n`

```

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

```

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

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

`\stex_if_do_html:TF`

```

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

```

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

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

```

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

```

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

`\stex_annotate:anv`

`\stex_annotate_invisible:n`

`\stex_annotate_invisible:nnn`

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

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

```

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

```

(End definition for `\stex_annotate:nnn`, `\stex_annotate_invisible:n`, and `\stex_annotate_invisible:nnn`. These functions are documented on page 65.)

## 24.5 Babel Languages

148 `<@=stex_language>`

`\c_stex_languages_prop`  
`\c_stex_language_abbrevs_prop`

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

```

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

```

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

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

```

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

```

```

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

```

## 24.6 Persistence

```

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

```



```

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

```

## 24.7 Auxiliary Methods

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

```

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

```

(End definition for \stex\_deactivate\_macro:Nn. This function is documented on page 65.)

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

```

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

```

(End definition for \stex\_reactivate\_macro:N. This function is documented on page 65.)

**\ignorespacesandpars**

```

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

```

```

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

```

(End definition for `\ignorespacesandpars`. This function is documented on page 65.)

`\MMTrule`

```

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

```

```

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

```

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

## Chapter 25

# STEX -MathHub Implementation

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

### 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`

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

```

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

```

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

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

```

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

```

(End definition for `\stex_path_to_string:NN` and `\stex_path_to_string:N`. These functions are documented on page 66.)

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

```

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

```

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

```

```

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

```

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

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

```

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

```

(End definition for `\stex_path_if_absolute:N $\underline{TF}$` . This function is documented on page 66.)

## 25.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

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

```

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

We determine the PWD

`\c_stex_pwd_seq`  
`\c_stex_pwd_str`

```

489 \sys_if_platform_windows:TF{
490   \begingroup\escapechar=-1\catcode'\=12
491   \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
492   \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
493   \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_
494   }}{
495   \stex_kpsewhich:n{-var-value~PWD}
496   }
497
498 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
499 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
500 \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 66.)

## 25.3 File Hooks and Tracking

501 `<@@=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

502 `\seq_gclear_new:N\g__stex_files_stack`

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`  
`\c_stex_mainfile_str`

```

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

```

(End definition for `\c_stex_mainfile_seq` and `\c_stex_mainfile_str`. These variables are documented on page 66.)

`\g_stex_currentfile_seq`

506 `\seq_gclear_new:N\g_stex_currentfile_seq`

(End definition for `\g_stex_currentfile_seq`. This variable is documented on page 67.)

`\stex_filestack_push:n`

```

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

```

```

513 }
514 \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
515 \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
516 }

```

(End definition for `\stex_filestack_push:n`. This function is documented on page 67.)

`\stex_filestack_pop:`

```

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

```

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

Hooks for the current file:

```

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

```

## 25.4 MathHub Repositories

```

534 <@=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.

```

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

```



```

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

```

(End definition for `\mathhub`, `\c_stex_mathhub_seq`, and `\c_stex_mathhub_str`. These variables are documented on page 67.)

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

```

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

```

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

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

599 \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:

```

600 \cs_new_protected:Nn \__stex_mathhub_find_manifest:N {
601   \seq_set_eq:NN\l_tmpa_seq #1
602   \bool_set_true:N\l_tmpa_bool
603   \bool_while_do:Nn \l_tmpa_bool {
604     \seq_if_empty:NTF \l_tmpa_seq {
605       \bool_set_false:N\l_tmpa_bool
606     }{
607       \file_if_exist:nTF{
608         \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
609       }{
610         \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
611         \bool_set_false:N\l_tmpa_bool
612       }{
613         \file_if_exist:nTF{
614           \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
615         }{
616           \seq_put_right:Nn\l_tmpa_seq{META-INF}
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             \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
628           }
629         }
630       }
631     }
632   }
633   \seq_set_eq:NN\l__stex_mathhub_manifest_file_seq\l_tmpa_seq
634 }

```

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

\c\_stex\_mathhub\_manifest\_ior

File variable used for MANIFEST-files

635 \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:

```

636 \cs_new_protected:Nn \__stex_mathhub_parse_manifest:n {
637   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
638   \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}

```

```

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

```

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

`\stex_set_current_repository:n`

```

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

```

(End definition for `\stex_set_current_repository:n`. This function is documented on page 67.)

`\stex_require_repository:n`

```

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

```

(End definition for `\stex_require_repository:n`. This function is documented on page 67.)

`\l_stex_current_repository_prop`

Current MathHub repository

```

694 %\prop_new:N \l_stex_current_repository_prop
695 \bool_if:NF \c_stex_persist_mode_bool {
696   \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
697   \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
698     \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
699   } {
700     \__stex_mathhub_parse_manifest:n { main }
701     \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
702     \l_tmpa_str
703     \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str_manifest_prop }
704     \c_stex_mathhub_main_manifest_prop
705     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
706     \stex_debug:nn{mathhub}{Current~repository:~
707     \prop_item:Nn \l_stex_current_repository_prop {id}
708   }
709 }
710 }

```

(End definition for `\l_stex_current_repository_prop`. This variable is documented on page 67.)

`\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.

```

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

```

```

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

```

(End definition for `\stex_in_repository:nn`. This function is documented on page 67.)

## 25.5 Using Content in Archives

`\mhpath`

```

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

```

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

`\inputref`

`\mhinput`

```

758 \newif \ifinputref \inputreffalse
759
760 \cs_new_protected:Nn \__stex_mathhub_mhinput:nn {
761   \stex_in_repository:nn {#1} {
762     \ifinputref
763       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
764     \else
765       \inputreftrue
766       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
767       \inputreffalse
768     \fi
769   }
770 }
771 \NewDocumentCommand \mhinput { 0{} m }{
772   \__stex_mathhub_mhinput:nn{ #1 }{ #2 }
773 }
774

```

```

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

```

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

#### `\addmhbibresource`

```

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

```

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

## `\libinput`

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

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

## `\libusepackage`

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

```

873 \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
874 \seq_put_right:No \l_tmpa_seq \l_tmpa_str
875 }
876
877 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2}
878 \IfFileExists{ \l_tmpa_str.sty }{
879 \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
880 }{}
881
882 \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
883 \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
884 }{
885 \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
886 \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
887 \usepackage[#1]{ #1 }
888 }
889 }{
890 \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
891 }
892 }
893 }

```

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

`\mhgraphics`  
`\cmhgraphics`

```

894
895 \AddToHook{begindocument}{
896 \ltx@ifpackageloaded{graphicx}{
897 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
898 \providecommand\mhgraphics[2] [] {%
899 \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
900 \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}}
901 \providecommand\cmhgraphics[2] [] {\begin{center}\mhgraphics[#1]{#2}\end{center}}
902 }{}

```

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

`\lstinputmhlisting`  
`\clstinputmhlisting`

```

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

```

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



## Chapter 26

# STEX -References Implementation

```
913 <*package>
914
915 %%%%%%%%%% references.dtx %%%%%%%%%%
916
917 <@@=stex_refs>
    Warnings and error messages
918
```

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

```
919 %\iow_new:N \c__stex_refs_refs_iow
920 \AtBeginDocument{
921 % \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
922 }
923 \AtEndDocument{
924 % \iow_close:N \c__stex_refs_refs_iow
925 }
```

`\STEXreftitle`

```
926 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
927
928 \NewDocumentCommand \STEXreftitle { m } {
929 \tl_gset:Nx \g__stex_refs_title_tl { #1 }
930 }
```

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

### 26.1 Document URIs and URLs

`\l_stex_current_docns_str`

```
931 \str_new:N \l_stex_current_docns_str
```

*(End definition for `\l_stex_current_docns_str`. This variable is documented on page 69.)*

`\stex_get_document_uri:`

```

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

```

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

`\l_stex_current_docurl_str`

```

972 \str_new:N \l_stex_current_docurl_str

```

(End definition for `\l_stex_current_docurl_str`. This variable is documented on page 69.)

`\stex_get_document_url:`

```

973 \cs_new_protected:Nn \stex_get_document_url: {
974   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
975   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
976   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str

```

```

977 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
978 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
979
980 \str_clear:N \l_tmpa_str
981 \prop_if_exist:NT \l_stex_current_repository_prop {
982   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
983     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
984       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
985     }
986   }
987 }
988
989 \str_if_empty:NTF \l_tmpa_str {
990   \str_set:Nx \l_stex_current_docurl_str {
991     file:/\stex_path_to_string:N \l_tmpa_seq
992   }
993 }{
994   \bool_set_true:N \l_tmpa_bool
995   \bool_while_do:Nn \l_tmpa_bool {
996     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
997     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
998       {source} { \bool_set_false:N \l_tmpa_bool }
999     }{}{
1000       \seq_if_empty:NT \l_tmpa_seq {
1001         \bool_set_false:N \l_tmpa_bool
1002       }
1003     }
1004   }
1005 }
1006 \seq_if_empty:NTF \l_tmpa_seq {
1007   \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
1008 }{
1009   \str_set:Nx \l_stex_current_docurl_str {
1010     \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
1011   }
1012 }
1013 }
1014 }

```

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

## 26.2 Setting Reference Targets

```

1015 \str_const:Nn \c__stex_refs_url_str{URL}
1016 \str_const:Nn \c__stex_refs_ref_str{REF}
1017 \str_new:N \l__stex_refs_curr_label_str
1018 % @currentlabel -> number
1019 % @currentlabelname -> title
1020 % @currentHref -> name.number <- id of some kind
1021 % \theH# -> \arabic{section}
1022 % \the# -> number
1023 % \hyper@makecurrent{#}
1024 \int_new:N \l__stex_refs_unnamed_counter_int

```

`\stex_ref_new_doc_target:n`

```

1025 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
1026   \stex_get_document_uri:
1027   \str_clear:N \l__stex_refs_curr_label_str
1028   \str_set:Nx \l_tmpa_str { #1 }
1029   \str_if_empty:NT \l_tmpa_str {
1030     \int_incr:N \l__stex_refs_unnamed_counter_int
1031     \str_set:Nx \l_tmpa_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
1032   }
1033   \str_set:Nx \l__stex_refs_curr_label_str {
1034     \l_stex_current_docns_str?\l_tmpa_str
1035   }
1036   \seq_if_exist:cF{g__stex_refs_labels_\l_tmpa_str_seq}{
1037     \seq_new:c {g__stex_refs_labels_\l_tmpa_str_seq}
1038   }
1039   \seq_if_in:coF{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str {
1040     \seq_gput_right:co{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str
1041   }
1042   \stex_if_smsmode:TF {
1043     \stex_get_document_url:
1044     \str_gset_eq:cN {sref_url_\l__stex_refs_curr_label_str_str}\l_stex_current_docurl_str
1045     \str_gset_eq:cN {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_url_str
1046   }{
1047     %\iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter\unexpanded\expandafter{
1048     \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
1049     \immediate\write\@auxout{\stexauxadddocref{\l_stex_current_docns_str}{\l_tmpa_str}}
1050     \str_gset:cx {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_ref_str
1051   }
1052 }

```

(End definition for `\stex_ref_new_doc_target:n`. This function is documented on page 69.)

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

```

1053 \cs_new_protected:Npn \stexauxadddocref #1 #2 {
1054   \str_set:Nn \l_tmpa_str {#1?#2}
1055   \str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
1056   \seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
1057     \seq_new:c {g__stex_refs_labels_#2_seq}
1058   }
1059   \seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
1060     \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
1061   }
1062 }

```

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

```

1063 \AtEndDocument{
1064   \def\stexauxadddocref#1 #2 {}{}
1065 }

```

`\stex_ref_new_sym_target:n`

```

1066 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
1067   \stex_if_smsmode:TF {
1068     \str_if_exist:cF{sref_sym_#1_type}{
1069       \stex_get_document_url:
1070       \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str

```

```

1071     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
1072   }
1073   ){
1074     \str_if_empty:NF \l__stex_refs_curr_label_str {
1075       \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
1076       \immediate\write\@auxout{
1077         \exp_not:N\expandafter\def\exp_not:N\csname \exp_not:N\detokenize{sref_sym_#1_label_
1078           \l__stex_refs_curr_label_str
1079         }
1080       }
1081     }
1082   }
1083 }

```

(End definition for `\stex_ref_new_sym_target:n`. This function is documented on page 69.)

## 26.3 Using References

```

1084 \str_new:N \l__stex_refs_indocument_str

```

**\sref** Optional arguments:

```

1085
1086 \keys_define:nn { stex / sref } {
1087   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
1088   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
1089   pre           .tl_set:N = \l__stex_refs_pre_tl ,
1090   post          .tl_set:N = \l__stex_refs_post_tl ,
1091 }
1092 \cs_new_protected:Nn \__stex_refs_args:n {
1093   \tl_clear:N \l__stex_refs_linktext_tl
1094   \tl_clear:N \l__stex_refs_fallback_tl
1095   \tl_clear:N \l__stex_refs_pre_tl
1096   \tl_clear:N \l__stex_refs_post_tl
1097   \str_clear:N \l__stex_refs_repo_str
1098   \keys_set:nn { stex / sref } { #1 }
1099 }

```

The actual macro:

```

1100 \NewDocumentCommand \sref { 0{} m}{
1101   \__stex_refs_args:n { #1 }
1102   \str_if_empty:NTF \l__stex_refs_indocument_str {
1103     \str_set:Nx \l_tmpa_str { #2 }
1104     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
1105     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
1106       \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
1107         \seq_get_left:cNF {g__stex_refs_labels_\l_tmpa_str _seq} \l_tmpa_str {
1108           \str_clear:N \l_tmpa_str
1109         }
1110       }{
1111         \str_clear:N \l_tmpa_str
1112       }
1113     }{
1114       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1115       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str

```

```

1116 \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
1117 \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str_seq}{
1118   \str_set_eq:NN \l_tmpc_str \l_tmpa_str
1119   \str_clear:N \l_tmpa_str
1120   \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str_seq} {
1121     \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
1122       \str_range:nnn { ##1 }{-\l_tmpa_int}{ -1 }
1123     }{
1124       \seq_map_break:n {
1125         \str_set:Nn \l_tmpa_str { ##1 }
1126       }
1127     }
1128   }
1129 }{
1130   \str_clear:N \l_tmpa_str
1131 }
1132 }
1133 \str_if_empty:NTF \l_tmpa_str {
1134   \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_lin
1135 }{
1136   \str_if_eq:cNTF {sref_\l_tmpa_str_type} \c__stex_refs_ref_str {
1137     \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1138       \cs_if_exist:cTF{autoref}{
1139         \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1140       }{
1141         \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1142       }
1143     }{
1144       \ltx@ifpackageloaded{hyperref}{
1145         \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
1146       }{
1147         \l__stex_refs_linktext_tl
1148       }
1149     }
1150   }{
1151     \ltx@ifpackageloaded{hyperref}{
1152       \href{\use:c{sref_url_\l_tmpa_str_str}}{\tl_if_empty:NTF \l__stex_refs_linktext_t
1153     }{
1154       \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs
1155     }
1156   }
1157 }
1158 }{
1159   % TODO
1160 }
1161 }

```

(End definition for `\sref`. This function is documented on page 70.)

## `\srefsym`

```

1162 \NewDocumentCommand \srefsym { 0{} m}{
1163   \stex_get_symbol:n { #2 }
1164   \__stex_refs_sym_aux:nn{##1}{\l_stex_get_symbol_uri_str}
1165 }

```

```

1166
1167 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1168   \str_if_exist:cTF {sref_sym_#2 _label_str }{
1169     \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1170   }{
1171     \__stex_refs_args:n { #1 }
1172     \str_if_empty:NTF \l__stex_refs_indocument_str {
1173       \tl_if_exist:cTF{sref_sym_#2 _type}{
1174         % doc uri in \l_tmpb_str
1175         \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1176         \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1177           % reference
1178           \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1179             \cs_if_exist:cTF{autoref}{
1180               \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1181             }{
1182               \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1183             }
1184           }{
1185             \ltx@ifpackageloaded{hyperref}{
1186               \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1187             }{
1188               \l__stex_refs_linktext_tl
1189             }
1190           }
1191         }{
1192           % URL
1193           \ltx@ifpackageloaded{hyperref}{
1194             \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl \
1195           }{
1196             \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_re
1197           }
1198         }
1199       }{
1200         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_l
1201       }
1202     }{
1203       % TODO
1204     }
1205   }
1206 }

```

(End definition for \srefsym. This function is documented on page 70.)

**\srefsymuri**

```

1207 \cs_new_protected:Npn \srefsymuri #1 #2 {
1208   \__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1209 }

```

(End definition for \srefsymuri. This function is documented on page 70.)

```

1210 </package>

```

## Chapter 27

# STEX -Modules Implementation

```
1211 <*package>
1212
1213 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1214
1215 <@@=stex_modules>
1216
1217 Warnings and error messages
1218 \msg_new:nnn{stex}{error/unknownmodule}{
1219   No~module~#1~found
1220 }
1221 \msg_new:nnn{stex}{error/syntax}{
1222   Syntax~error:~#1
1223 }
1224 \msg_new:nnn{stex}{error/siglanguage}{
1225   Module~#1~declares~signature~#2,~but~does~not~
1226   declare~its~language
1227 }
1228 \msg_new:nnn{stex}{warning/deprecated}{
1229   #1~is~deprecated;~please~use~#2~instead!
1230 }
1231 \msg_new:nnn{stex}{error/conflictingmodules}{
1232   Conflicting~imports~for~module~#1
1233 }
```

`\l_stex_current_module_str` The current module:

```
1233 \str_new:N \l_stex_current_module_str
```

*(End definition for `\l_stex_current_module_str`. This variable is documented on page 72.)*

`\l_stex_all_modules_seq` Stores all available modules

```
1234 \seq_new:N \l_stex_all_modules_seq
```

*(End definition for `\l_stex_all_modules_seq`. This variable is documented on page 72.)*



```

\stex_if_in_module_p:
\stex_if_in_module:TF
1235 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1236   \str_if_empty:NTF \l_stex_current_module_str
1237   \prg_return_false: \prg_return_true:
1238 }

```

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

```

\stex_if_module_exists_p:n
\stex_if_module_exists:nTF
1239 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1240   \prop_if_exist:cTF { c_stex_module_#1_prop }
1241   \prg_return_true: \prg_return_false:
1242 }

```

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

`\stex_add_to_current_module:n` Only allowed within modules:

```

\STEXexport
1243 \cs_new_protected:Nn \stex_execute_in_module:n { \stex_if_in_module:T {
1244   \stex_add_to_current_module:n { #1 }
1245   \stex_do_up_to_module:n { #1 }
1246 }}
1247 \cs_generate_variant:Nn \stex_execute_in_module:n {x}
1248
1249 \cs_new_protected:Nn \stex_add_to_current_module:n {
1250   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str_code} { #1 }
1251 }
1252 \cs_generate_variant:Nn \stex_add_to_current_module:n {x}
1253 \cs_new_protected:Npn \STEXexport {
1254   \ExplSyntaxOn
1255   \__stex_modules_export:n
1256 }
1257 \cs_new_protected:Nn \__stex_modules_export:n {
1258   \ignorespacesandpars#1\ExplSyntaxOff
1259   \stex_add_to_current_module:n { \ignorespacesandpars#1}
1260   \stex_smsmode_do:
1261 }
1262 \let \stex_module_export_helper:n \use:n
1263 \stex_deactivate_macro:Nn \STEXexport {module~environments}

```

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

```

\stex_add_constant_to_current_module:n
1264 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1265   \str_set:Nx \l_tmpa_str { #1 }
1266   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str_constants} { \l_tmpa_str }
1267 }

```

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

```

\stex_add_import_to_current_module:n
1268 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1269   \str_set:Nx \l_tmpa_str { #1 }
1270   \exp_args:Nno

```

```

1271 \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1272 \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1273 }
1274 }

```

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

`\stex_collect_imports:n`

```

1275 \cs_new_protected:Nn \stex_collect_imports:n {
1276 \seq_clear:N \l_stex_collect_imports_seq
1277 \__stex_modules_collect_imports:n {#1}
1278 }
1279 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1280 \seq_map_inline:cn {c_stex_module_#1_imports} {
1281 \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1282 \__stex_modules_collect_imports:n { ##1 }
1283 }
1284 }
1285 \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1286 \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1287 }
1288 }

```

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

`\stex_do_up_to_module:n`

```

1289 \int_new:N \l__stex_modules_group_depth_int
1290 \cs_new_protected:Nn \stex_do_up_to_module:n {
1291 \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1292 #1
1293 }{
1294 #1
1295 \expandafter \tl_gset:Nn
1296 \csname l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1297 \expandafter\expandafter\expandafter\endcsname
1298 \expandafter\expandafter\expandafter { \csname
1299 l__stex_modules_aftergroup_\l_stex_current_module_str _tl\endcsname #1 }
1300 \aftergroup\__stex_modules_aftergroup_do:
1301 }
1302 }
1303 \cs_generate_variant:Nn \stex_do_up_to_module:n {x}
1304 \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1305 \stex_debug:nn{aftergroup}{\cs_meaning:c{
1306 l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1307 }}}
1308 \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1309 \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1310 \tl_gclear:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1311 }{
1312 \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1313 \aftergroup\__stex_modules_aftergroup_do:
1314 }
1315 }
1316 \cs_new_protected:Nn \stex_reset_up_to_module:n {
1317 \expandafter\let\csname l__stex_modules_aftergroup_#1_tl\endcsname\undefined

```

1318 }

(End definition for \stex\_do\_up\_to\_module:n. This function is documented on page 72.)

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

1319

(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.

```
1320 \str_new:N \l_stex_module_ns_str
1321 \str_new:N \l_stex_module_subpath_str
1322 \cs_new_protected:Nn \__stex_modules_compute_namespace:nN {
1323   \seq_set_eq:NN \l_tmpa_seq #2
1324   % split off file extension
1325   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str % <- filename
1326   \exp_args:Nnno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1327   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str % <- filename without suffixes
1328   \seq_put_right:No \l_tmpa_seq \l_tmpb_str % <- file path including name without suffixes
1329
1330   \bool_set_true:N \l_tmpa_bool
1331   \bool_while_do:Nn \l_tmpa_bool {
1332     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1333     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1334       {source} { \bool_set_false:N \l_tmpa_bool }
1335     }{}{
1336       \seq_if_empty:NT \l_tmpa_seq {
1337         \bool_set_false:N \l_tmpa_bool
1338       }
1339     }
1340   }
1341
1342   \stex_path_to_string:NN \l_tmpa_seq \l_stex_module_subpath_str
1343   % \l_tmpa_seq <- sub-path relative to archive
1344   \str_if_empty:NTF \l_stex_module_subpath_str {
1345     \str_set:Nx \l_stex_module_ns_str {#1}
1346   }{
1347     \str_set:Nx \l_stex_module_ns_str {
1348       #1/\l_stex_module_subpath_str
1349     }
1350   }
1351 }
1352
1353 \cs_new_protected:Nn \stex_modules_current_namespace: {
1354   \str_clear:N \l_stex_module_subpath_str
1355   \prop_if_exist:NTF \l_stex_current_repository_prop {
1356     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1357     \__stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1358   }{
1359     % split off file extension
1360     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1361     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
```

```

1362 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1363 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1364 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1365 \str_set:Nx \l_stex_module_ns_str {
1366   file:/\stex_path_to_string:N \l_tmpa_seq
1367 }
1368 }
1369 }

```

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

## 27.1 The smodule environment

smodule arguments:

```

1370 \keys_define:nn { stex / module } {
1371   title      .tl_set:N      = \smodulename ,
1372   type       .str_set_x:N   = \smodulename ,
1373   id         .str_set_x:N   = \smoduleid ,
1374   deprecate  .str_set_x:N   = \l_stex_module_deprecate_str ,
1375   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1376   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1377   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1378   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1379   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1380   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1381   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1382 }
1383
1384 \cs_new_protected:Nn \__stex_modules_args:n {
1385   \str_clear:N \smodulename
1386   \str_clear:N \smodulename
1387   \str_clear:N \smoduleid
1388   \str_clear:N \l_stex_module_ns_str
1389   \str_clear:N \l_stex_module_deprecate_str
1390   \str_clear:N \l_stex_module_lang_str
1391   \str_clear:N \l_stex_module_sig_str
1392   \str_clear:N \l_stex_module_creators_str
1393   \str_clear:N \l_stex_module_contributors_str
1394   \str_clear:N \l_stex_module_meta_str
1395   \str_clear:N \l_stex_module_srccite_str
1396   \keys_set:nn { stex / module } { #1 }
1397 }
1398
1399 % module parameters here? In the body?
1400

```

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

```

1401 \cs_new_protected:Nn \stex_module_setup:nn {
1402   \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1403   \str_set:Nx \l_stex_module_name_str { #2 }
1404   \__stex_modules_args:n { #1 }

```

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

Are we in a nested module?

```

1405 \stex_if_in_module:TF {
1406   % Nested module
1407   \prop_get:cnN {c_stex_module\_l_stex_current_module_str _prop}
1408   { ns } \l_stex_module_ns_str
1409   \str_set:Nx \l_stex_module_name_str {
1410     \prop_item:cn {c_stex_module\_l_stex_current_module_str _prop}
1411     { name } / \l_stex_module_name_str
1412   }
1413   \str_if_empty:NT \l_stex_module_lang_str {
1414     \str_set:Nx \l_stex_module_lang_str {
1415       \prop_item:cn {c_stex_module\_l_stex_current_module_str _prop}
1416       { lang }
1417     }
1418   }
1419 }{
1420   % not nested:
1421   \str_if_empty:NT \l_stex_module_ns_str {
1422     \stex_modules_current_namespace:
1423     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1424       / {\l_stex_module_ns_str}
1425     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1426     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1427       \str_set:Nx \l_stex_module_ns_str {
1428         \stex_path_to_string:N \l_tmpa_seq
1429       }
1430     }
1431   }
1432 }

```

Next, we determine the language of the module:

```

1433 \str_if_empty:NT \l_stex_module_lang_str {
1434   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1435   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1436   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1437   \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
1438     \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
1439       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
1440     }
1441   }
1442   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1443   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
1444     \seq_pop_right:NN \l_tmpa_seq \l_stex_module_lang_str
1445     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1446       inferred~from~file~name}
1447   }
1448 }
1449
1450 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
1451   \exp_args:NNo \stex_set_language:Nn \l_tmpa_str \l_stex_module_lang_str
1452 }}

```

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

```

1453 \str_if_empty:NTF \l_stex_module_sig_str {
1454   \exp_args:Nnx \prop_gset_from_keyval:cn {
1455     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1456   } {
1457     name      = \l_stex_module_name_str ,
1458     ns        = \l_stex_module_ns_str ,
1459     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1460     lang      = \l_stex_module_lang_str ,
1461     sig       = \l_stex_module_sig_str ,
1462     deprecate = \l_stex_module_deprecate_str ,
1463     meta      = \l_stex_module_meta_str
1464   }
1465   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1466   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1467   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _copymodules}
1468   \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1469   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1470 \str_if_empty:NT \l_stex_module_meta_str {
1471   \str_set:Nx \l_stex_module_meta_str {
1472     \c_stex_metatheory_ns_str ? Metatheory
1473   }
1474 }
1475 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1476   \bool_set_true:N \l_stex_in_meta_bool
1477   \exp_args:Nx \stex_add_to_current_module:n {
1478     \bool_set_true:N \l_stex_in_meta_bool
1479     \stex_activate_module:n {\l_stex_module_meta_str}
1480     \bool_set_false:N \l_stex_in_meta_bool
1481   }
1482   \stex_activate_module:n {\l_stex_module_meta_str}
1483   \bool_set_false:N \l_stex_in_meta_bool
1484 }
1485 }{
1486   \str_if_empty:NT \l_stex_module_lang_str {
1487     \msg_error:nnxx{stex}{error/siglanguage}{
1488       \l_stex_module_ns_str?\l_stex_module_name_str
1489     }{\l_stex_module_sig_str}
1490   }
1491   \stex_debug:nn{modules}{Signature~\l_stex_module_sig_str~for~\l_stex_module_ns_str?\l_st
1492   \stex_if_module_exists:nTF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1493     \stex_debug:nn{modules}{(already exists)}
1494   }{
1495     \stex_debug:nn{modules}{(needs loading)}
1496     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1497     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1498     \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1499     \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1500     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1501     \str_set:Nx \l_tmpa_str {
1502       \stex_path_to_string:N \l_tmpa_seq /

```

```

1503     \l_tmpa_str . \l_stex_module_sig_str .tex
1504 }
1505 \IfFileExists \l_tmpa_str {
1506     \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1507         \str_clear:N \l_stex_current_module_str
1508         \seq_clear:N \l_stex_all_modules_seq
1509         \stex_debug:nn{modules}{Loading~signature}
1510     }
1511 }{
1512     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1513 }
1514 }
1515 \stex_if_smsmode:F {
1516     \stex_activate_module:n {
1517         \l_stex_module_ns_str ? \l_stex_module_name_str
1518     }
1519 }
1520 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1521 }
1522 \str_if_empty:NF \l_stex_module_deprecate_str {
1523     \msg_warning:nnxx{stex}{warning/deprecated}{
1524         Module~\l_stex_current_module_str
1525     }{
1526         \l_stex_module_deprecate_str
1527     }
1528 }
1529 \seq_put_right:Nx \l_stex_all_modules_seq {
1530     \l_stex_module_ns_str ? \l_stex_module_name_str
1531 }
1532 \tl_clear:c{l__stex_modules_aftergroup_\l_stex_module_ns_str ? \l_stex_module_name_str _tl}
1533 }

```

(End definition for `\stex_module_setup:nn`. This function is documented on page [73](#).)

**smodule** The module environment.

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

```

1534 \cs_new_protected:Nn \_stex_modules_begin_module: {
1535     \stex_reactivate_macro:N \STEXexport
1536     \stex_reactivate_macro:N \importmodule
1537     \stex_reactivate_macro:N \symdecl
1538     \stex_reactivate_macro:N \notation
1539     \stex_reactivate_macro:N \symdef
1540
1541     \stex_debug:nn{modules}{
1542         New~module:\\
1543         Namespace:~\l_stex_module_ns_str\\
1544         Name:~\l_stex_module_name_str\\
1545         Language:~\l_stex_module_lang_str\\
1546         Signature:~\l_stex_module_sig_str\\
1547         Metatheory:~\l_stex_module_meta_str\\
1548         File:~\stex_path_to_string:N \g_stex_currentfile_seq
1549     }
1550

```

```

1551 \stex_if_do_html:T{
1552   \begin{stex_annotate_env} {theory} {
1553     \l_stex_module_ns_str ? \l_stex_module_name_str
1554   }
1555
1556   \stex_annotate_invisible:nnn{header}{} {
1557     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1558     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1559     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1560       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1561     }
1562     \str_if_empty:NF \smoduletype {
1563       \stex_annotate:nnn{type}{\smoduletype}{}
1564     }
1565   }
1566 }
1567 % TODO: Inherit metatheory for nested modules?
1568 }
1569 \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}`

```

1570 \cs_new_protected:Nn \_stex_modules_end_module: {
1571   \stex_debug:nn{modules}{Closing module~\prop_item:cn {c_stex_module_\l_stex_current_module}
1572   \stex_reset_up_to_module:n \l_stex_current_module_str
1573   \stex_if_smsmode:T {
1574     \stex_persist:x {
1575       \prop_set_from_keyval:cn{c_stex_module_\l_stex_current_module_str _prop}{
1576         \exp_after:wN \prop_to_keyval:N \csname c_stex_module_\l_stex_current_module_str _pr
1577       }
1578       \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _constants}{
1579         \seq_use:cn{c_stex_module_\l_stex_current_module_str _constants},
1580       }
1581       \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _imports}{
1582         \seq_use:cn{c_stex_module_\l_stex_current_module_str _imports},
1583       }
1584       \tl_set:cn {c_stex_module_\l_stex_current_module_str _code}
1585     }
1586     \exp_after:wN \let \exp_after:wN \l_tmpa_tl \csname c_stex_module_\l_stex_current_module
1587     \exp_after:wN \stex_persist:n \exp_after:wN { \exp_after:wN { \l_tmpa_tl } }
1588   }
1589 }

```

(End definition for `\_stex_modules_end_module:.`)

The core environment

```

1590 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1591 \NewDocumentEnvironment { smodule } { 0 } { m } {
1592   \stex_module_setup:nn{#1}{#2}
1593   %\par
1594   \stex_if_smsmode:F{
1595     \tl_if_empty:NF \smoduletitle {
1596       \exp_args:No \stex_document_title:n \smoduletitle
1597     }

```



```

1598 \tl_clear:N \l_tmpa_tl
1599 \clist_map_inline:Nn \smodulotype {
1600   \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1601     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1602   }
1603 }
1604 \tl_if_empty:NTF \l_tmpa_tl {
1605   \__stex_modules_smodule_start:
1606 }{
1607   \l_tmpa_tl
1608 }
1609 }
1610 \__stex_modules_begin_module:
1611 \str_if_empty:NF \smoduleid {
1612   \stex_ref_new_doc_target:n \smoduleid
1613 }
1614 \stex_smsmode_do:
1615 } {
1616   \__stex_modules_end_module:
1617   \stex_if_smsmode:F {
1618     \end{stex_annotate_env}
1619     \clist_set:Nn \l_tmpa_clist \smodulotype
1620     \tl_clear:N \l_tmpa_tl
1621     \clist_map_inline:Nn \l_tmpa_clist {
1622       \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1623         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1624       }
1625     }
1626     \tl_if_empty:NTF \l_tmpa_tl {
1627       \__stex_modules_smodule_end:
1628     }{
1629       \l_tmpa_tl
1630     }
1631   }
1632 }

```

### **\stexpatchmodule**

```

1633 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1634 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1635
1636 \newcommand\stexpatchmodule[3] [] {
1637   \str_set:Nx \l_tmpa_str{ #1 }
1638   \str_if_empty:NTF \l_tmpa_str {
1639     \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1640     \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1641   }{
1642     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1643     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1644   }
1645 }

```

(End definition for \stexpatchmodule. This function is documented on page 73.)

## 27.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n 1646 \NewDocumentCommand \STEXModule { m } {
1647   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1648   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1649   \tl_set:Nn \l_tmpa_tl {
1650     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1651   }
1652   \seq_map_inline:Nn \l_stex_all_modules_seq {
1653     \str_set:Nn \l_tmpb_str { ##1 }
1654     \str_if_eq:eeT { \l_tmpa_str } {
1655       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1656     } {
1657       \seq_map_break:n {
1658         \tl_set:Nn \l_tmpa_tl {
1659           \stex_invoke_module:n { ##1 }
1660         }
1661       }
1662     }
1663   }
1664   \l_tmpa_tl
1665 }
1666
1667 \cs_new_protected:Nn \stex_invoke_module:n {
1668   \stex_debug:nn{modules}{Invoking~module~#1}
1669   \peek_charcode_remove:NTF ! {
1670     \__stex_modules_invoke_uri:nN { #1 }
1671   } {
1672     \peek_charcode_remove:NTF ? {
1673       \__stex_modules_invoke_symbol:nn { #1 }
1674     } {
1675       \msg_error:nnx{stex}{error/syntax}{
1676         ?~or~!~expected~after~
1677         \c_backslash_str STEXModule{#1}
1678       }
1679     }
1680   }
1681 }
1682
1683 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1684   \str_set:Nn #2 { #1 }
1685 }
1686
1687 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1688   \stex_invoke_symbol:n{#1?#2}
1689 }

```

(End definition for `\STEXModule` and `\stex_invoke_module:n`. These functions are documented on page 73.)

```

\stex_activate_module:n
1690 \bool_new:N \l_stex_in_meta_bool
1691 \bool_set_false:N \l_stex_in_meta_bool

```

```

1692 \cs_new_protected:Nn \stex_activate_module:n {
1693   \stex_debug:nn{modules}{Activating~module~#1}
1694   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1695     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1696     \use:c{ c_stex_module_#1_code }
1697   }
1698 }

```

*(End definition for \stex\_activate\_module:n. This function is documented on page 74.)*

```

1699 \endpackage

```

## Chapter 28

# STEX -Module Inheritance Implementation

```
1700 <*package>
1701
1702 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1703
```

### 28.1 SMS Mode

```
1704 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1705 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1706 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1707 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1708
1709 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1710   \makeatletter
1711   \makeatother
1712   \ExplSyntaxOn
1713   \ExplSyntaxOff
1714   \rustexBREAK
1715 }
1716
1717 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1718   \symdef
1719   \importmodule
1720   \notation
1721   \symdecl
1722   \STEXexport
1723   \inlineass
1724   \inlinedef
1725   \inlineex
1726   \endinput
1727   \setnotation
```

```

1728 \copynotation
1729 \assign
1730 \renamedekl
1731 \donotcopy
1732 \instantiate
1733 \textsymdecl
1734 }
1735
1736 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1737   \tl_to_str:n {
1738     smodule,
1739     copymodule,
1740     interpretmodule,
1741     realization,
1742     sdefinition,
1743     sexample,
1744     sassertion,
1745     sparagraph,
1746     mathstructure
1747   }
1748 }

```

(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 75.)

`\stex_if_smsmode_p:`

```

\stex_if_smsmode:TF
1749 \bool_new:N \g__stex_smsmode_bool
1750 \bool_set_false:N \g__stex_smsmode_bool
1751 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1752   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1753 }

```

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

`\_stex_smsmode_in_smsmode:nn`

```

1754 \cs_new_protected:Nn \_stex_smsmode_in_smsmode:nn { \stex_suppress_html:n {
1755   \vbox_set:Nn \l_tmpa_box {
1756     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1757     \bool_gset_true:N \g__stex_smsmode_bool
1758     #2
1759     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1760   }
1761   \box_clear:N \l_tmpa_box
1762 } }

```

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

`\stex_file_in_smsmode:nn`

```

1763 \quark_new:N \q__stex_smsmode_break
1764
1765 \NewDocumentCommand \_stex_smsmode_importmodule: { 0{} m } {
1766   \seq_gput_right:Nn \l__stex_smsmode_importmodules_seq {{#1}{#2}}
1767   \stex_smsmode_do:
1768 }
1769

```

```

1770 \cs_new_protected:Nn \__stex_smsmode_module:nn {
1771   \__stex_modules_args:n{#1}
1772   \stex_if_in_module:F {
1773     \str_if_empty:NF \l_stex_module_sig_str {
1774       \stex_modules_current_namespace:
1775       \str_set:Nx \l_stex_module_name_str { #2 }
1776       \stex_if_module_exists:nF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1777         \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1778         \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1779         \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1780         \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1781         \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1782         \str_set:Nx \l_tmpa_str {
1783           \stex_path_to_string:N \l_tmpa_seq /
1784           \l_tmpa_str . \l_stex_module_sig_str .tex
1785         }
1786         \IfFileExists \l_tmpa_str {
1787           \exp_args:NNx \seq_gput_right:Nn \l__stex_smsmode_sigmodules_seq \l_tmpa_str
1788         }{
1789           \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1790         }
1791       }
1792     }
1793   }
1794 }
1795
1796 \prg_new_conditional:Nnn \__stex_smsmode_check_import_pair:nn {T,F,TF} {
1797   %\stex_debug:nn{import-pair}{\detokenize{#1}~{#2}}
1798   \tl_if_empty:nTF{#1}{
1799     \prop_if_exist:NTF \l_stex_current_repository_prop
1800     {
1801       %\stex_debug:nn{import-pair}{in repository \prop_item:Nn \l_stex_current_repository_
1802       \prg_return_true:
1803     } {
1804       \seq_set_split:Nnn \l_tmpa_seq ? {#2}
1805       \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
1806       \tl_if_empty:NT \l_tmpa_tl {
1807         \seq_pop_left:NN \l_tmpa_seq \l_tmpa_tl
1808       }
1809       %\stex_debug:nn{import-pair}{\seq_use:Nn \l_tmpa_seq,~of~length~\seq_count:N \l_tmpa
1810       \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1
1811       \prg_return_true: \prg_return_false:
1812     }
1813   }\prg_return_true:
1814 }
1815
1816 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1817   \stex_filestack_push:n{#1}
1818   \seq_gclear:N \l__stex_smsmode_importmodules_seq
1819   \seq_gclear:N \l__stex_smsmode_sigmodules_seq
1820   % ----- new -----
1821   \__stex_smsmode_in_smsmode:nn{#1}{
1822     \let\importmodule\__stex_smsmode_importmodule:
1823     \let\stex_module_setup:nn\__stex_smsmode_module:nn

```

```

1824 \let\__stex_modules_begin_module:\relax
1825 \let\__stex_modules_end_module:\relax
1826 \seq_clear:N \g_stex_smsmode_allowedenvs_seq
1827 \exp_args:NNx \seq_put_right:Nn \g_stex_smsmode_allowedenvs_seq {\tl_to_str:n{smodule}}
1828 \tl_clear:N \g_stex_smsmode_allowedmacros_tl
1829 \tl_clear:N \g_stex_smsmode_allowedmacros_escape_tl
1830 \tl_put_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\importmodule}
1831 \everyeof{\q__stex_smsmode_break\noexpand}
1832 \expandafter\expandafter\expandafter
1833 \stex_smsmode_do:
1834 \csname @ @ input\endcsname "#1"\relax
1835
1836 \seq_map_inline:Nn \l__stex_smsmode_sigmodules_seq {
1837   \stex_filestack_push:n{##1}
1838   \expandafter\expandafter\expandafter
1839   \stex_smsmode_do:
1840   \csname @ @ input\endcsname "##1"\relax
1841   \stex_filestack_pop:
1842 }
1843 }
1844 % ----- new -----
1845 \__stex_smsmode_in_smsmode:nn{#1} {
1846   #2
1847   % ----- new -----
1848   \begingroup
1849   \%stex_debug:nn{smsmode}{Here:~\seq_use:Nn\l__stex_smsmode_importmodules_seq, }
1850   \seq_map_inline:Nn \l__stex_smsmode_importmodules_seq {
1851     \__stex_smsmode_check_import_pair:nnT ##1 { \begingroup
1852       \stex_import_module_uri:nn ##1
1853       \stex_import_require_module:nnnn
1854       \l_stex_import_ns_str
1855       \l_stex_import_archive_str
1856       \l_stex_import_path_str
1857       \l_stex_import_name_str \endgroup
1858     }
1859   }
1860   \endgroup
1861   \%stex_debug:nn{smsmode}{Actually~loading~file~#1}
1862   % ----- new -----
1863   \everyeof{\q__stex_smsmode_break\noexpand}
1864   \expandafter\expandafter\expandafter
1865   \stex_smsmode_do:
1866   \csname @ @ input\endcsname "#1"\relax
1867 }
1868 \stex_filestack_pop:
1869 }

```

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

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

```

1870 \cs_new_protected:Npn \stex_smsmode_do: {
1871   \stex_if_smsmode:T {
1872     \__stex_smsmode_do:w

```

```

1873 }
1874 }
1875 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1876   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
1877     \expandafter\if\expandafter\relax\noexpand#1
1878     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1879   \else\expandafter\__stex_smsmode_do:w\fi
1880 }{
1881   \__stex_smsmode_do:w % #1
1882 }
1883 }
1884 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1885   \cs_if_eq:NNTF #1 \q__stex_smsmode_break {
1886     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1887       #1\__stex_smsmode_do:w
1888     }{
1889       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1890         #1
1891       }{
1892         \cs_if_eq:NNTF \begin #1 {
1893           \__stex_smsmode_check_begin:n
1894         }{
1895           \cs_if_eq:NNTF \end #1 {
1896             \__stex_smsmode_check_end:n
1897           }{
1898             \__stex_smsmode_do:w
1899           }
1900         }
1901       }
1902     }
1903   }
1904 }
1905
1906 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1907   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1908     \begin{#1}
1909   }{
1910     \__stex_smsmode_do:w
1911   }
1912 }
1913 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1914   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1915     \end{#1}\__stex_smsmode_do:w
1916   }{
1917     \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1918   }
1919 }

```

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

## 28.2 Inheritance

```

1920 <@@=stex_importmodule>

```



`\stex_import_module_uri:nn`

```
1921 \cs_new_protected:Nn \stex_import_module_uri:nn {
1922   \str_set:Nx \l_stex_import_archive_str { #1 }
1923   \str_set:Nn \l_stex_import_path_str { #2 }
1924
1925   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1926   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1927   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1928
1929   \stex_modules_current_namespace:
1930   \bool_lazy_all:nTF {
1931     {\str_if_empty_p:N \l_stex_import_archive_str}
1932     {\str_if_empty_p:N \l_stex_import_path_str}
1933     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1934   }{
1935     \str_set_eq:NN \l_stex_import_path_str \l_stex_module_subpath_str
1936     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1937   }{
1938     \str_if_empty:NT \l_stex_import_archive_str {
1939       \prop_if_exist:NT \l_stex_current_repository_prop {
1940         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1941       }
1942     }
1943     \str_if_empty:NTF \l_stex_import_archive_str {
1944       \str_if_empty:NF \l_stex_import_path_str {
1945         \stex_path_from_string:Nn \l_tmpb_seq {
1946           \l_stex_module_ns_str / .. / \l_stex_import_path_str
1947         }
1948         \str_set:Nx \l_stex_import_ns_str {\stex_path_to_string:N \l_tmpb_seq}
1949         \str_replace_once:Nnn \l_stex_import_ns_str {file://} {file:///}
1950       }
1951     }{
1952       \stex_require_repository:n \l_stex_import_archive_str
1953       \prop_get:cnN { c_stex_mathhub_\l_stex_import_archive_str_manifest_prop } { ns }
1954       \l_stex_import_ns_str
1955       \str_if_empty:NF \l_stex_import_path_str {
1956         \str_set:Nx \l_stex_import_ns_str {
1957           \l_stex_import_ns_str / \l_stex_import_path_str
1958         }
1959       }
1960     }
1961   }
1962 }
```

(End definition for `\stex_import_module_uri:nn`. This function is documented on page 77.)

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

Store the return values of `\stex_import_module_uri:nn`.

```
1963 \str_new:N \l_stex_import_name_str
1964 \str_new:N \l_stex_import_archive_str
1965 \str_new:N \l_stex_import_path_str
1966 \str_new:N \l_stex_import_ns_str
```

(End definition for `\l_stex_import_name_str` and others. These variables are documented on page 77.)

```

\stex_import_require_module:nnnnn {\langle ns \rangle} {\langle archive-ID \rangle} {\langle path \rangle} {\langle name \rangle}
1967 \cs_new_protected:Nn \stex_import_require_module:nnnnn {
1968 \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1969
1970 \stex_debug:nn{requiremodule}{Here:\!\!\sim1:\!\!\sim2:\!\!\sim3:\!\!\sim4:\!\!\sim4}
1971
1972 \exp_args:NNxx \seq_set_split:Nnn \l_tmpa_seq {\tl_to_str:n{/}} {#4}
1973 \seq_get_left:NN \l_tmpa_seq \l_tmpc_str
1974
1975 %\stex_debug:nn{requiremodule}{Top~module:\l_tmpc_str}
1976
1977 % archive
1978 \str_set:Nx \l_tmpa_str { #2 }
1979 \str_if_empty:NTF \l_tmpa_str {
1980 \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1981 \seq_put_right:Nn \l_tmpa_seq {...}
1982 } {
1983 \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1984 \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1985 \seq_put_right:Nn \l_tmpa_seq { source }
1986 }
1987
1988 % path
1989 \str_set:Nx \l_tmpb_str { #3 }
1990 \str_if_empty:NTF \l_tmpb_str {
1991 \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / \l_tmpc_str }
1992
1993 \ltx@ifpackageloaded{babel} {
1994 \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1995 { \language } \l_tmpb_str {
1996 \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1997 }
1998 } {
1999 \str_clear:N \l_tmpb_str
2000 }
2001
2002 \stex_debug:nn{modules}{Checking~a1~\l_tmpa_str.\l_tmpb_str.tex}
2003 \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2004 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2005 }{
2006 \stex_debug:nn{modules}{Checking~a2~\l_tmpa_str.tex}
2007 \IfFileExists{ \l_tmpa_str.tex }{
2008 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2009 }{
2010 % try english as default
2011 \stex_debug:nn{modules}{Checking~a3~\l_tmpa_str.en.tex}
2012 \IfFileExists{ \l_tmpa_str.en.tex }{
2013 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2014 }{
2015 \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2016 }
2017 }
2018 }
2019

```

```

2020 } {
2021   \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
2022   \seq_concat:NNN \l_tmpb_seq \l_tmpa_seq \l_tmpb_seq
2023
2024   \ltx@ifpackageloaded{babel} {
2025     \exp_args:NnX \prop_get:NnNF \c_stex_language_abbrevs_prop
2026       { \language } \l_tmpb_str {
2027       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
2028     }
2029   } {
2030     \str_clear:N \l_tmpb_str
2031   }
2032
2033   \stex_path_canonicalize:N \l_tmpb_seq
2034   \stex_path_to_string:NN \l_tmpb_seq \l_tmpa_str
2035
2036   \stex_debug:nn{modules}{Checking~b1~\l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex}
2037   \IfFileExists{ \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }{
2038     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }
2039   }{
2040     \stex_debug:nn{modules}{Checking~b2~\l_tmpa_str/\l_tmpc_str.tex}
2041     \IfFileExists{ \l_tmpa_str/\l_tmpc_str.tex }{
2042       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.tex }
2043     }{
2044       % try english as default
2045       \stex_debug:nn{modules}{Checking~b3~\l_tmpa_str/\l_tmpc_str.en.tex}
2046       \IfFileExists{ \l_tmpa_str/\l_tmpc_str.en.tex }{
2047         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.en.tex }
2048       }{
2049         \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.\l_tmpb_str.tex}
2050         \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2051           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2052         }{
2053           \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.tex}
2054           \IfFileExists{ \l_tmpa_str.tex }{
2055             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2056           }{
2057             % try english as default
2058             \stex_debug:nn{modules}{Checking~b5~\l_tmpa_str.en.tex}
2059             \IfFileExists{ \l_tmpa_str.en.tex }{
2060               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2061             }{
2062               \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2063             }
2064           }
2065         }
2066       }
2067     }
2068   }
2069 }
2070
2071 \str_if_eq:eeF{\g__stex_importmodule_file_str}{\seq_use:Nn \g_stex_currentfile_seq /}{
2072   \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
2073     \seq_clear:N \l_stex_all_modules_seq

```

```

2074 \str_clear:N \l_stex_current_module_str
2075 \str_set:Nx \l_tmpb_str { #2 }
2076 \str_if_empty:NF \l_tmpb_str {
2077   \stex_set_current_repository:n { #2 }
2078 }
2079 \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
2080 }
2081
2082 \stex_if_module_exists:nF { #1 ? #4 } {
2083   \msg_error:nnx{stex}{error/unknownmodule}{
2084     #1?#4~(in~file~\g__stex_importmodule_file_str)
2085   }
2086 }
2087 }
2088
2089 }
2090 \stex_activate_module:n { #1 ? #4 }
2091 }

```

(End definition for `\stex_import_require_module:nnnn`. This function is documented on page 77.)

## `\importmodule`

```

2092 \NewDocumentCommand \importmodule { 0{} m } {
2093   \stex_import_module_uri:nn { #1 } { #2 }
2094   \stex_debug:nn{modules}{Importing~module:~
2095     \l_stex_import_ns_str ? \l_stex_import_name_str
2096   }
2097   \stex_import_require_module:nnnn
2098   { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2099   { \l_stex_import_path_str } { \l_stex_import_name_str }
2100   \stex_if_smsmode:F {
2101     \stex_annotate_invisible:nnn
2102     {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
2103   }
2104   \exp_args:Nx \stex_add_to_current_module:n {
2105     \stex_import_require_module:nnnn
2106     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2107     { \l_stex_import_path_str } { \l_stex_import_name_str }
2108   }
2109   \exp_args:Nx \stex_add_import_to_current_module:n {
2110     \l_stex_import_ns_str ? \l_stex_import_name_str
2111   }
2112   \stex_smsmode_do:
2113   \ignorespacesandpars
2114 }
2115 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

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

## `\usemodule`

```

2116 \NewDocumentCommand \usemodule { 0{} m } {
2117   \stex_if_smsmode:F {
2118     \stex_import_module_uri:nn { #1 } { #2 }
2119     \stex_import_require_module:nnnn
2120     { \l_stex_import_ns_str } { \l_stex_import_archive_str }

```

```

2121 { \l_stex_import_path_str } { \l_stex_import_name_str }
2122 \stex_annotate_invisible:nnn
2123 {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
2124 }
2125 \stex_smsmode_do:
2126 \ignorespacesandpars
2127 }

```

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

```

2128 \cs_new_protected:Nn \stex_csl_to_imports:Nn {
2129   \tl_if_empty:nF{#2}{
2130     \clist_set:Nn \l_tmpa_clist {#2}
2131     \clist_map_inline:Nn \l_tmpa_clist {
2132       \tl_if_head_eq_charcode:nNTF {##1} [{
2133         #1 ##1
2134       }{
2135         #1{##1}
2136       }
2137     }
2138   }
2139 }
2140 \cs_generate_variant:Nn \stex_csl_to_imports:Nn {No}
2141
2142
2143 \endpackage

```

## Chapter 29

# STEX -Symbols Implementation

```
2144 <*package>
2145
2146 %%%%%%%%%% symbols.dtx %%%%%%%%%%
2147
2148 Warnings and error messages
2149 \msg_new:nnn{stex}{error/wrongargs}{
2150   args~value~in~symbol~declaration~for~#1~
2151   needs~to~be~i,~a,~b~or~B,~but~#2~given
2152 }
2153 \msg_new:nnn{stex}{error/unknownsymbol}{
2154   No~symbol~#1~found!
2155 }
2156 \msg_new:nnn{stex}{error/seqlength}{
2157   Expected~#1~arguments;~got~#2!
2158 }
2159 \msg_new:nnn{stex}{error/unknownnotation}{
2160   Unknown~notation~#1~for~#2!
2161 }
```

### 29.1 Symbol Declarations

```
2161 <@@=stex_symdecl>
\stex_all_symbols:n Map over all available symbols
2162 \cs_new_protected:Nn \stex_all_symbols:n {
2163   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
2164   \seq_map_inline:Nn \l_stex_all_modules_seq {
2165     \seq_map_inline:cn{c_stex_module_##1_constants}{
2166       \__stex_symdecl_all_symbols_cs{##1?####1}
2167     }
2168   }
2169 }
```

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

## **\STEXsymbol**

```
2170 \NewDocumentCommand \STEXsymbol { m } {  
2171   \stex_get_symbol:n { #1 }  
2172   \exp_args:No  
2173   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }  
2174 }
```

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

symdecl arguments:

```
2175 \keys_define:nn { stex / symdecl } {  
2176   name      .str_set_x:N = \l_stex_symdecl_name_str ,  
2177   local     .bool_set:N = \l_stex_symdecl_local_bool ,  
2178   args      .str_set_x:N = \l_stex_symdecl_args_str ,  
2179   type      .tl_set:N = \l_stex_symdecl_type_tl ,  
2180   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,  
2181   align     .str_set:N = \l_stex_symdecl_align_str , % TODO(?)  
2182   gfc       .str_set:N = \l_stex_symdecl_gfc_str , % TODO(?)  
2183   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)  
2184   def       .tl_set:N = \l_stex_symdecl_definiens_tl ,  
2185   reorder   .str_set_x:N = \l_stex_symdecl_reorder_str ,  
2186   assoc     .choices:nn =  
2187     {bin,binl,binr,pre,conj,pwconj}  
2188     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}  
2189 }  
2190  
2191 \bool_new:N \l_stex_symdecl_make_macro_bool  
2192  
2193 \cs_new_protected:Nn \__stex_symdecl_args:n {  
2194   \str_clear:N \l_stex_symdecl_name_str  
2195   \str_clear:N \l_stex_symdecl_args_str  
2196   \str_clear:N \l_stex_symdecl_deprecate_str  
2197   \str_clear:N \l_stex_symdecl_reorder_str  
2198   \str_clear:N \l_stex_symdecl_assoctype_str  
2199   \bool_set_false:N \l_stex_symdecl_local_bool  
2200   \tl_clear:N \l_stex_symdecl_type_tl  
2201   \tl_clear:N \l_stex_symdecl_definiens_tl  
2202  
2203   \keys_set:nn { stex / symdecl } { #1 }  
2204 }
```

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

```
2205  
2206 \NewDocumentCommand \symdecl { s m O{} } {  
2207   \__stex_symdecl_args:n { #3 }  
2208   \IfBooleanTF #1 {  
2209     \bool_set_false:N \l_stex_symdecl_make_macro_bool  
2210   } {  
2211     \bool_set_true:N \l_stex_symdecl_make_macro_bool  
2212   }  
2213   \stex_symdecl_do:n { #2 }  
2214   \stex_smsmode_do:  
2215 }
```

```

2216
2217 \cs_new_protected:Nn \stex_symdecl_do:nn {
2218   \__stex_symdecl_args:n{#1}
2219   \bool_set_false:N \l_stex_symdecl_make_macro_bool
2220   \stex_symdecl_do:n{#2}
2221 }
2222
2223 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

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

```

2224 \cs_new_protected:Nn \stex_symdecl_do:n {
2225   \stex_if_in_module:F {
2226     % TODO throw error? some default namespace?
2227   }
2228
2229   \str_if_empty:NT \l_stex_symdecl_name_str {
2230     \str_set:Nx \l_stex_symdecl_name_str { #1 }
2231   }
2232
2233   \prop_if_exist:cT { l_stex_symdecl_
2234     \l_stex_current_module_str ?
2235     \l_stex_symdecl_name_str
2236     _prop
2237   }{
2238     % TODO throw error (beware of circular dependencies)
2239   }
2240
2241   \prop_clear:N \l_tmpa_prop
2242   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
2243   \seq_clear:N \l_tmpa_seq
2244   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
2245   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
2246
2247   \str_if_empty:NT \l_stex_symdecl_deprecate_str {
2248     \str_if_empty:NF \l_stex_module_deprecate_str {
2249       \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
2250     }
2251   }
2252   \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
2253
2254   \exp_args:No \stex_add_constant_to_current_module:n {
2255     \l_stex_symdecl_name_str
2256   }
2257
2258   % arity/args
2259   \int_zero:N \l_tmpb_int
2260
2261   \bool_set_true:N \l_tmpa_bool
2262   \str_map_inline:Nn \l_stex_symdecl_args_str {
2263     \token_case_meaning:NnF ##1 {
2264       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2265       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }

```



```

2266     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2267     {\tl_to_str:n a} {
2268         \bool_set_false:N \l_tmpa_bool
2269         \int_incr:N \l_tmpb_int
2270     }
2271     {\tl_to_str:n B} {
2272         \bool_set_false:N \l_tmpa_bool
2273         \int_incr:N \l_tmpb_int
2274     }
2275     ){
2276         \msg_error:nnxx{stex}{error/wrongargs}{
2277             \l_stex_current_module_str ?
2278             \l_stex_symdecl_name_str
2279         }{##1}
2280     }
2281 }
2282 \bool_if:NTF \l_tmpa_bool {
2283     % possibly numeric
2284     \str_if_empty:NTF \l_stex_symdecl_args_str {
2285         \prop_put:Nnn \l_tmpa_prop { args } {}
2286         \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2287     }{
2288         \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
2289         \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2290         \str_clear:N \l_tmpa_str
2291         \int_step_inline:nn \l_tmpa_int {
2292             \str_put_right:Nn \l_tmpa_str i
2293         }
2294         \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
2295     }
2296 } {
2297     \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2298     \prop_put:Nnx \l_tmpa_prop { arity }
2299     { \str_count:N \l_stex_symdecl_args_str }
2300 }
2301 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2302
2303 \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
2304     \prop_put:Nnx \l_tmpa_prop { defined }{ false }
2305 }{
2306     \prop_put:Nnx \l_tmpa_prop { defined }{ true }
2307 }
2308
2309 % semantic macro
2310
2311 \bool_if:NT \l_stex_symdecl_make_macro_bool {
2312     \exp_args:Nx \stex_do_up_to_module:n {
2313         \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2314             \l_stex_current_module_str ? \l_stex_symdecl_name_str
2315         }}
2316     }
2317 }
2318
2319 \stex_debug:nn{symbols}{New~symbol:~

```

```

2320 \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2321 Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2322 Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
2323 Definiens:~\exp_not:o {\l_stex_symdecl_definiens_tl}
2324 }
2325
2326 % circular dependencies require this:
2327 \stex_if_do_html:T {
2328   \stex_annotate_invisible:nnn {symdecl} {
2329     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2330   } {
2331     \tl_if_empty:NF \l_stex_symdecl_type_tl {
2332       \stex_annotate_invisible:nnn{type}{\l_stex_symdecl_type_tl$}
2333     }
2334     \stex_annotate_invisible:nnn{args}{\prop_item:Nn \l_tmpa_prop { args }}{ }
2335     \stex_annotate_invisible:nnn{macroname}{#1}{ }
2336     \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2337       \stex_annotate_invisible:nnn{definiens}{ }
2338       {\l_stex_symdecl_definiens_tl$}
2339     }
2340     \str_if_empty:NF \l_stex_symdecl_assoc_type_str {
2341       \stex_annotate_invisible:nnn{assoc_type}{\l_stex_symdecl_assoc_type_str}{ }
2342     }
2343     \str_if_empty:NF \l_stex_symdecl_reorder_str {
2344       \stex_annotate_invisible:nnn{reorderargs}{\l_stex_symdecl_reorder_str}{ }
2345     }
2346   }
2347 }
2348 \prop_if_exist:cF {
2349   \l_stex_symdecl_
2350   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2351   _prop
2352 } {
2353   \bool_if:NTF \l_stex_symdecl_local_bool \stex_do_up_to_module:x \stex_execute_in_module:
2354   \__stex_symdecl_restore_symbol:nnnnnnn
2355   {\l_stex_symdecl_name_str}
2356   { \prop_item:Nn \l_tmpa_prop {args} }
2357   { \prop_item:Nn \l_tmpa_prop {arity} }
2358   { \prop_item:Nn \l_tmpa_prop {assocs} }
2359   { \prop_item:Nn \l_tmpa_prop {defined} }
2360   {\bool_if:NT \l_stex_symdecl_make_macro_bool {#1} }
2361   {\l_stex_current_module_str}
2362 }
2363 }
2364 }
2365 \cs_new_protected:Nn \__stex_symdecl_restore_symbol:nnnnnnn {
2366   \prop_clear:N \l_tmpa_prop
2367   \prop_put:Nnn \l_tmpa_prop { module } { #7 }
2368   \prop_put:Nnn \l_tmpa_prop { name } { #1 }
2369   \prop_put:Nnn \l_tmpa_prop { args } { #2 }
2370   \prop_put:Nnn \l_tmpa_prop { arity } { #3 }
2371   \prop_put:Nnn \l_tmpa_prop { assocs } { #4 }
2372   \prop_put:Nnn \l_tmpa_prop { defined } { #5 }
2373   \tl_if_empty:nF{#6}{

```

```

2374     \tl_set:cx{#6}{\stex_invoke_symbol:n{\detokenize{#7 ? #1}}}
2375   }
2376   \prop_set_eq:cN{l_stex_symdecl_ \detokenize{#7 ? #1} _prop}\l_tmpa_prop
2377   \seq_clear:c{l_stex_symdecl_ \detokenize{#7 ? #1} _notations}
2378 }

```

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

## `\textsymdecl`

```

2379
2380 \keys_define:nn { stex / textsymdecl } {
2381   name      .str_set_x:N = \l__stex_symdecl_name_str ,
2382   type      .tl_set:N    = \l__stex_symdecl_type_tl
2383 }
2384
2385 \cs_new_protected:Nn \_stex_textsymdecl_args:n {
2386   \str_clear:N \l__stex_symdecl_name_str
2387   \tl_clear:N \l__stex_symdecl_type_tl
2388   \keys_set:nn { stex / textsymdecl } { #1 }
2389 }
2390
2391 \NewDocumentCommand \textsymdecl {m O{} m} {
2392   \_stex_textsymdecl_args:n { #2 }
2393   \str_if_empty:NTF \l__stex_symdecl_name_str {
2394     \_stex_symdecl_args:n{name=#1,#2}
2395   }{
2396     \_stex_symdecl_args:n{#2}
2397   }
2398   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2399   \stex_symdecl_do:n{#1-sym}
2400   \stex_execute_in_module:n{
2401     \cs_set_nopar:cpn{#1name}{
2402       \ifvmode\hbox_unpack:N\c_empty_box\fi
2403       \hbox{#3}\xspace
2404     }
2405     \cs_set_nopar:cpn{#1}{
2406       \ifmmode\csname#1-sym\expandafter\endcsname\else
2407       \ifvmode\hbox_unpack:N\c_empty_box\fi
2408       \symref{#1-sym}{\hbox{#3}}\expandafter\xspace
2409       \fi
2410     }
2411   }
2412   \stex_execute_in_module:x{
2413     \_stex_notation_restore_notation:nnnnn
2414     {\l_stex_current_module_str?\tl_if_empty:NTF\l__stex_symdecl_name_str{#1}\l__stex_symdecl
2415     }{0}
2416     {\exp_not:n{\STEXInternalTermMathOMSiiii{\STEXInternalCurrentSymbolStr}{}}{\neginfprec}{
2417       \comp{\hbox{#3}}\STEXInternalSymbolAfterInvokationTL
2418     }}}}
2419   {}
2420 }
2421 \stex_smsmode_do:
2422 }

```

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

`\stex_get_symbol:n`

```
2423 \str_new:N \l_stex_get_symbol_uri_str
2424
2425 \cs_new_protected:Nn \stex_get_symbol:n {
2426   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2427     \tl_set:Nn \l_tmpa_tl { #1 }
2428     \__stex_symdecl_get_symbol_from_cs:
2429   }{
2430     % argument is a string
2431     % is it a command name?
2432     \cs_if_exist:cTF { #1 }{
2433       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2434       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2435       \str_if_empty:NTF \l_tmpa_str {
2436         \exp_args:Nx \cs_if_eq:NNTF {
2437           \tl_head:N \l_tmpa_tl
2438         } \stex_invoke_symbol:n {
2439           \__stex_symdecl_get_symbol_from_cs:
2440         }{
2441           \__stex_symdecl_get_symbol_from_string:n { #1 }
2442         }
2443       } {
2444         \__stex_symdecl_get_symbol_from_string:n { #1 }
2445       }
2446     }{
2447       % argument is not a command name
2448       \__stex_symdecl_get_symbol_from_string:n { #1 }
2449       % \l_stex_all_symbols_seq
2450     }
2451   }
2452   \str_if_eq:eeF {
2453     \prop_item:cn {
2454       l_stex_symdecl\l_stex_get_symbol_uri_str _prop
2455     }{ deprecate }
2456   }{}{
2457     \msg_warning:nxxx{stex}{warning/deprecated}{
2458       Symbol~\l_stex_get_symbol_uri_str
2459     }{
2460       \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{ deprecate }
2461     }
2462   }
2463 }
2464
2465 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2466   \tl_set:Nn \l_tmpa_tl {
2467     \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2468   }
2469   \str_set:Nn \l_tmpa_str { #1 }
2470
2471   %\int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2472
2473   \str_if_in:NnTF \l_tmpa_str ? {
2474     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
2475     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
```

```

2476 \str_set:Nx \l_tmpb_str {\seq_use:Nn \l_tmpa_seq ?}
2477 }{
2478 \str_clear:N \l_tmpb_str
2479 }
2480 \str_if_empty:NTF \l_tmpb_str {
2481 \seq_map_inline:Nn \l_stex_all_modules_seq {
2482 \seq_map_inline:cn{c_stex_module_###1_constants}{
2483 \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2484 \seq_map_break:n{\seq_map_break:n{
2485 \tl_set:Nn \l_tmpa_tl {
2486 \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2487 }
2488 }}
2489 }
2490 }
2491 }
2492 }{
2493 \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpb_str }
2494 \seq_map_inline:Nn \l_stex_all_modules_seq {
2495 \str_if_eq:eeT{ \l_tmpb_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2496 \seq_map_inline:cn{c_stex_module_###1_constants}{
2497 \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2498 \seq_map_break:n{\seq_map_break:n{
2499 \tl_set:Nn \l_tmpa_tl {
2500 \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2501 }
2502 }}
2503 }
2504 }
2505 }
2506 }
2507 }
2508
2509 \l_tmpa_tl
2510 }
2511
2512 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
2513 \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2514 { \tl_tail:N \l_tmpa_tl }
2515 \tl_if_single:NTF \l_tmpa_tl {
2516 \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2517 \exp_after:wN \str_set:Nn \exp_after:wN
2518 \l_stex_get_symbol_uri_str \l_tmpa_tl
2519 }{
2520 % TODO
2521 % tail is not a single group
2522 }
2523 }{
2524 % TODO
2525 % tail is not a single group
2526 }
2527 }

```

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

## 29.2 Notations

```

2528 <@@=stex_notation>

      notation arguments:
2529 \keys_define:nn { stex / notation } {
2530 % lang .tl_set_x:N = \l__stex_notation_lang_str ,
2531 variant .tl_set_x:N = \l__stex_notation_variant_str ,
2532 prec .str_set_x:N = \l__stex_notation_prec_str ,
2533 op .tl_set:N = \l__stex_notation_op_tl ,
2534 primary .bool_set:N = \l__stex_notation_primary_bool ,
2535 primary .default:n = {true} ,
2536 unknown .code:n = \str_set:Nx
2537     \l__stex_notation_variant_str \l_keys_key_str
2538 }
2539
2540 \cs_new_protected:Nn \stex_notation_args:n {
2541 % \str_clear:N \l__stex_notation_lang_str
2542 \str_clear:N \l__stex_notation_variant_str
2543 \str_clear:N \l__stex_notation_prec_str
2544 \tl_clear:N \l__stex_notation_op_tl
2545 \bool_set_false:N \l__stex_notation_primary_bool
2546
2547 \keys_set:nn { stex / notation } { #1 }
2548 }

\notation

2549 \NewDocumentCommand \notation { s m O{}} {
2550 \stex_notation_args:n { #3 }
2551 \tl_clear:N \l_stex_symdecl_definiens_tl
2552 \stex_get_symbol:n { #2 }
2553 \tl_set:Nn \l_stex_notation_after_do_tl {
2554     \__stex_notation_final:
2555     \IfBooleanTF#1{
2556         \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2557     }{}
2558     \stex_smsmode_do:\ignorespacesandpars
2559 }
2560 \stex_notation_do:nnnnn
2561 { \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str_prop} { args } }
2562 { \prop_item:cn { l_stex_symdecl\l_stex_get_symbol_uri_str_prop } { arity } }
2563 { \l__stex_notation_variant_str }
2564 { \l__stex_notation_prec_str }
2565 }
2566 \stex_deactivate_macro:Nn \notation {module-environments}

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

\stex_notation_do:nnnnn

2567 \seq_new:N \l__stex_notation_precedences_seq
2568 \tl_new:N \l__stex_notation_opprec_tl
2569 \int_new:N \l__stex_notation_currarg_int
2570 \tl_new:N \STEXInternalSymbolAfterInvokationTL
2571
2572 \cs_new_protected:Nn \stex_notation_do:nnnnn {

```

```

2573 \let\STEXInternalCurrentSymbolStr\relax
2574 \seq_clear:N \l__stex_notation_precedences_seq
2575 \tl_clear:N \l__stex_notation_opprec_tl
2576 \str_set:Nx \l__stex_notation_args_str { #1 }
2577 \str_set:Nx \l__stex_notation_arity_str { #2 }
2578 \str_set:Nx \l__stex_notation_suffix_str { #3 }
2579 \str_set:Nx \l__stex_notation_prec_str { #4 }
2580
2581 % precedences
2582 \str_if_empty:NTF \l__stex_notation_prec_str {
2583   \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2584     \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2585   }{
2586     \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2587   }
2588 } {
2589   \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2590     \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2591     \int_step_inline:nn { \l__stex_notation_arity_str } {
2592       \exp_args:NNo
2593       \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2594     }
2595   }{
2596     \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2597     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2598       \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2599       \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2600         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2601           \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2602         \seq_map_inline:Nn \l_tmpa_seq {
2603           \seq_put_right:Nn \l_tmpb_seq { ##1 }
2604         }
2605       }
2606     }{
2607       \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2608         \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2609       }{
2610         \tl_set:No \l__stex_notation_opprec_tl { 0 }
2611       }
2612     }
2613   }
2614 }
2615
2616 \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2617 \int_step_inline:nn { \l__stex_notation_arity_str } {
2618   \seq_pop_left:NNTF \l_tmpa_seq \l_tmpb_str {
2619     \exp_args:NNo
2620     \seq_put_right:No \l__stex_notation_precedences_seq {
2621       \l__stex_notation_opprec_tl
2622     }
2623   }
2624 }
2625 \tl_clear:N \l__stex_notation_dummyargs_tl
2626

```

```

2627 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2628   \exp_args:NNe
2629   \cs_set:Npn \l_stex_notation_macrocode_cs {
2630     \STEXInternalTermMathOMSiiii { \STEXInternalCurrentSymbolStr }
2631     { \l__stex_notation_suffix_str }
2632     { \l__stex_notation_opprec_tl }
2633     { \exp_not:n { #5 } }
2634   }
2635   \l_stex_notation_after_do_tl
2636 }{
2637   \str_if_in:NnTF \l__stex_notation_args_str b {
2638     \exp_args:Nne \use:nn
2639     {
2640       \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2641       \cs_set:Npn \l__stex_notation_arity_str } { {
2642         \STEXInternalTermMathOMBiiii { \STEXInternalCurrentSymbolStr }
2643         { \l__stex_notation_suffix_str }
2644         { \l__stex_notation_opprec_tl }
2645         { \exp_not:n { #5 } }
2646       }}
2647   }{
2648     \str_if_in:NnTF \l__stex_notation_args_str B {
2649       \exp_args:Nne \use:nn
2650       {
2651         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2652         \cs_set:Npn \l__stex_notation_arity_str } { {
2653           \STEXInternalTermMathOMBiiii { \STEXInternalCurrentSymbolStr }
2654           { \l__stex_notation_suffix_str }
2655           { \l__stex_notation_opprec_tl }
2656           { \exp_not:n { #5 } }
2657         } }
2658     }{
2659       \exp_args:Nne \use:nn
2660       {
2661         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2662         \cs_set:Npn \l__stex_notation_arity_str } { {
2663           \STEXInternalTermMathOMAiiii { \STEXInternalCurrentSymbolStr }
2664           { \l__stex_notation_suffix_str }
2665           { \l__stex_notation_opprec_tl }
2666           { \exp_not:n { #5 } }
2667         } }
2668     }
2669   }
2670
2671   \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2672   \int_zero:N \l__stex_notation_currarg_int
2673   \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2674   \__stex_notation_arguments:
2675 }
2676 }

```

*(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



```

2677 \cs_new_protected:Nn \__stex_notation_arguments: {
2678   \int_incr:N \l__stex_notation_currarg_int
2679   \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2680     \l_stex_notation_after_do_tl
2681   }{
2682     \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2683     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2684     \str_if_eq:VnTF \l_tmpa_str a {
2685       \__stex_notation_argument_assoc:nn{a}
2686     }{
2687       \str_if_eq:VnTF \l_tmpa_str B {
2688         \__stex_notation_argument_assoc:nn{B}
2689       }{
2690         \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpb_str
2691         \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2692           { \STEXInternalTermMathArgiii
2693             { \l_tmpa_str\int_use:N \l__stex_notation_currarg_int }
2694             { \l_tmpb_str }
2695             { ####\int_use:N \l__stex_notation_currarg_int }
2696           }
2697         }
2698         \__stex_notation_arguments:
2699       }
2700     }
2701   }
2702 }

```

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

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

```

2703 \cs_new_protected:Nn \__stex_notation_argument_assoc:nn {
2704
2705   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2706     {\l__stex_notation_arity_str}{
2707       #2
2708     }
2709   \int_zero:N \l_tmpa_int
2710   \tl_clear:N \l_tmpa_tl
2711   \str_map_inline:Nn \l__stex_notation_args_str {
2712     \int_incr:N \l_tmpa_int
2713     \tl_put_right:Nx \l_tmpa_tl {
2714       \str_if_eq:nnTF {##1}{a}{ {} }{
2715         \str_if_eq:nnTF {##1}{B}{ {} }{
2716           {\_stex_term_arg:nn{##1}\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa
2717         }
2718       }
2719     }
2720   }
2721   \exp_after:wN\exp_after:wN\exp_after:wN \def
2722   \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2723   \exp_after:wN\exp_after:wN\exp_after:wN ##
2724   \exp_after:wN\exp_after:wN\exp_after:wN 1
2725   \exp_after:wN\exp_after:wN\exp_after:wN ##
2726   \exp_after:wN\exp_after:wN\exp_after:wN 2

```

```

2727 \exp_after:wN\exp_after:wN\exp_after:wN {
2728   \exp_after:wN \exp_after:wN \exp_after:wN
2729   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2730     \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2731   }
2732 }
2733
2734 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2735 \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
2736   \STEXInternalTermMathAssocArgiiii
2737   { #1\int_use:N \l__stex_notation_currarg_int }
2738   { \l_tmpa_str }
2739   { ####\int_use:N \l__stex_notation_currarg_int }
2740   { \l_tmpa_cs {####1} {####2} }
2741 } }
2742 \__stex_notation_arguments:
2743 }

```

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

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

```

2744 \cs_new_protected:Nn \__stex_notation_restore_notation:nnnnn {
2745   \cs_generate_from_arg_count:cNnn{stex_notation_\detokenize{#1} \c_hash_str \detokenize{#2}}
2746   \cs_set_nopar:Npn {#3}{#4}
2747   \tl_if_empty:nF {#5}{
2748     \tl_set:cn{stex_op_notation_\detokenize{#1} \c_hash_str \detokenize{#2}_cs}{\comp{ #5 }
2749   }
2750   \seq_if_exist:cT { l_stex_symdecl_\detokenize{#1} _notations }{
2751     \seq_put_right:cx { l_stex_symdecl_\detokenize{#1} _notations } { \detokenize{#2} }
2752   }
2753 }
2754
2755 \cs_new_protected:Nn \__stex_notation_final: {
2756
2757   \stex_execute_in_module:x {
2758     \__stex_notation_restore_notation:nnnnn
2759     {\l_stex_get_symbol_uri_str}
2760     {\l__stex_notation_suffix_str}
2761     {\l__stex_notation_arity_str}
2762     {
2763       \exp_after:wN \exp_after:wN \exp_after:wN
2764       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2765       { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \STEXInt
2766     }
2767     {\exp_args:No \exp_not:n \l__stex_notation_op_tl }
2768   }
2769
2770   \stex_debug:nn{symbols}{
2771     Notation~\l__stex_notation_suffix_str
2772     ~for~\l_stex_get_symbol_uri_str^^J
2773     Operator~precedence:~\l__stex_notation_opprec_tl^^J
2774     Argument~precedences:~
2775     \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2776     Notation: \cs_meaning:c {

```

```

2777     stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2778     \l__stex_notation_suffix_str
2779     _cs
2780 }
2781 }
2782 % HTML annotations
2783 \stex_if_do_html:T {
2784   \stex_annotate_invisible:nnn { notation }
2785   { \l_stex_get_symbol_uri_str } {
2786     \stex_annotate_invisible:nnn { notationfragment }
2787     { \l__stex_notation_suffix_str }{}
2788     \stex_annotate_invisible:nnn { precedence }
2789     { \l__stex_notation_prec_str }{}
2790
2791     \int_zero:N \l_tmpa_int
2792     \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2793     \tl_clear:N \l_tmpa_tl
2794     \int_step_inline:nn { \l__stex_notation_arity_str }{
2795       \int_incr:N \l_tmpa_int
2796       \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2797       \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_rema
2798       \str_if_eq:VnTF \l_tmpb_str a {
2799         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2800           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2801           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2802         } }
2803       }{
2804         \str_if_eq:VnTF \l_tmpb_str B {
2805           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2806             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2807             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2808           } }
2809         }{
2810           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2811             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
2812           } }
2813         }
2814       }
2815     }
2816     \stex_annotate_invisible:nnn { notationcomp }{}{
2817       \str_set:Nx \STEXInternalCurrentSymbolStr {\l_stex_get_symbol_uri_str }
2818       $ \exp_args:Nno \use:nn { \use:c {
2819         stex_notation_ \STEXInternalCurrentSymbolStr
2820         \c_hash_str \l__stex_notation_suffix_str _cs
2821       } } { \l_tmpa_tl } $
2822     }
2823     \tl_if_empty:NF \l__stex_notation_op_tl {
2824       \stex_annotate_invisible:nnn { notationopcomp }{}{
2825         $\l__stex_notation_op_tl$
2826       }
2827     }
2828   }
2829 }
2830 }

```

(End definition for `\_stex_notation_final:`)

`\setnotation`

```

2831 \keys_define:nn { stex / setnotation } {
2832   % lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2833   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2834   unknown .code:n      = \str_set:Nx
2835     \l__stex_notation_variant_str \l_keys_key_str
2836 }
2837
2838 \cs_new_protected:Nn \stex_setnotation_args:n {
2839   % \str_clear:N \l__stex_notation_lang_str
2840   \str_clear:N \l__stex_notation_variant_str
2841   \keys_set:nn { stex / setnotation } { #1 }
2842 }
2843
2844 \cs_new_protected:Nn \__stex_notation_setnotation:nn {
2845   \seq_if_exist:cT{l_stex_symdecl_#1_notations}{
2846     \seq_remove_all:cn { l_stex_symdecl_#1_notations }{ #2 }
2847     \seq_put_left:cn { l_stex_symdecl_#1_notations }{ #2 }
2848   }
2849 }
2850
2851 \cs_new_protected:Nn \stex_setnotation:n {
2852   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1_notations }
2853     { \l__stex_notation_variant_str }{
2854     \stex_execute_in_module:x{ \__stex_notation_setnotation:nn {#1}{\l__stex_notation_vari
2855     \stex_debug:nn {notations}{
2856       Setting~default~notation~
2857       {\l__stex_notation_variant_str }~for~
2858       #1 \\
2859       \expandafter\meaning\csname
2860       l_stex_symdecl_#1_notations\endcsname
2861     }
2862   }{
2863     \msg_error:nxxx{stex}{unknownnotation}{\l__stex_notation_variant_str}{#1}
2864   }
2865 }
2866
2867 \NewDocumentCommand \setnotation {m m} {
2868   \stex_get_symbol:n { #1 }
2869   \stex_setnotation_args:n { #2 }
2870   \stex_setnotation:n{\l_stex_get_symbol_uri_str}
2871   \stex_smsmode_do:\ignorespacesandpars
2872 }
2873
2874 \cs_new_protected:Nn \stex_copy_notations:nn {
2875   \stex_debug:nn {notations}{
2876     Copying~notations~from~#2~to~#1\\
2877     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
2878   }
2879   \tl_clear:N \l_tmpa_tl
2880   \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2881     \tl_put_right:Nn \l_tmpa_tl { {##### #1} }

```

```

2882 }
2883 \seq_map_inline:cn {l_stex_symdecl_#2_notations}{\begingroup
2884   \stex_debug:nn{Here}{Here:~##1}
2885   \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2886   \edef \l_tmpa_tl {
2887     \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2888     \exp_after:wN\exp_after:wN\exp_after:wN {
2889       \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2890     }
2891   }
2892
2893   \exp_after:wN \def \exp_after:wN \l_tmpa_tl
2894   \exp_after:wN #####\exp_after:wN 1 \exp_after:wN #####\exp_after:wN 2
2895   \exp_after:wN { \l_tmpa_tl }
2896
2897   \edef \l_tmpa_tl {
2898     \exp_after:wN \exp_not:n \exp_after:wN {
2899       \l_tmpa_tl {##### 1}{##### 2}
2900     }
2901   }
2902
2903   \stex_debug:nn{Here}{Here:~\expandafter\detokenize\expandafter{\l_tmpa_tl}}
2904
2905   \stex_execute_in_module:x {
2906     \__stex_notation_restore_notation:nnnnn
2907     {#1}{##1}
2908     { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }
2909     { \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl} }
2910     {
2911       \cs_if_exist:cT{stex_op_notation_ #2\c_hash_str ##1 _cs}{
2912         \exp_args:NNo\exp_args:No\exp_not:n{\csname stex_op_notation_ #2\c_hash_str ##1
2913       }
2914     }
2915   }\endgroup
2916 }
2917 }
2918
2919 \NewDocumentCommand \copynotation {m m} {
2920   \stex_get_symbol:n { #1 }
2921   \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2922   \stex_get_symbol:n { #2 }
2923   \exp_args:Noo
2924   \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2925   \stex_smsmode_do:\ignorespacesandpars
2926 }
2927

```

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

## \symdef

```

2928 \keys_define:nn { stex / symdef } {
2929   name .str_set_x:N = \l_stex_symdecl_name_str ,
2930   local .bool_set:N = \l_stex_symdecl_local_bool ,
2931   args .str_set_x:N = \l_stex_symdecl_args_str ,

```

```

2932 type .tl_set:N = \l_stex_symdecl_type_tl ,
2933 def .tl_set:N = \l_stex_symdecl_definiens_tl ,
2934 reorder .str_set_x:N = \l_stex_symdecl_reorder_str ,
2935 op .tl_set:N = \l__stex_notation_op_tl ,
2936 % lang .str_set_x:N = \l__stex_notation_lang_str ,
2937 variant .str_set_x:N = \l__stex_notation_variant_str ,
2938 prec .str_set_x:N = \l__stex_notation_prec_str ,
2939 assoc .choices:nn =
2940 {bin,binl,binr,pre,conj,pwconj}
2941 {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
2942 unknown .code:n = \str_set:Nx
2943 \l__stex_notation_variant_str \l_keys_key_str
2944 }
2945
2946 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2947 \str_clear:N \l_stex_symdecl_name_str
2948 \str_clear:N \l_stex_symdecl_args_str
2949 \str_clear:N \l_stex_symdecl_assoctype_str
2950 \str_clear:N \l_stex_symdecl_reorder_str
2951 \bool_set_false:N \l_stex_symdecl_local_bool
2952 \tl_clear:N \l_stex_symdecl_type_tl
2953 \tl_clear:N \l_stex_symdecl_definiens_tl
2954 % \str_clear:N \l__stex_notation_lang_str
2955 \str_clear:N \l__stex_notation_variant_str
2956 \str_clear:N \l__stex_notation_prec_str
2957 \tl_clear:N \l__stex_notation_op_tl
2958
2959 \keys_set:nn { stex / symdef } { #1 }
2960 }
2961
2962 \NewDocumentCommand \symdef { m O{} } {
2963 \__stex_notation_symdef_args:n { #2 }
2964 \bool_set_true:N \l_stex_symdecl_make_macro_bool
2965 \stex_symdecl_do:n { #1 }
2966 \tl_set:Nn \l_stex_notation_after_do_tl {
2967 \__stex_notation_final:
2968 \stex_smsmode_do:\ignorespacesandpars
2969 }
2970 \str_set:Nx \l_stex_get_symbol_uri_str {
2971 \l_stex_current_module_str ? \l_stex_symdecl_name_str
2972 }
2973 \exp_args:Nx \stex_notation_do:nnnnn
2974 { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } }
2975 { \prop_item:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } }
2976 { \l__stex_notation_variant_str }
2977 { \l__stex_notation_prec_str }
2978 }
2979 \stex_deactivate_macro:Nn \symdef {module~environments}

```

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

## 29.3 Variables

```

2980 <@@=stex_variables>

```

```

2981
2982 \keys_define:nn { stex / vardef } {
2983   name .str_set_x:N = \l__stex_variables_name_str ,
2984   args .str_set_x:N = \l__stex_variables_args_str ,
2985   type .tl_set:N     = \l__stex_variables_type_tl ,
2986   def .tl_set:N      = \l__stex_variables_def_tl ,
2987   op .tl_set:N       = \l__stex_variables_op_tl ,
2988   prec .str_set_x:N  = \l__stex_variables_prec_str ,
2989   reorder .str_set_x:N = \l__stex_variables_reorder_str ,
2990   assoc .choices:nn =
2991     {bin,binl,binr,pre,conj,pwconj}
2992     {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
2993   bind .choices:nn =
2994     {forall,exists}
2995     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2996 }
2997
2998 \cs_new_protected:Nn \__stex_variables_args:n {
2999   \str_clear:N \l__stex_variables_name_str
3000   \str_clear:N \l__stex_variables_args_str
3001   \str_clear:N \l__stex_variables_prec_str
3002   \str_clear:N \l__stex_variables_assoctype_str
3003   \str_clear:N \l__stex_variables_reorder_str
3004   \str_clear:N \l__stex_variables_bind_str
3005   \tl_clear:N \l__stex_variables_type_tl
3006   \tl_clear:N \l__stex_variables_def_tl
3007   \tl_clear:N \l__stex_variables_op_tl
3008
3009   \keys_set:nn { stex / vardef } { #1 }
3010 }
3011
3012 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{}} {
3013   \__stex_variables_args:n {#2}
3014   \str_if_empty:NT \l__stex_variables_name_str {
3015     \str_set:Nx \l__stex_variables_name_str { #1 }
3016   }
3017   \prop_clear:N \l_tmpa_prop
3018   \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
3019
3020   \int_zero:N \l_tmpb_int
3021   \bool_set_true:N \l_tmpa_bool
3022   \str_map_inline:Nn \l__stex_variables_args_str {
3023     \token_case_meaning:NnF ##1 {
3024       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
3025       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
3026       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
3027       {\tl_to_str:n a} {
3028         \bool_set_false:N \l_tmpa_bool
3029         \int_incr:N \l_tmpb_int
3030       }
3031       {\tl_to_str:n B} {
3032         \bool_set_false:N \l_tmpa_bool
3033         \int_incr:N \l_tmpb_int
3034       }

```

```

3035     }{
3036         \msg_error:nnxx{stex}{error/wrongargs}{
3037             variable~\l__stex_variables_name_str
3038         }{##1}
3039     }
3040 }
3041 \bool_if:NTF \l_tmpa_bool {
3042     % possibly numeric
3043     \str_if_empty:NTF \l__stex_variables_args_str {
3044         \prop_put:Nnn \l_tmpa_prop { args } {}
3045         \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
3046     }{
3047         \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
3048         \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
3049         \str_clear:N \l_tmpa_str
3050         \int_step_inline:nn \l_tmpa_int {
3051             \str_put_right:Nn \l_tmpa_str i
3052         }
3053         \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
3054         \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3055     }
3056 } {
3057     \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3058     \prop_put:Nnx \l_tmpa_prop { arity }
3059     { \str_count:N \l__stex_variables_args_str }
3060 }
3061 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
3062 \tl_set:cx { #1 }{ \stex_invoke_variable:n { \l__stex_variables_name_str } }
3063
3064 \prop_set_eq:cN { l_stex_variable_\l__stex_variables_name_str _prop } \l_tmpa_prop
3065
3066 \tl_if_empty:NF \l__stex_variables_op_tl {
3067     \cs_set:cpx {
3068         stex_var_op_notation_\l__stex_variables_name_str _cs
3069     } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
3070 }
3071
3072 \tl_set:Nn \l_stex_notation_after_do_tl {
3073     \exp_args:Nne \use:nn {
3074         \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str _cs }
3075         \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
3076     } {{
3077         \exp_after:wN \exp_after:wN \exp_after:wN
3078         \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
3079         { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \STEXInter
3080     }}
3081 \stex_if_do_html:T {
3082     \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
3083         \stex_annotate_invisible:nnn { precedence }
3084         { \l__stex_variables_prec_str }{}
3085         \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{-}{\l__
3086         \stex_annotate_invisible:nnn{args}{\l__stex_variables_args_str }{}
3087         \stex_annotate_invisible:nnn{macroname}{#1}{-}
3088         \tl_if_empty:NF \l__stex_variables_def_tl {

```



```

3089         \stex_annotate_invisible:nnn{definiens}{}
3090         {${\l__stex_variables_def_tl$}
3091     }
3092     \str_if_empty:NF \l__stex_variables_assoctype_str {
3093         \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{}
3094     }
3095     \str_if_empty:NF \l__stex_variables_reorder_str {
3096         \stex_annotate_invisible:nnn{reorderargs}{\l__stex_variables_reorder_str}{}
3097     }
3098     \int_zero:N \l_tmpa_int
3099     \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
3100     \tl_clear:N \l_tmpa_tl
3101     \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
3102         \int_incr:N \l_tmpa_int
3103         \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
3104         \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables
3105         \str_if_eq:VnTF \l_tmpb_str a {
3106             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3107                 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3108                 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3109             } }
3110         }{
3111             \str_if_eq:VnTF \l_tmpb_str B {
3112                 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3113                     \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3114                     \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3115                 } }
3116             }{
3117                 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3118                     \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
3119                 } }
3120             }
3121         }
3122     }
3123     \stex_annotate_invisible:nnn { notationcomp }{}{
3124         \str_set:Nx \STEXInternalCurrentSymbolStr {var://\l__stex_variables_name_str }
3125         $ \exp_args:Nno \use:nn { \use:c {
3126             stex_var_notation_\l__stex_variables_name_str_cs
3127         } } { \l_tmpa_tl } $
3128     }
3129     \tl_if_empty:NF \l__stex_variables_op_tl {
3130         \stex_annotate_invisible:nnn { notationopcomp }{}{
3131             ${\l__stex_variables_op_tl$}
3132         }
3133     }
3134 }
3135 \str_if_empty:NF \l__stex_variables_bind_str {
3136     \stex_annotate_invisible:nnn {bindtype}{\l__stex_variables_bind_str,\l__stex_variab
3137 }
3138 }\ignorespacesandpars
3139 }
3140
3141 \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
3142 }

```

```

3143
3144 \cs_new:Nn \_stex_reset:N {
3145   \tl_if_exist:NTF #1 {
3146     \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
3147   }{
3148     \let \exp_not:N #1 \exp_not:N \undefined
3149   }
3150 }
3151
3152 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
3153   \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
3154   \exp_args:Nnx \use:nn {
3155     % TODO
3156     \stex_annotate_invisible:nnn {vardecl}{\clist_use:Nn\l__stex_variables_names,}{
3157       #2
3158     }
3159   }{
3160     \_stex_reset:N \varnot
3161     \_stex_reset:N \vartype
3162     \_stex_reset:N \vardefi
3163   }
3164 }
3165
3166 \NewDocumentCommand \vardef { s } {
3167   \IfBooleanTF#1 {
3168     \__stex_variables_do_complex:nn
3169   }{
3170     \__stex_variables_do_simple:nnn
3171   }
3172 }
3173
3174 \NewDocumentCommand \svar { 0{} m }{
3175   \tl_if_empty:nTF {#1}{
3176     \str_set:Nn \l_tmpa_str { #2 }
3177   }{
3178     \str_set:Nn \l_tmpa_str { #1 }
3179   }
3180   \_stex_term_omv:nn {
3181     var://\l_tmpa_str
3182   }{
3183     \exp_args:Nnx \use:nn {
3184       \def\comp{\_varcomp}
3185       \str_set:Nx \STEXInternalCurrentSymbolStr { var://\l_tmpa_str }
3186       \comp{ #2 }
3187     }{
3188       \_stex_reset:N \comp
3189       \_stex_reset:N \STEXInternalCurrentSymbolStr
3190     }
3191   }
3192 }
3193
3194
3195
3196 \keys_define:nn { stex / varseq } {

```

```

3197 name .str_set_x:N = \l__stex_variables_name_str ,
3198 args .int_set:N = \l__stex_variables_args_int ,
3199 type .tl_set:N = \l__stex_variables_type_tl ,
3200 mid .tl_set:N = \l__stex_variables_mid_tl ,
3201 bind .choices:nn =
3202     {forall,exists}
3203     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
3204 }
3205
3206 \cs_new_protected:Nn \__stex_variables_seq_args:n {
3207     \str_clear:N \l__stex_variables_name_str
3208     \int_set:Nn \l__stex_variables_args_int 1
3209     \tl_clear:N \l__stex_variables_type_tl
3210     \str_clear:N \l__stex_variables_bind_str
3211
3212     \keys_set:nn { stex / varseq } { #1 }
3213 }
3214
3215 \NewDocumentCommand \varseq {m O{} m m m}{
3216     \__stex_variables_seq_args:n { #2 }
3217     \str_if_empty:NT \l__stex_variables_name_str {
3218         \str_set:Nx \l__stex_variables_name_str { #1 }
3219     }
3220     \prop_clear:N \l_tmpa_prop
3221     \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
3222
3223     \seq_set_from_clist:Nn \l_tmpa_seq {#3}
3224     \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
3225         \msg_error:nnxx{stex}{error/seqlength}
3226         {\int_use:N \l__stex_variables_args_int}
3227         {\seq_count:N \l_tmpa_seq}
3228     }
3229     \seq_set_from_clist:Nn \l_tmpb_seq {#4}
3230     \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
3231         \msg_error:nnxx{stex}{error/seqlength}
3232         {\int_use:N \l__stex_variables_args_int}
3233         {\seq_count:N \l_tmpb_seq}
3234     }
3235     \prop_put:Nnn \l_tmpa_prop {starts} {#3}
3236     \prop_put:Nnn \l_tmpa_prop {ends} {#4}
3237
3238     \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str_cs}
3239     \cs_set:Npn {\int_use:N \l__stex_variables_args_int} { #5 }
3240
3241     \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str_cs}}
3242     \int_step_inline:nn \l__stex_variables_args_int {
3243         \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
3244     }
3245     \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
3246     \tl_put_right:Nn \l_tmpa_tl {\,\ellipses,}
3247     \tl_if_empty:NF \l__stex_variables_mid_tl {
3248         \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
3249         \tl_put_right:Nn \l_tmpa_tl {\,\ellipses,}
3250     }

```

```

3251 \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3252 \int_step_inline:nn \l__stex_variables_args_int {
3253   \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
3254 }
3255 \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
3256 \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
3257
3258
3259 \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl
3260
3261 \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}
3262
3263 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3264
3265 \int_step_inline:nn \l__stex_variables_args_int {
3266   \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
3267     \STEXInternalTermMathArgiii{i##1}{0}{\exp_not:n{####}##1}
3268   }}
3269 }
3270
3271 \tl_set:Nx \l_tmpa_tl {
3272   \STEXInternalTermMathOMaiiii { varseq://\l__stex_variables_name_str}{0}{
3273     \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
3274   }
3275 }
3276
3277 \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \STEXInternalSymbolAfterInvokationTL} }
3278
3279 \exp_args:Nno \use:nn {
3280 \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
3281   \cs_set:Npn {\int_use:N \l__stex_variables_args_int}{\l_tmpa_tl}
3282
3283 \stex_debug:nn{sequences}{New~Sequence:~
3284   \expandafter\meaning\csname stex_varseq_\l__stex_variables_name_str _cs\endcsname\\~\\
3285   \prop_to_keyval:N \l_tmpa_prop
3286 }
3287 \stex_if_do_html:T{\stex_annotate_invisible:nnn{varseq}{\l__stex_variables_name_str}{
3288   \tl_if_empty:NF \l__stex_variables_type_tl {
3289     \stex_annotate:nnn {type}{\l__stex_variables_type_tl$}
3290   }
3291   \stex_annotate:nnn {args}{\int_use:N \l__stex_variables_args_int}{}
3292   \str_if_empty:NF \l__stex_variables_bind_str {
3293     \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{}
3294   }
3295   \stex_annotate:nnn{startindex}{\l__stex_variables_startindex}{\l__stex_variables_startindex$}
3296   \stex_annotate:nnn{endindex}{\l__stex_variables_endindex}{\l__stex_variables_endindex$}
3297
3298   \tl_clear:N \l_tmpa_tl
3299   \int_step_inline:nn \l__stex_variables_args_int {
3300     \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3301       \stex_annotate:nnn{argmarker}{##1}{}
3302     } }
3303   }
3304   \stex_annotate_invisible:nnn { notationcomp }{}{

```

```

3305     \str_set:Nx \STEXInternalCurrentSymbolStr {varseq://\l__stex_variables_name_str }
3306     $ \exp_args:Nno \use:nn { \use:c {
3307         stex_varseq_\l__stex_variables_name_str _cs
3308     } } { \l_tmpa_tl } $
3309 }
3310 \stex_annotate_invisible:nnn { notationopcomp }{}{
3311     $ \prop_item:Nn \l_tmpa_prop { notation } $
3312 }
3313
3314 }}
3315
3316 \prop_set_eq:cN {stex_varseq_\l__stex_variables_name_str _prop}\l_tmpa_prop
3317 \ignorespacesandpars
3318 }
3319
3320 \end{package}

```

## Chapter 30

# STEX -Terms Implementation

```
3321 <*package>
3322
3323 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
3324
3325 <@@=stex_terms>
3326
3327 Warnings and error messages
3328 \msg_new:nnn{stex}{error/nonotation}{
3329   Symbol~#1~invoked,~but~has~no~notation#2!
3330 }
3331 \msg_new:nnn{stex}{error/notationarg}{
3332   Error~in~parsing~notation~#1
3333 }
3334 \msg_new:nnn{stex}{error/noop}{
3335   Symbol~#1~has~no~operator~notation~for~notation~#2
3336 }
3337 \msg_new:nnn{stex}{error/notallowed}{
3338   Symbol~invocation~#1~not~allowed~in~notation~component~of~#2
3339 }
3340 \msg_new:nnn{stex}{error/doubleargument}{
3341   Argument~#1~of~symbol~#2~already~assigned
3342 }
3343 \msg_new:nnn{stex}{error/overarity}{
3344   Argument~#1~invalid~for~symbol~#2~with~arity~#3
3345 }
```

### 30.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
3345
3346
3347 \bool_new:N \l_stex_allow_semantic_bool
3348 \bool_set_true:N \l_stex_allow_semantic_bool
3349
```

```

3350 \cs_new_protected:Nn \stex_invoke_symbol:n {
3351   \ifvmode\indent\fi
3352   \bool_if:NTF \l_stex_allow_semantic_bool {
3353     \str_if_eq:eeF {
3354       \prop_item:cn {
3355         l_stex_symdecl_#1_prop
3356       }{ deprecate }
3357     }{}{
3358       \msg_warning:nxxx{stex}{warning/deprecated}{
3359         Symbol~#1
3360       }{
3361         \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
3362       }
3363     }
3364     \if_mode_math:
3365       \exp_after:wN \__stex_terms_invoke_math:n
3366     \else:
3367       \exp_after:wN \__stex_terms_invoke_text:n
3368     \fi: { #1 }
3369   }{
3370     \msg_error:nxxx{stex}{error/notallowed}{#1}{\STEXInternalCurrentSymbolStr}
3371   }
3372 }
3373
3374 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
3375   \peek_charcode_remove:NTF ! {
3376     \__stex_terms_invoke_op_custom:nn {#1}
3377   }{
3378     \__stex_terms_invoke_custom:nn {#1}
3379   }
3380 }
3381
3382 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
3383   \peek_charcode_remove:NTF ! {
3384     % operator
3385     \peek_charcode_remove:NTF * {
3386       % custom op
3387       \__stex_terms_invoke_op_custom:nn {#1}
3388     }{
3389       % op notation
3390       \peek_charcode:NTF [ {
3391         \__stex_terms_invoke_op_notation:nw {#1}
3392       }{
3393         \__stex_terms_invoke_op_notation:nw {#1}[]
3394       }
3395     }
3396   }{
3397     \peek_charcode_remove:NTF * {
3398       \__stex_terms_invoke_custom:nn {#1}
3399       % custom
3400     }{
3401       % normal
3402       \peek_charcode:NTF [ {
3403         \__stex_terms_invoke_notation:nw {#1}

```

```

3404     }{
3405         \__stex_terms_invoke_notation:nw {#1}[]
3406     }
3407 }
3408 }
3409 }
3410
3411
3412 \cs_new_protected:Nn \__stex_terms_invoke_op_custom:nn {
3413     \exp_args:Nnx \use:nn {
3414         \def\comp{\_comp}
3415         \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3416         \bool_set_false:N \l_stex_allow_semantic_bool
3417         \stex_term_oms:nnn {#1}{#1 \c_hash_str CUSTOM-}{
3418             \comp{ #2 }
3419         }
3420     }{
3421         \stex_reset:N \comp
3422         \stex_reset:N \STEXInternalCurrentSymbolStr
3423         \bool_set_true:N \l_stex_allow_semantic_bool
3424     }
3425 }
3426
3427 \keys_define:nn { stex / terms } {
3428     % lang .tl_set_x:N = \l_stex_notation_lang_str ,
3429     variant .tl_set_x:N = \l_stex_notation_variant_str ,
3430     unknown .code:n      = \str_set:Nx
3431         \l_stex_notation_variant_str \l_keys_key_str
3432 }
3433
3434 \cs_new_protected:Nn \__stex_terms_args:n {
3435     % \str_clear:N \l_stex_notation_lang_str
3436     \str_clear:N \l_stex_notation_variant_str
3437
3438     \keys_set:nn { stex / terms } { #1 }
3439 }
3440
3441 \cs_new_protected:Nn \stex_find_notation:nn {
3442     \__stex_terms_args:n { #2 }
3443     \seq_if_empty:cTF {
3444         l_stex_symdecl_ #1 _notations
3445     } {
3446         \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
3447     } {
3448         \str_if_empty:NTF \l_stex_notation_variant_str {
3449             \seq_get_left:cN {l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
3450         }{
3451             \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
3452                 \l_stex_notation_variant_str
3453             }{
3454                 % \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3455             }{
3456                 \msg_error:nnxx{stex}{error/nonotation}{#1}{
3457                     ~\l_stex_notation_variant_str

```



```

3458     }
3459   }
3460 }
3461 }
3462 }
3463
3464 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3465   \exp_args:Nnx \use:nn {
3466     \def\comp{\_comp}
3467     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3468     \stex_find_notation:nn { #1 }{ #2 }
3469     \bool_set_false:N \l_stex_allow_semantic_bool
3470     \cs_if_exist:cTF {
3471       stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3472     }{
3473       \_stex_term_oms:nnn { #1 }{
3474         #1 \c_hash_str \l_stex_notation_variant_str
3475       }{
3476         \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3477       }
3478     }{
3479       \int_compare:nNnTF {\prop_item:cn {\l_stex_symdecl_#1_prop}{arity}} = 0{
3480         \cs_if_exist:cTF {
3481           stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3482         }{
3483           \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
3484             \_stex_reset:N \comp
3485             \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
3486             \_stex_reset:N \STEXInternalCurrentSymbolStr
3487             \bool_set_true:N \l_stex_allow_semantic_bool
3488           }
3489           \def\comp{\_comp}
3490           \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3491           \bool_set_false:N \l_stex_allow_semantic_bool
3492           \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3493         }{
3494           \msg_error:nnxx{stex}{error/nonotation}{#1}{
3495             ~\l_stex_notation_variant_str
3496           }
3497         }
3498       }{
3499         \msg_error:nnxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3500       }
3501     }
3502   }{
3503     \_stex_reset:N \comp
3504     \_stex_reset:N \STEXInternalCurrentSymbolStr
3505     \bool_set_true:N \l_stex_allow_semantic_bool
3506   }
3507 }
3508
3509 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {
3510   \stex_find_notation:nn { #1 }{ #2 }
3511   \cs_if_exist:cTF {

```

```

3512     stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3513   }{
3514     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
3515       \_stex_reset:N \comp
3516       \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
3517       \_stex_reset:N \STEXInternalCurrentSymbolStr
3518       \bool_set_true:N \l_stex_allow_semantic_bool
3519     }
3520     \def\comp{\_comp}
3521     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3522     \bool_set_false:N \l_stex_allow_semantic_bool
3523     \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3524   }{
3525     \msg_error:nnxx{stex}{error/nonotation}{#1}{
3526       ~\l_stex_notation_variant_str
3527     }
3528   }
3529 }
3530
3531 \prop_new:N \l__stex_terms_custom_args_prop
3532
3533 \cs_new_protected:Nn \__stex_terms_custom_comp:n { \bool_set_false:N \l_stex_allow_semantic_bool }
3534
3535 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3536   \exp_args:Nnx \use:nn {
3537     \def\comp{\__stex_terms_custom_comp:n}
3538     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3539     \prop_clear:N \l__stex_terms_custom_args_prop
3540     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3541     \prop_get:cnN {
3542       l_stex_symdecl_#1 _prop
3543     } { args } \l_tmpa_str
3544     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3545     \tl_set:Nn \arg { \__stex_terms_arg: }
3546     \str_if_empty:NTF \l_tmpa_str {
3547       \_stex_term_oms:nnn {#1}{#1\c_hash_str CUSTOM-}{\ignorespaces#2}
3548     }{
3549       \str_if_in:NnTF \l_tmpa_str b {
3550         \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
3551       }{
3552         \str_if_in:NnTF \l_tmpa_str B {
3553           \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
3554         }{
3555           \_stex_term_oma:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
3556         }
3557       }
3558     }
3559     % TODO check that all arguments exist
3560   }{
3561     \_stex_reset:N \STEXInternalCurrentSymbolStr
3562     \_stex_reset:N \arg
3563     \_stex_reset:N \comp
3564     \_stex_reset:N \l__stex_terms_custom_args_prop
3565     %\bool_set_true:N \l_stex_allow_semantic_bool

```

```

3566 }
3567 }
3568
3569 \NewDocumentCommand \__stex_terms_arg: { s O{} m}{
3570   \tl_if_empty:nTF {#2}{
3571     \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
3572     \bool_set_true:N \l_tmpa_bool
3573     \bool_do_while:Nn \l_tmpa_bool {
3574       \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int
3575         \int_incr:N \l_tmpa_int
3576       }{
3577         \bool_set_false:N \l_tmpa_bool
3578       }
3579     }
3580   }{
3581     \int_set:Nn \l_tmpa_int { #2 }
3582   }
3583   \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
3584   \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3585     \msg_error:nnxxx{stex}{error/overarity}
3586     {\int_use:N \l_tmpa_int}
3587     {\STEXInternalCurrentSymbolStr}
3588     {\str_count:N \l_tmpa_str}
3589   }
3590   \str_set:Nx \l_tmpa_str {\str_item:Nn \l_tmpa_str \l_tmpa_int}
3591   \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
3592     \bool_lazy_any:nF {
3593       {\str_if_eq_p:Vn \l_tmpa_str {a}}
3594       {\str_if_eq_p:Vn \l_tmpa_str {B}}
3595     }{
3596       \msg_error:nnxx{stex}{error/doubleargument}
3597       {\int_use:N \l_tmpa_int}
3598       {\STEXInternalCurrentSymbolStr}
3599     }
3600   }
3601   \exp_args:NNx \prop_put:Nnn \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {\ign
3602   \bool_set_true:N \l_stex_allow_semantic_bool
3603   \IfBooleanTF#1{
3604     \stex_annotate_invisible:n { %TODO
3605       \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
3606     }
3607   }{ %TODO
3608     \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
3609   }
3610   \bool_set_false:N \l_stex_allow_semantic_bool
3611 }
3612
3613
3614 \cs_new_protected:Nn \_stex_term_arg:nn {
3615   \bool_set_true:N \l_stex_allow_semantic_bool
3616   \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3617   \bool_set_false:N \l_stex_allow_semantic_bool
3618 }
3619

```

```

3620 \cs_new_protected:Npn \STEXInternalTermMathArgiii #1#2#3 {
3621   \exp_args:Nnx \use:nn
3622     { \int_set:Nn \l__stex_terms_downprec { #2 }
3623       \stex_term_arg:nn { #1 }{ #3 }
3624     }
3625     { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3626   }

```

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

`\STEXInternalTermMathAssocArgiii`

```

3627 \cs_new_protected:Npn \STEXInternalTermMathAssocArgiii #1#2#3#4 {
3628   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3629   \tl_set:Nn \l_tmpb_tl {\STEXInternalTermMathArgiii{#1}{#2}}
3630   \tl_if_empty:nTF { #3 }{
3631     \STEXInternalTermMathArgiii{#1}{#2}{}
3632   }{
3633     \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 } }{
3634       \expandafter\if\expandafter\relax\noexpand#3
3635         \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_maybe_sequence:Nn#3{#1}}
3636       \else
3637         \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_simple:nn{#1}{#3}}
3638       \fi
3639       \l_tmpa_tl
3640     }{
3641       \__stex_terms_math_assoc_arg_simple:nn{#1}{#3}
3642     }
3643   }
3644 }
3645
3646 \cs_new_protected:Nn \__stex_terms_math_assoc_arg_maybe_sequence:Nn {
3647   \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
3648   \str_if_empty:NTF \l_tmpa_str {
3649     \exp_args:Nx \cs_if_eq:NNTF {
3650       \tl_head:N #1
3651     } \stex_invoke_sequence:n {
3652       \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
3653       \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
3654       \tl_set:Nx \l_tmpa_tl {\prop_item:cn {stex_varseq_\l_tmpa_str _prop}{notation}}
3655       \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
3656       \tl_set:Nx \l_tmpa_tl {\exp_not:N \exp_not:n{
3657         \exp_not:n{\exp_args:Nnx \use:nn} {
3658           \exp_not:n {
3659             \def\comp{\_varcomp}
3660             \str_set:Nn \STEXInternalCurrentSymbolStr
3661               { \varseq: // \l_tmpa_str }
3662             \exp_not:n{ ##1 }
3663           }{
3664             \exp_not:n {
3665               \stex_reset:N \comp
3666               \stex_reset:N \STEXInternalCurrentSymbolStr
3667             }
3668           }
3669         }}}

```

```

3670 \exp_args:Nno \use:nn {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
3671 \seq_reverse:N \l_tmpa_seq
3672 \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
3673 \seq_map_inline:Nn \l_tmpa_seq {
3674   \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3675     \exp_args:Nno
3676     \l_tmpa_cs { ##1 } \l_tmpa_tl
3677   }
3678 }
3679 \tl_set:Nx \l_tmpa_tl {
3680   \stex_term_omv:nn {varseq://\l_tmpa_str}{
3681     \exp_args:No \exp_not:n \l_tmpa_tl
3682   }
3683 }
3684 \exp_args:No\l_tmpb_tl\l_tmpa_tl
3685 }{
3686   \stex_terms_math_assoc_arg_simple:nn{#2} { #1 }
3687 }
3688 } {
3689   \stex_terms_math_assoc_arg_simple:nn{#2} { #1 }
3690 }
3691 }
3692 }
3693
3694 \cs_new_protected:Nn \stex_terms_math_assoc_arg_simple:nn {
3695   \clist_set:Nn \l_tmpa_clist{ #2 }
3696   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3697     \tl_set:Nn \l_tmpa_tl { \stex_term_arg:nn{A#1}{ #2 } }
3698   }{
3699     \clist_reverse:N \l_tmpa_clist
3700     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3701     \tl_set:Nx \l_tmpa_tl { \stex_term_arg:nn{A#1}{
3702       \exp_args:No \exp_not:n \l_tmpa_tl
3703     }}
3704     \clist_map_inline:Nn \l_tmpa_clist {
3705       \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3706         \exp_args:Nno
3707         \l_tmpa_cs { \stex_term_arg:nn{A#1}{##1} } \l_tmpa_tl
3708       }
3709     }
3710   }
3711   \exp_args:No\l_tmpb_tl\l_tmpa_tl
3712 }

```

(End definition for `\STEXInternalTermMathAssocArgiiii`. This function is documented on page 81.)

## 30.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
3713 \tl_const:Nx \infprec {\int_use:N \c_max_int}
3714 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}

```

```

3715 \int_new:N \l__stex_terms_downprec
3716 \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 81.)

Bracketing:

```

\l__stex_terms_left_bracket_str
\l__stex_terms_right_bracket_str

```

```

3717 \tl_set:Nn \l__stex_terms_left_bracket_str (
3718 \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

```

3719 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
3720   \bool_if:NTF \l__stex_terms_brackets_done_bool {
3721     \bool_set_false:N \l__stex_terms_brackets_done_bool
3722     #2
3723   } {
3724     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
3725       \bool_if:NTF \l__stex_inarray_bool { #2 } {
3726         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
3727         \dobrackets { #2 }
3728       }
3729     }{ #2 }
3730   }
3731 }

```

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

**\dobrackets**

```

3732 \bool_new:N \l__stex_terms_brackets_done_bool
3733 %\RequirePackage{scalerel}
3734 \cs_new_protected:Npn \dobrackets #1 {
3735   %\ThisStyle{\if D\m@switch
3736   %   \exp_args:Nnx \use:nn
3737   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
3738   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
3739   % \else
3740   \exp_args:Nnx \use:nn
3741   {
3742     \bool_set_true:N \l__stex_terms_brackets_done_bool
3743     \int_set:Nn \l__stex_terms_downprec \infpref
3744     \l__stex_terms_left_bracket_str
3745     #1
3746   }
3747   {
3748     \bool_set_false:N \l__stex_terms_brackets_done_bool
3749     \l__stex_terms_right_bracket_str
3750     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3751   }
3752   %\fi}
3753 }

```

(End definition for `\dobrackets`. This function is documented on page 81.)

**\withbrackets**

```
3754 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
3755   \exp_args:Nnx \use:nn
3756   {
3757     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
3758     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
3759     #3
3760   }
3761   {
3762     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
3763     {\l__stex_terms_left_bracket_str}
3764     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
3765     {\l__stex_terms_right_bracket_str}
3766   }
3767 }
```

(End definition for \withbrackets. This function is documented on page 81.)

**\STEXinvisible**

```
3768 \cs_new_protected:Npn \STEXinvisible #1 {
3769   \stex_annotate_invisible:n { #1 }
3770 }
```

(End definition for \STEXinvisible. This function is documented on page 81.)

OMDoc terms:

**\STEXInternalTermMathOMSiiii**

```
3771 \cs_new_protected:Nn \_stex_term_oms:nnn {
3772   \stex_annotate:nnn{ OMID }{ #2 }{
3773     #3
3774   }
3775 }
3776
3777 \cs_new_protected:Npn \STEXInternalTermMathOMSiiii #1#2#3#4 {
3778   \_stex_terms_maybe_brackets:nn { #3 }{
3779     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3780   }
3781 }
```

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

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

```
3782 \cs_new_protected:Nn \_stex_term_omv:nn {
3783   \stex_annotate:nnn{ OMV }{ #1 }{
3784     #2
3785   }
3786 }
```

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

**\STEXInternalTermMathOMAiiai**

```
3787 \cs_new_protected:Nn \_stex_term_oma:nnn {
3788   \stex_annotate:nnn{ OMA }{ #2 }{
3789     #3
3790   }
```

```

3791 }
3792
3793 \cs_new_protected:Npn \STEXInternalTermMathOMAiiai #1#2#3#4 {
3794   \__stex_terms_maybe_brackets:nn { #3 }{
3795     \stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3796   }
3797 }

```

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

\STEXInternalTermMathOMBiaiai

```

3798 \cs_new_protected:Nn \stex_term_ombind:nnn {
3799   \stex_annotate:nnn{ OMBIND }{ #2 }{
3800     #3
3801   }
3802 }
3803
3804 \cs_new_protected:Npn \STEXInternalTermMathOMBiaiai #1#2#3#4 {
3805   \__stex_terms_maybe_brackets:nn { #3 }{
3806     \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3807   }
3808 }

```

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

\symref  
\symname

```

3809 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3810
3811 \keys_define:nn { stex / symname } {
3812   pre      .tl_set_x:N      = \l__stex_terms_pre_tl ,
3813   post     .tl_set_x:N      = \l__stex_terms_post_tl ,
3814   root     .tl_set_x:N      = \l__stex_terms_root_tl
3815 }
3816
3817 \cs_new_protected:Nn \stex_symname_args:n {
3818   \tl_clear:N \l__stex_terms_post_tl
3819   \tl_clear:N \l__stex_terms_pre_tl
3820   \tl_clear:N \l__stex_terms_root_str
3821   \keys_set:nn { stex / symname } { #1 }
3822 }
3823
3824 \NewDocumentCommand \symref { m m }{
3825   \let\compemph_uri_prev:\compemph@uri
3826   \let\compemph@uri\symrefemph@uri
3827   \STEXsymbol{#1}!\{ #2 }
3828   \let\compemph@uri\compemph_uri_prev:
3829 }
3830
3831 \NewDocumentCommand \synonym { O{} m m }{
3832   \stex_symname_args:n { #1 }
3833   \let\compemph_uri_prev:\compemph@uri
3834   \let\compemph@uri\symrefemph@uri
3835   % TODO
3836   \STEXsymbol{#2}!\{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
3837   \let\compemph@uri\compemph_uri_prev:

```



```

3838 }
3839
3840 \NewDocumentCommand \symname { 0{} m }{
3841   \stex_symname_args:n { #1 }
3842   \stex_get_symbol:n { #2 }
3843   \str_set:Nx \l_tmpa_str {
3844     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3845   }
3846   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3847
3848   \let\compemph_uri_prev:\compemph@uri
3849   \let\compemph@uri\symrefemph@uri
3850   \exp_args:NNx \use:nn
3851   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmmode*\fi{
3852     \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
3853   } }
3854   \let\compemph@uri\compemph_uri_prev:
3855 }
3856
3857 \NewDocumentCommand \Symname { 0{} m }{
3858   \stex_symname_args:n { #1 }
3859   \stex_get_symbol:n { #2 }
3860   \str_set:Nx \l_tmpa_str {
3861     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3862   }
3863   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3864   \let\compemph_uri_prev:\compemph@uri
3865   \let\compemph@uri\symrefemph@uri
3866   \exp_args:NNx \use:nn
3867   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmmode*\fi{
3868     \exp_after:wN \stex_capitalize:n \l_tmpa_str
3869     \l__stex_terms_post_tl
3870   } }
3871   \let\compemph@uri\compemph_uri_prev:
3872 }

```

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

### 30.3 Notation Components

```

3873 <@@=stex_notationcomps>

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

3874 \cs_new_protected:Npn \_comp #1 {
3875   \str_if_empty:NF \STEXInternalCurrentSymbolStr {
3876     \stex_html_backend:TF {
3877       \stex_annotate:nnn { comp }{ \STEXInternalCurrentSymbolStr }{ #1 }
3878     }{
3879       \exp_args:Nnx \compemph@uri { #1 } { \STEXInternalCurrentSymbolStr }
3880     }
3881   }
3882 }
3883
3884 \cs_new_protected:Npn \_varcomp #1 {

```

```

3885 \str_if_empty:NF \STEXInternalCurrentSymbolStr {
3886   \stex_html_backend:TF {
3887     \stex_annotate:nnn { varcomp }{ \STEXInternalCurrentSymbolStr }{ #1 }
3888   }{
3889     \exp_args:Nnx \varemp@uri { #1 } { \STEXInternalCurrentSymbolStr }
3890   }
3891 }
3892 }
3893
3894 \def\comp{\_comp}
3895
3896 \cs_new_protected:Npn \compemph@uri #1 #2 {
3897   \compemph{ #1 }
3898 }
3899
3900
3901 \cs_new_protected:Npn \compemph #1 {
3902   #1
3903 }
3904
3905 \cs_new_protected:Npn \defemph@uri #1 #2 {
3906   \defemph{#1}
3907 }
3908
3909 \cs_new_protected:Npn \defemph #1 {
3910   \textbf{#1}
3911 }
3912
3913 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3914   \symrefemph{#1}
3915 }
3916
3917 \cs_new_protected:Npn \symrefemph #1 {
3918   \emph{#1}
3919 }
3920
3921 \cs_new_protected:Npn \varemp@uri #1 #2 {
3922   \varemp{#1}
3923 }
3924
3925 \cs_new_protected:Npn \varemp #1 {
3926   #1
3927 }

```

(End definition for `\comp` and others. These functions are documented on page 81.)

## **\ellipses**

```

3928 \NewDocumentCommand \ellipses {} { \ldots }

```

(End definition for `\ellipses`. This function is documented on page 81.)

```

\parray
\prmatrix
\parrayline
\parraylineh
\parraycell
3929 \bool_new:N \l_stex_inparray_bool
3930 \bool_set_false:N \l_stex_inparray_bool
3931 \NewDocumentCommand \parray { m m } {

```

```

3932 \begingroup
3933 \bool_set_true:N \l_stex_inarray_bool
3934 \begin{array}{#1}
3935   #2
3936 \end{array}
3937 \endgroup
3938 }
3939
3940 \NewDocumentCommand \prmatrix { m } {
3941   \begingroup
3942   \bool_set_true:N \l_stex_inarray_bool
3943   \begin{matrix}
3944     #1
3945   \end{matrix}
3946 \endgroup
3947 }
3948
3949 \def \maybepline {
3950   \bool_if:NT \l_stex_inarray_bool {\hline}
3951 }
3952
3953 \def \parrayline #1 #2 {
3954   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3955 }
3956
3957 \def \pmrow #1 { \parrayline{}{ #1 } }
3958
3959 \def \parraylineh #1 #2 {
3960   #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3961 }
3962
3963 \def \parraycell #1 {
3964   #1 \bool_if:NT \l_stex_inarray_bool {&}
3965 }

```

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

## 30.4 Variables

```

3966 <@@=stex_variables>

```

\stex\_invoke\_variable:n Invokes a variable

```

3967 \cs_new_protected:Nn \stex_invoke_variable:n {
3968   \if_mode_math:
3969     \exp_after:wN \__stex_variables_invoke_math:n
3970   \else:
3971     \exp_after:wN \__stex_variables_invoke_text:n
3972   \fi: {#1}
3973 }
3974
3975 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
3976   \peek_charcode_remove:NTF ! {
3977     \__stex_variables_invoke_op_custom:nn {#1}
3978   }{

```

```

3979     \__stex_variables_invoke_custom:nn {#1}
3980   }
3981 }
3982
3983
3984 \cs_new_protected:Nn \__stex_variables_invoke_math:n {
3985   \peek_charcode_remove:NTF ! {
3986     \peek_charcode_remove:NTF ! {
3987       \peek_charcode:NTF [ {
3988         % TODO throw error
3989       }{
3990         \__stex_variables_invoke_op_custom:nn
3991       }
3992     }{
3993       \__stex_variables_invoke_op:n { #1 }
3994     }
3995   }{
3996     \peek_charcode_remove:NTF * {
3997       \__stex_variables_invoke_custom:nn { #1 }
3998     }{
3999       \__stex_variables_invoke_math_ii:n { #1 }
4000     }
4001   }
4002 }
4003
4004 \cs_new_protected:Nn \__stex_variables_invoke_op_custom:nn {
4005   \exp_args:Nnx \use:nn {
4006     \def\comp{\_varcomp}
4007     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4008     \bool_set_false:N \l_stex_allow_semantic_bool
4009     \stex_term_omv:nn {var://#1}{
4010       \comp{ #2 }
4011     }
4012   }{
4013     \_stex_reset:N \comp
4014     \_stex_reset:N \STEXInternalCurrentSymbolStr
4015     \bool_set_true:N \l_stex_allow_semantic_bool
4016   }
4017 }
4018
4019 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
4020   \cs_if_exist:cTF {
4021     stex_var_op_notation_ #1 _cs
4022   }{
4023     \exp_args:Nnx \use:nn {
4024       \def\comp{\_varcomp}
4025       \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4026       \_stex_term_omv:nn { var://#1 }{
4027         \use:c{stex_var_op_notation_ #1 _cs }
4028       }
4029     }{
4030       \_stex_reset:N \comp
4031       \_stex_reset:N \STEXInternalCurrentSymbolStr
4032     }

```

```

4033 }{
4034   \int_compare:nNnTF {\prop_item:cn {l_stex_variable_#1_prop}{arity}} = 0{
4035     \__stex_variables_invoke_math_ii:n {#1}
4036   }{
4037     \msg_error:nnxx{stex}{error/noop}{variable~#1}{}
4038   }
4039 }
4040 }
4041
4042 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n #1 {
4043   \cs_if_exist:cTF {
4044     stex_var_notation_#1_cs
4045   }{
4046     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4047       \_stex_reset:N \comp
4048       \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
4049       \_stex_reset:N \STEXInternalCurrentSymbolStr
4050       \bool_set_true:N \l_stex_allow_semantic_bool
4051     }
4052     \def\comp{\_varcomp}
4053     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4054     \bool_set_false:N \l_stex_allow_semantic_bool
4055     \use:c{stex_var_notation_#1_cs}
4056   }{
4057     \msg_error:nnxx{stex}{error/nonotation}{variable~#1}{s}
4058   }
4059 }
4060
4061 \cs_new_protected:Nn \__stex_variables_invoke_custom:nn {
4062   \exp_args:Nnx \use:nn {
4063     \def\comp{\_varcomp}
4064     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4065     \prop_clear:N \l__stex_terms_custom_args_prop
4066     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
4067     \prop_get:cnN {
4068       l_stex_variable_#1 _prop
4069     }{ args } \l_tmpa_str
4070     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
4071     \tl_set:Nn \arg { \__stex_terms_arg: }
4072     \str_if_empty:NTF \l_tmpa_str {
4073       \_stex_term_omv:nn {var://#1}{\ignorespaces#2}
4074     }{
4075       \str_if_in:NnTF \l_tmpa_str b {
4076         \_stex_term_ombind:nnn {var://#1}{\ignorespaces#2}
4077       }{
4078         \str_if_in:NnTF \l_tmpa_str B {
4079           \_stex_term_ombind:nnn {var://#1}{\ignorespaces#2}
4080         }{
4081           \_stex_term_oma:nnn {var://#1}{\ignorespaces#2}
4082         }
4083       }
4084     }
4085     % TODO check that all arguments exist
4086   }{

```

```

4087 \_stex_reset:N \STEXInternalCurrentSymbolStr
4088 \_stex_reset:N \arg
4089 \_stex_reset:N \comp
4090 \_stex_reset:N \l__stex_terms_custom_args_prop
4091 %\bool_set_true:N \l_stex_allow_semantic_bool
4092 }
4093 }

```

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

## 30.5 Sequences

```

4094 <@@=stex_sequences>
4095
4096 \cs_new_protected:Nn \stex_invoke_sequence:n {
4097   \peek_charcode_remove:NTF ! {
4098     \_stex_term_omv:nn {varseq://#1}{
4099       \exp_args:Nnx \use:nn {
4100         \def\comp{\_varcomp}
4101         \str_set:Nn \STEXInternalCurrentSymbolStr {varseq://#1}
4102         \prop_item:cn{stex_varseq_#1_prop}{notation}
4103       }{
4104         \_stex_reset:N \comp
4105         \_stex_reset:N \STEXInternalCurrentSymbolStr
4106       }
4107     }
4108   }{
4109     \bool_set_false:N \l_stex_allow_semantic_bool
4110     \def\comp{\_varcomp}
4111     \str_set:Nn \STEXInternalCurrentSymbolStr {varseq://#1}
4112     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4113       \_stex_reset:N \comp
4114       \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
4115       \_stex_reset:N \STEXInternalCurrentSymbolStr
4116       \bool_set_true:N \l_stex_allow_semantic_bool
4117     }
4118     \use:c { stex_varseq_#1_cs }
4119   }
4120 }
4121 </package>

```

## Chapter 31

# STEX -Structural Features Implementation

```
4122 ⟨*package⟩
4123
4124 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4125
      Warnings and error messages
4126 \msg_new:nnn{stex}{error/copymodule/notallowed}{
4127   Symbol~#1~can~not~be~assigned~in~copymodule~#2
4128 }
4129 \msg_new:nnn{stex}{error/interpretmodule/nodfiniens}{
4130   Symbol~#1~not~assigned~in~interpretmodule~#2
4131 }
4132
4133 \msg_new:nnn{stex}{error/unknownstructure}{
4134   No~structure~#1~found!
4135 }
4136
4137 \msg_new:nnn{stex}{error/unknownfield}{
4138   No~field~#1~in~instance~#2~found!\#3
4139 }
4140
4141 \msg_new:nnn{stex}{error/keyval}{
4142   Invalid~key=value~pair:#1
4143 }
4144 \msg_new:nnn{stex}{error/instantiate/missing}{
4145   Assignments~missing~in~instantiate:~#1
4146 }
4147 \msg_new:nnn{stex}{error/incompatible}{
4148   Incompatible~signature:~#1~(#2)~and~#3~(#4)
4149 }
4150
```

## 31.1 Imports with modification

```

4151 <@@=stex_copymodule>
4152 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
4153   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4154     \tl_set:Nn \l_tmpa_tl { #1 }
4155     \__stex_copymodule_get_symbol_from_cs:
4156   }{
4157     % argument is a string
4158     % is it a command name?
4159     \cs_if_exist:cTF { #1 }{
4160       \cs_set_eq:Nc \l_tmpa_tl { #1 }
4161       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4162       \str_if_empty:NTF \l_tmpa_str {
4163         \exp_args:Nx \cs_if_eq:NNTF {
4164           \tl_head:N \l_tmpa_tl
4165         } \stex_invoke_symbol:n {
4166           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
4167         }{
4168           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4169         }
4170       } {
4171         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4172       }
4173     }{
4174       % argument is not a command name
4175       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4176       % \l_stex_all_symbols_seq
4177     }
4178   }
4179 }
4180
4181 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
4182   \str_set:Nn \l_tmpa_str { #1 }
4183   \bool_set_false:N \l_tmpa_bool
4184   \bool_if:NF \l_tmpa_bool {
4185     \tl_set:Nn \l_tmpa_tl {
4186       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
4187     }
4188     \str_set:Nn \l_tmpa_str { #1 }
4189     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4190     \seq_map_inline:Nn #2 {
4191       \str_set:Nn \l_tmpb_str { ##1 }
4192       \str_if_eq:eeT { \l_tmpa_str } {
4193         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4194       } {
4195         \seq_map_break:n {
4196           \tl_set:Nn \l_tmpa_tl {
4197             \str_set:Nn \l_stex_get_symbol_uri_str {
4198               ##1
4199             }
4200           }
4201         }
4202       }

```



```

4203     }
4204     \l_tmpa_tl
4205   }
4206 }
4207
4208 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
4209   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4210     { \tl_tail:N \l_tmpa_tl }
4211   \tl_if_single:NTF \l_tmpa_tl {
4212     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
4213       \exp_after:wN \str_set:Nn \exp_after:wN
4214         \l_stex_get_symbol_uri_str \l_tmpa_tl
4215       \__stex_copymodule_get_symbol_check:n { #1 }
4216     }{
4217       % TODO
4218       % tail is not a single group
4219     }
4220   }{
4221     % TODO
4222     % tail is not a single group
4223   }
4224 }
4225
4226 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
4227   \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
4228     \msg_error:nxxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
4229       :~\seq_use:Nn #1 {,~}
4230     }
4231   }
4232 }
4233
4234 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
4235   % import module
4236   \stex_import_module_uri:nn { #1 } { #2 }
4237   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
4238   \stex_import_require_module:nnnn
4239     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4240     { \l_stex_import_path_str } { \l_stex_import_name_str }
4241
4242   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
4243   \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
4244
4245   % fields
4246   \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
4247   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4248     \seq_map_inline:cn {c_stex_module_##1_constants}{
4249       \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
4250         ##1 ? ####1
4251       }
4252     }
4253   }
4254
4255   % setup prop
4256   \seq_clear:N \l_tmpa_seq

```

```

4257 \exp_args:Nn \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
4258   name      = \l_stex_current_copymodule_name_str ,
4259   module    = \l_stex_current_module_str ,
4260   from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
4261   includes  = \l_tmpa_seq %,
4262 % fields    = \l_tmpa_seq
4263 }
4264 \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
4265   as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
4266 \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_seq {,
4267 \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
4268
4269 \stex_if_do_html:T {
4270   \begin{stex_annotate_env} {#4} {
4271     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4272   }
4273   \stex_annotate_invisible:nnn{domain}{\l_stex_import_ns_str ?\l_stex_import_name_str}{}
4274 }
4275 }
4276
4277 \cs_new_protected:Nn \stex_copymodule_end:n {
4278   % apply to every field
4279   \def \l_tmpa_cs ##1 ##2 {#1}
4280
4281   \tl_clear:N \__stex_copymodule_module_tl
4282   \tl_clear:N \__stex_copymodule_exec_tl
4283
4284   %\prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
4285   \seq_clear:N \__stex_copymodule_fields_seq
4286
4287   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4288     \seq_map_inline:cn {c_stex_module_##1_constants}{
4289
4290       \tl_clear:N \__stex_copymodule_curr_symbol_tl % <- wrap in current symbol html
4291       \l_tmpa_cs{##1}{####1}
4292
4293       \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
4294         \str_set_eq:Nc \__stex_copymodule_curr_name_str {l__stex_copymodule_copymodule_##1?####1_name_str}
4295         \stex_if_do_html:T {
4296           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4297             \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?####1_name_str}}
4298           }
4299         }
4300       }{
4301         \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str /
4302       }
4303
4304       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
4305       \prop_put:Nnx \l_tmpa_prop { name } \__stex_copymodule_curr_name_str
4306       \prop_put:Nnx \l_tmpa_prop { module } \l_stex_current_module_str
4307
4308       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4309         \stex_if_do_html:T {
4310           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {

```

```

4311         $\stex_annotate_invisible:nnn{definiens}{\exp_after:wN \exp_not:N\csname l__st
4312     }
4313 }
4314 \prop_put:Nnn \l_tmpa_prop { defined } { true }
4315 }
4316
4317 \stex_add_constant_to_current_module:n \__stex_copymodule_curr_name_str
4318 \tl_put_right:Nx \__stex_copymodule_module_tl {
4319     \seq_clear:c {l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_n
4320     \prop_set_from_keyval:cn {
4321         l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_name_str _prop
4322     }{
4323         \prop_to_keyval:N \l_tmpa_prop
4324     }
4325 }
4326
4327 \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
4328     \stex_if_do_html:T {
4329         \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4330             \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1
4331         }
4332     }
4333     \tl_put_right:Nx \__stex_copymodule_module_tl {
4334         \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
4335             \stex_invoke_symbol:n {
4336                 \l_stex_current_module_str ? \__stex_copymodule_curr_name_str
4337             }
4338         }
4339     }
4340 }
4341
4342 \seq_put_right:Nx \__stex_copymodule_fields_seq {\l_stex_current_module_str ? \__stex_
4343
4344 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4345     \stex_copy_notations:nn {\l_stex_current_module_str ? \__stex_copymodule_curr_name_s
4346 }
4347
4348 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4349     \stex_if_do_html:TF{
4350         \stex_annotate_invisible:nnn{assignment} {##1?####1} { \exp_after:wN \exp_not:n \e
4351     }{
4352         \exp_after:wN \exp_not:n \exp_after:wN {\__stex_copymodule_curr_symbol_tl}
4353     }
4354 }
4355 }
4356 }
4357
4358
4359 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \__stex_copymodule_fields_seq
4360 \tl_put_left:Nx \__stex_copymodule_module_tl {
4361     \prop_set_from_keyval:cn {
4362         l_stex_copymodule_ \l_stex_current_module_str? \l_stex_current_copymodule_name_str _pro
4363     }{
4364         \prop_to_keyval:N \l_stex_current_copymodule_prop

```

```

4365     }
4366 }
4367
4368 \seq_gput_right:cx{c_stex_module_\l_stex_current_module_str _copymodules}{
4369   \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4370 }
4371
4372 \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4373 \stex_debug:nn{copymodule}{result:\meaning \__stex_copymodule_module_tl}
4374 \stex_debug:nn{copymodule}{output:\meaning \__stex_copymodule_exec_tl}
4375
4376 \__stex_copymodule_exec_tl
4377 \stex_if_do_html:T {
4378   \end{stex_annotate_env}
4379 }
4380 }
4381
4382 \NewDocumentEnvironment {copymodule} { 0{} m m}{
4383   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ copymodule }
4384   \stex_deactivate_macro:Nn \symdecl {module~environments}
4385   \stex_deactivate_macro:Nn \symdef {module~environments}
4386   \stex_deactivate_macro:Nn \notation {module~environments}
4387   \stex_reactivate_macro:N \assign
4388   \stex_reactivate_macro:N \renamedekl
4389   \stex_reactivate_macro:N \donotcopy
4390   \stex_smsmode_do:
4391 }{
4392   \stex_copymodule_end:n {}
4393 }
4394
4395 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
4396   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ interpretmodule }
4397   \stex_deactivate_macro:Nn \symdecl {module~environments}
4398   \stex_deactivate_macro:Nn \symdef {module~environments}
4399   \stex_deactivate_macro:Nn \notation {module~environments}
4400   \stex_reactivate_macro:N \assign
4401   \stex_reactivate_macro:N \renamedekl
4402   \stex_reactivate_macro:N \donotcopy
4403   \stex_smsmode_do:
4404 }{
4405   \stex_copymodule_end:n {
4406     \tl_if_exist:cF {
4407       l__stex_copymodule_copymodule_##1?##2_def_tl
4408     }{
4409       \str_if_eq:eeF {
4410         \prop_item:cn{
4411           l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
4412         }{ true }{
4413           \msg_error:nxxx{stex}{error/interpretmodule/nodéfiniens}{
4414             ##1?##2
4415           }{\l_stex_current_copymodule_name_str}
4416         }
4417       }
4418     }

```

```

4419 }
4420
4421 \iffalse \begin{stex_annotate_env} \fi
4422 \NewDocumentEnvironment {realization} { 0 } { m } {
4423   \stex_copymodule_start:nnnn { #1 } { #2 } { #2 } { realize }
4424   \stex_deactivate_macro:Nn \symdecl {module~environments}
4425   \stex_deactivate_macro:Nn \symdef {module~environments}
4426   \stex_deactivate_macro:Nn \notation {module~environments}
4427   \stex_reactivate_macro:N \donotcopy
4428   \stex_reactivate_macro:N \assign
4429   \stex_smsmode_do:
4430 } {
4431   \stex_import_module_uri:nn { #1 } { #2 }
4432   \tl_clear:N \__stex_copymodule_exec_tl
4433   \tl_set:Nx \__stex_copymodule_module_tl {
4434     \stex_import_require_module:nnnn
4435     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4436     { \l_stex_import_path_str } { \l_stex_import_name_str }
4437   }
4438
4439   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4440     \seq_map_inline:cn {c_stex_module_##1_constants}{
4441       \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str / #1 }
4442       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?###1_def_tl}{
4443         \stex_if_do_html:T {
4444           \tl_put_right:Nx \__stex_copymodule_exec_tl {
4445             \stex_annotate_invisible:nnn{assignment} {##1?###1} {
4446               $\stex_annotate_invisible:nnn{definien}{}{\exp_after:wN \exp_not:N\csname l__
4447             }
4448           }
4449         }
4450         \tl_put_right:Nx \__stex_copymodule_module_tl {
4451           \prop_put:cnn {l_stex_symdecl_##1?###1_prop}{ defined }{ true }
4452         }
4453       }
4454     }
4455
4456     \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4457
4458     \__stex_copymodule_exec_tl
4459     \stex_if_do_html:T {\end{stex_annotate_env}}
4460   }
4461
4462   \NewDocumentCommand \donotcopy { m } {
4463     \str_clear:N \l_stex_import_name_str
4464     \str_set:Nn \l_tmpa_str { #1 }
4465     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4466     \seq_map_inline:Nn \l_stex_all_modules_seq {
4467       \str_set:Nn \l_tmpb_str { ##1 }
4468       \str_if_eq:eeT { \l_tmpa_str } {
4469         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4470       } {
4471         \seq_map_break:n {
4472           \stex_if_do_html:T {

```

```

4473         \stex_if_smsmode:F {
4474             \stex_annotate_invisible:nnn{donotcopy}{##1}{
4475                 \stex_annotate:nnn{domain}{##1}{}}
4476         }
4477     }
4478 }
4479 \str_set_eq:NN \l_stex_import_name_str \l_tmpb_str
4480 }
4481 }
4482 \seq_map_inline:cn {c_stex_module_###1_copymodules}{
4483     \str_set:Nn \l_tmpb_str { #####1 }
4484     \str_if_eq:eeT { \l_tmpa_str } {
4485         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4486     } {
4487         \seq_map_break:n {\seq_map_break:n {
4488             \stex_if_do_html:T {
4489                 \stex_if_smsmode:F {
4490                     \stex_annotate_invisible:nnn{donotcopy}{#####1}{
4491                         \stex_annotate:nnn{domain}{
4492                             \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4493                         }{}
4494                     }
4495                 }
4496             }
4497             \str_set:Nx \l_stex_import_name_str {
4498                 \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4499             }
4500         }}
4501     }
4502 }
4503 }
4504 \str_if_empty:NTF \l_stex_import_name_str {
4505     % TODO throw error
4506 }{
4507     \stex_collect_imports:n {\l_stex_import_name_str }
4508     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4509         \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
4510         \seq_map_inline:cn {c_stex_module_###1_constants}{
4511             \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? #####1 }
4512             \bool_lazy_any:nT {
4513                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_###1?#####1_name_str}}
4514                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_###1?#####1_macroname_str}}
4515                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_###1?#####1_def_tl}}
4516             }{
4517                 % TODO throw error
4518             }
4519         }
4520     }
4521     \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
4522     \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_name_str }
4523     \prop_put:Nno \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
4524 }
4525 \stex_smsmode_do:
4526 }

```

```

4527
4528 \NewDocumentCommand \assign { m m }{
4529   \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
4530   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
4531   \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_def_tl}{#2}
4532   \stex_smsmode_do:
4533 }
4534
4535 \keys_define:nn { stex / renamedecl } {
4536   name          .str_set_x:N = \l_stex_renamedecl_name_str
4537 }
4538 \cs_new_protected:Nn \__stex_copymodule_renamedecl_args:n {
4539   \str_clear:N \l_stex_renamedecl_name_str
4540   \keys_set:nn { stex / renamedecl } { #1 }
4541 }
4542
4543 \NewDocumentCommand \renamedecl { O{} m m }{
4544   \__stex_copymodule_renamedecl_args:n { #1 }
4545   \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
4546   \stex_debug:nn{renamedecl}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
4547   \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_macroname_str}{#3}
4548   \str_if_empty:NTF \l_stex_renamedecl_name_str {
4549     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4550       \l_stex_get_symbol_uri_str
4551     } }
4552   } {
4553     \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_name_str}{\l_stex
4554       \stex_debug:nn{renamedecl}{@~\l_stex_current_module_str ? \l_stex_renamedecl_name_str}
4555       \prop_set_eq:cc {l_stex_symdecl_
4556         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4557         _prop
4558       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str_prop}
4559       \seq_set_eq:cc {l_stex_symdecl_
4560         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4561         _notations
4562       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str_notations}
4563       \prop_put:cnx {l_stex_symdecl_
4564         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4565         _prop
4566       }{ name }{ \l_stex_renamedecl_name_str }
4567       \prop_put:cnx {l_stex_symdecl_
4568         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4569         _prop
4570       }{ module }{ \l_stex_current_module_str }
4571       \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
4572         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4573       }
4574       \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4575         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4576       } }
4577     }
4578   \stex_smsmode_do:
4579 }
4580

```

```

4581 \stex_deactivate_macro:Nn \assign {copymodules}
4582 \stex_deactivate_macro:Nn \renamedekl {copymodules}
4583 \stex_deactivate_macro:Nn \donotcopy {copymodules}
4584
4585

```

## 31.2 The feature environment

structural@feature

```

4586 <@@=stex_features>
4587
4588 \NewDocumentEnvironment{structural_feature_module}{ m m m }{
4589   \stex_if_in_module:F {
4590     \msg_set:nnn{stex}{error/nomodule}{
4591       Structural~Feature~has~to~occur~in~a~module:\\
4592       Feature~#2~of~type~#1\\
4593       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
4594     }
4595     \msg_error:nn{stex}{error/nomodule}
4596   }
4597
4598   \str_set_eq:NN \l_stex_feature_parent_str \l_stex_current_module_str
4599
4600   \stex_module_setup:nn{meta=NONE}{#2 - #1}
4601
4602   \stex_if_do_html:T {
4603     \begin{stex_annotate_env}{ feature:#1 }{\l_stex_feature_parent_str ? #2 - #1}
4604     \stex_annotate_invisible:nnn{header}{\{ #3 }
4605   }
4606 }{
4607   \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
4608   \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
4609   \stex_debug:nn{features}{
4610     Feature: \l_stex_last_feature_str
4611   }
4612   \stex_if_do_html:T {
4613     \end{stex_annotate_env}
4614   }
4615 }

```

## 31.3 Structure

structure

```

4616 <@@=stex_structures>
4617 \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
4618   \prop_if_exist:cF {c_stex_module_ \l_stex_current_module_str _structures}{
4619     \prop_new:c {c_stex_module_ \l_stex_current_module_str _structures}
4620   }
4621   \prop_gput:cxn{c_stex_module_ \l_stex_current_module_str _structures}
4622   {#1}{#2}
4623 }
4624

```



```

4625 \keys_define:nn { stex / features / structure } {
4626   name          .str_set_x:N = \l__stex_structures_name_str ,
4627 }
4628
4629 \cs_new_protected:Nn \__stex_structures_structure_args:n {
4630   \str_clear:N \l__stex_structures_name_str
4631   \keys_set:nn { stex / features / structure } { #1 }
4632 }
4633
4634 \NewDocumentEnvironment{mathstructure}{m O{}}{
4635   \__stex_structures_structure_args:n { #2 }
4636   \str_if_empty:NT \l__stex_structures_name_str {
4637     \str_set:Nx \l__stex_structures_name_str { #1 }
4638   }
4639   \stex_suppress_html:n {
4640     \bool_set_true:N \l_stex_symdecl_make_macro_bool
4641     \exp_args:Nx \stex_symdecl_do:nn {
4642       name = \l__stex_structures_name_str ,
4643       def = {\STEXsymbol{module-type}}{
4644         \STEXInternalTermMathOMSiiii {
4645           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4646             { ns } ?
4647           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4648             { name } / \l__stex_structures_name_str - structure
4649         }{}{0}{}
4650       }}
4651     }{ #1 }
4652   }
4653   \exp_args:Nnnx
4654   \begin{structural_feature_module}{ structure }
4655     { \l__stex_structures_name_str }{}
4656   \stex_smsmode_do:
4657 }{
4658   \end{structural_feature_module}
4659   \_stex_reset_up_to_module:n \l_stex_last_feature_str
4660   \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
4661   \seq_clear:N \l_tmpa_seq
4662   \seq_map_inline:Nn \l_stex_collect_imports_seq {
4663     \seq_map_inline:cn{c_stex_module_##1_constants}{
4664       \seq_put_right:Nn \l_tmpa_seq { ##1 ? ####1 }
4665     }
4666   }
4667   \exp_args:Nnno
4668   \prop_gput:cnn {c_stex_module_ \l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
4669   \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
4670   \stex_add_structure_to_current_module:nn
4671     \l__stex_structures_name_str
4672     \l_stex_last_feature_str
4673
4674   \stex_execute_in_module:x {
4675     \tl_set:cn { #1 }{
4676       \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures
4677     }
4678   }

```

```

4679 }
4680
4681 \cs_new:Nn \stex_invoke_structure:nn {
4682   \stex_invoke_symbol:n { #1?#2 }
4683 }
4684
4685 \cs_new_protected:Nn \stex_get_structure:n {
4686   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4687     \tl_set:Nn \l_tmpa_tl { #1 }
4688     \__stex_structures_get_from_cs:
4689   }{
4690     \cs_if_exist:cTF { #1 }{
4691       \cs_set_eq:Nc \l_tmpa_cs { #1 }
4692       \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
4693       \str_if_empty:NTF \l_tmpa_str {
4694         \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs} \stex_invoke_structure:nn {
4695           \__stex_structures_get_from_cs:
4696         }{
4697           \__stex_structures_get_from_string:n { #1 }
4698         }
4699       }{
4700         \__stex_structures_get_from_string:n { #1 }
4701       }
4702     }{
4703       \__stex_structures_get_from_string:n { #1 }
4704     }
4705   }
4706 }
4707
4708 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
4709   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4710     { \tl_tail:N \l_tmpa_tl }
4711   \str_set:Nx \l_tmpa_str {
4712     \exp_after:wN \use_i:nn \l_tmpa_tl
4713   }
4714   \str_set:Nx \l_tmpb_str {
4715     \exp_after:wN \use_ii:nn \l_tmpa_tl
4716   }
4717   \str_set:Nx \l_stex_get_structure_str {
4718     \l_tmpa_str ? \l_tmpb_str
4719   }
4720   \str_set:Nx \l_stex_get_structure_module_str {
4721     \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
4722   }
4723 }
4724
4725 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
4726   \tl_set:Nn \l_tmpa_tl {
4727     \msg_error:nnn{stex}{error/unknownstructure}{#1}
4728   }
4729   \str_set:Nn \l_tmpa_str { #1 }
4730   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4731
4732   \seq_map_inline:Nn \l_stex_all_modules_seq {

```

```

4733 \prop_if_exist:cT {c_stex_module_##1_structures} {
4734 \prop_map_inline:cn {c_stex_module_##1_structures} {
4735 \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
4736 \prop_map_break:n{\seq_map_break:n{
4737 \tl_set:Nn \l_tmpa_tl {
4738 \str_set:Nn \l_stex_get_structure_str {##1?####1}
4739 \str_set:Nn \l_stex_get_structure_module_str {####2}
4740 }
4741 }}
4742 }
4743 }
4744 }
4745 }
4746 \l_tmpa_tl
4747 }

```

**\instantiate**

```

4748
4749 \keys_define:nn { stex / instantiate } {
4750 name .str_set_x:N = \l__stex_structures_name_str
4751 }
4752 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
4753 \str_clear:N \l__stex_structures_name_str
4754 \keys_set:nn { stex / instantiate } { #1 }
4755 }
4756
4757 \NewDocumentCommand \instantiate {m O{} m m O{}}{
4758 \beginingroup
4759 \stex_get_structure:n {#3}
4760 \__stex_structures_instantiate_args:n { #2 }
4761 \str_if_empty:NT \l__stex_structures_name_str {
4762 \str_set:Nn \l__stex_structures_name_str { #1 }
4763 }
4764 \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4765 \seq_clear:N \l__stex_structures_fields_seq
4766 \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4767 \seq_map_inline:Nn \l_stex_collect_imports_seq {
4768 \seq_map_inline:cn {c_stex_module_##1_constants}{
4769 \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4770 }
4771 }
4772
4773 \tl_if_empty:nF{#5}{
4774 \seq_set_split:Nnn \l_tmpa_seq , {#5}
4775 \prop_clear:N \l_tmpa_prop
4776 \seq_map_inline:Nn \l_tmpa_seq {
4777 \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4778 \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4779 \msg_error:nnn{stex}{error/keyval}{##1}
4780 }
4781 \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
4782 \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4783 \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
4784 \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}

```

```

4785     \exp_args:Nxx \str_if_eq:nnF
4786     {\prop_item:cn{l_stex_symdecl\_l__stex_structures_dom_str _prop}{args}}
4787     {\prop_item:cn{l_stex_symdecl\_l_stex_get_symbol_uri_str _prop}{args}}{
4788     \msg_error:nnxxxx{stex}{error/incompatible}
4789     {l__stex_structures_dom_str}
4790     {\prop_item:cn{l_stex_symdecl\_l__stex_structures_dom_str _prop}{args}}
4791     {\l_stex_get_symbol_uri_str}
4792     {\prop_item:cn{l_stex_symdecl\_l_stex_get_symbol_uri_str _prop}{args}}
4793   }
4794   \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
4795 }
4796 }
4797
4798 \seq_map_inline:Nn \l__stex_structures_fields_seq {
4799   \str_set:Nx \l_tmpa_str {field:\l__stex_structures_name_str . \prop_item:cn {l_stex_sy
4800   \stex_debug:nn{instantiate}{Field~\l_tmpa_str :~##1}
4801
4802   \stex_add_constant_to_current_module:n {\l_tmpa_str}
4803   \stex_execute_in_module:x {
4804     \prop_set_from_keyval:cn { l_stex_symdecl\_l_stex_current_module_str?\l_tmpa_str _p
4805     name   = \l_tmpa_str ,
4806     args   = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4807     arity  = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4808     assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4809   }
4810   \seq_clear:c {l_stex_symdecl\_l_stex_current_module_str?\l_tmpa_str _notations}
4811 }
4812
4813 \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4814   \stex_find_notation:nn{##1}{}
4815   \stex_execute_in_module:x {
4816     \seq_put_right:cn {l_stex_symdecl\_l_stex_current_module_str?\l_tmpa_str _notation
4817   }
4818
4819   \stex_copy_control_sequence_ii:ccN
4820   {stex_notation\_l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_notation_
4821   {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4822   \l_tmpa_tl
4823   \exp_args:No \stex_execute_in_module:n \l_tmpa_tl
4824
4825
4826   \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
4827     \tl_set_eq:Nc \l_tmpa_cs {stex_op_notation_##1\c_hash_str \l_stex_notation_variant
4828     \stex_execute_in_module:x {
4829       \tl_set:cn
4830       {stex_op_notation\_l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_not
4831       { \exp_args:No \exp_not:n \l_tmpa_cs}
4832     }
4833   }
4834
4835 }
4836
4837 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\l_stex_cur
4838 }

```

```

4839
4840 \stex_execute_in_module:x {
4841   \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
4842     domain = \l_stex_get_structure_module_str ,
4843     \prop_to_keyval:N \l_tmpa_prop
4844   }
4845   \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
4846 }
4847 \stex_debug:nn{instantiate}{
4848   Instance~\l_stex_current_module_str?\l__stex_structures_name_str \
4849   \prop_to_keyval:N \l_tmpa_prop
4850 }
4851 \exp_args:Nxx \stex_symdecl_do:nn {
4852   type={\STEXsymbol{module-type}}{
4853     \STEXInternalTermMathOMSiiii {
4854       \l_stex_get_structure_module_str
4855     }{}{0}{}
4856   }}
4857 }{\l__stex_structures_name_str}
4858 % {
4859   \str_set:Nx \l_stex_get_symbol_uri_str {\l_stex_current_module_str?\l__stex_structures
4860   \tl_set:Nn \l_stex_notation_after_do_tl {\l__stex_notation_final:}
4861   \stex_notation_do:nnnnn{}{}{}{}{\comp{#4}}
4862 % }
4863 %\exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}
4864 \endgroup
4865 \stex_smsmode_do:\ignorespacesandpars
4866 }
4867
4868 \cs_new_protected:Nn \stex_symbol_or_var:n {
4869   \cs_if_exist:cTF{#1}{
4870     \cs_set_eq:Nc \l_tmpa_tl { #1 }
4871     \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4872     \str_if_empty:NTF \l_tmpa_str {
4873       \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
4874       \stex_invoke_variable:n {
4875         \bool_set_true:N \l_stex_symbol_or_var_bool
4876         \bool_set_false:N \l_stex_instance_or_symbol_bool
4877         \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4878         \tl_set:Nx \l_tmpa_tl {\exp_after:wN \use:n \l_tmpa_tl}
4879         \str_set:Nx \l_stex_get_symbol_uri_str {
4880           \exp_after:wN \use:n \l_tmpa_tl
4881         }
4882       }{ % TODO \stex_invoke_varinstance:n
4883         \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl } \stex_invoke_varinstance:n {
4884           \bool_set_true:N \l_stex_symbol_or_var_bool
4885           \bool_set_true:N \l_stex_instance_or_symbol_bool
4886           \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4887           \tl_set:Nx \l_tmpa_tl {\exp_after:wN \use:n \l_tmpa_tl}
4888           \str_set:Nx \l_stex_get_symbol_uri_str {
4889             \exp_after:wN \use:n \l_tmpa_tl
4890           }
4891         }{
4892           \bool_set_false:N \l_stex_symbol_or_var_bool

```

```

4893         \stex_get_symbol:n{#1}
4894     }
4895 }
4896 }{
4897     \__stex_structures_symbolorvar_from_string:n{ #1 }
4898 }
4899 }{
4900     \__stex_structures_symbolorvar_from_string:n{ #1 }
4901 }
4902 }
4903
4904 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
4905     \prop_if_exist:cTF {l_stex_variable_#1 _prop}{
4906         \bool_set_true:N \l_stex_symbol_or_var_bool
4907         \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
4908     }{
4909         \bool_set_false:N \l_stex_symbol_or_var_bool
4910         \stex_get_symbol:n{#1}
4911     }
4912 }
4913
4914 \keys_define:nn { stex / varinstantiate } {
4915     name          .str_set_x:N = \l__stex_structures_name_str,
4916     bind          .choices:nn =
4917         {forall,exists}
4918         {\str_set:Nx \l__stex_structures_bind_str {\l_keys_choice_tl}}
4919 }
4920
4921 \cs_new_protected:Nn \__stex_structures_varinstantiate_args:n {
4922     \str_clear:N \l__stex_structures_name_str
4923     \str_clear:N \l__stex_structures_bind_str
4924     \keys_set:nn { stex / varinstantiate } { #1 }
4925 }
4926
4927 \NewDocumentCommand \varinstantiate {m O{} m m O{}}{
4928     \begingroup
4929         \stex_get_structure:n {#3}
4930         \__stex_structures_varinstantiate_args:n { #2 }
4931         \str_if_empty:NT \l__stex_structures_name_str {
4932             \str_set:Nn \l__stex_structures_name_str { #1 }
4933         }
4934         \stex_if_do_html:TF{
4935             \stex_annotate:nnn{varinstance}{\l__stex_structures_name_str}
4936         }{\use:n}
4937     {
4938         \stex_if_do_html:T{
4939             \stex_annotate_invisible:nnn{domain}{\l_stex_get_structure_module_str}{}
4940         }
4941         \seq_clear:N \l__stex_structures_fields_seq
4942         \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4943         \seq_map_inline:Nn \l_stex_collect_imports_seq {
4944             \seq_map_inline:cn {c_stex_module_##1_constants}{
4945                 \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4946             }

```

```

4947 }
4948 \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4949 \prop_clear:N \l_tmpa_prop
4950 \tl_if_empty:nF {#5} {
4951   \seq_set_split:Nnn \l_tmpa_seq , {#5}
4952   \seq_map_inline:Nn \l_tmpa_seq {
4953     \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4954     \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4955       \msg_error:nnn{stex}{error/keyval}{##1}
4956     }
4957     \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_structures_dom_str
4958     \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4959     \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_in_seq
4960     \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
4961     \stex_if_do_html:T{
4962       \stex_annotate:nnn{assign}{\l__stex_structures_dom_str,
4963         \bool_if:NTF\l_stex_symbol_or_var_bool{var://}{}\l_stex_get_symbol_uri_str}{}}
4964     }
4965     \bool_if:NTF \l_stex_symbol_or_var_bool {
4966       \exp_args:Nxx \str_if_eq:nnF
4967         {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4968         {\prop_item:cn{l_stex_variable\l_stex_get_symbol_uri_str _prop}{args}}{
4969         \msg_error:nnxxx{stex}{error/incompatible}
4970         {\l__stex_structures_dom_str}
4971         {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4972         {\l_stex_get_symbol_uri_str}
4973         {\prop_item:cn{l_stex_variable\l_stex_get_symbol_uri_str _prop}{args}}
4974       }
4975       \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n {
4976     }}{
4977       \exp_args:Nxx \str_if_eq:nnF
4978         {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4979         {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}{
4980         \msg_error:nnxxx{stex}{error/incompatible}
4981         {\l__stex_structures_dom_str}
4982         {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4983         {\l_stex_get_symbol_uri_str}
4984         {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}
4985       }
4986       \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n {
4987     }}
4988     }
4989   }
4990   \tl_gclear:N \g__stex_structures_aftergroup_tl
4991   \seq_map_inline:Nn \l__stex_structures_fields_seq {
4992     \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdecl\l__stex_structures_fields_seq ##1} \l_tmpa_str}
4993     \stex_debug:nn{varinstantiate}{Field~\l_tmpa_str :~##1}
4994     \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4995       \stex_find_notation:nn{##1}{}
4996       \cs_gset_eq:cc{g__stex_structures_tmpa\l_tmpa_str _cs}
4997         {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4998       \stex_debug:nn{varinstantiate}{Notation:~\cs_meaning:c{g__stex_structures_tmpa\l_tmpa_str _cs}}
4999       \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
5000         \cs_gset_eq:cc {g__stex_structures_tmpa_op\l_tmpa_str _cs}

```

```

5001         {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5002         \stex_debug:nn{varinstantiate}{Operator-Notation:~\cs_meaning:c{g__stex_struct
5003     }
5004 }
5005
5006 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
5007     \prop_set_from_keyval:cn { \l_stex_variable_ \l_tmpa_str _prop}{
5008         name      = \l_tmpa_str ,
5009         args      = \prop_item:cn { \l_stex_symdecl_##1_prop}{args} ,
5010         arity     = \prop_item:cn { \l_stex_symdecl_##1_prop}{arity} ,
5011         assocs    = \prop_item:cn { \l_stex_symdecl_##1_prop}{assocs}
5012     }
5013     \cs_set_eq:cc {stex_var_notation_\l_tmpa_str _cs}
5014     {g__stex_structures_tmpa_\l_tmpa_str _cs}
5015     \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str _cs}
5016     {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
5017 }
5018 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn { \l_stex_symdecl_##1_prop}{name}}{\stex_inv
5019 }
5020 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
5021     \prop_set_from_keyval:cn { \l_stex_varinstance_\l__stex_structures_name_str _prop }{
5022         domain = \l_stex_get_structure_module_str ,
5023         \prop_to_keyval:N \l_tmpa_prop
5024     }
5025     \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
5026     \tl_set:cn { \l_stex_varinstance_\l__stex_structures_name_str _op_tl}{
5027         \exp_args:Nnx \exp_not:N \use:nn {
5028             \str_set:Nn \exp_not:N \STEXInternalCurrentSymbolStr {var://\l__stex_structures_
5029             \_stex_term_omv:nn {var://\l__stex_structures_name_str}{
5030                 \exp_not:n{
5031                     \_varcomp{#4}
5032                 }
5033             }
5034         }{
5035             \exp_not:n{\_stex_reset:N \STEXInternalCurrentSymbolStr}
5036         }
5037     }
5038 }
5039 }
5040 \stex_debug:nn{varinstantiate}{\expandafter\detokenize\expandafter{\g__stex_structures_a
5041 \aftergroup\g__stex_structures_aftergroup_tl
5042 \endgroup
5043 \stex_smsmode_do:\ignorespacesandpars
5044 }
5045
5046 \cs_new_protected:Nn \stex_invoke_instance:n {
5047     \peek_charcode_remove:NTF ! {
5048         \stex_invoke_symbol:n{#1}
5049     }{
5050         \_stex_invoke_instance:nn {#1}
5051     }
5052 }
5053
5054

```



```

5055 \cs_new_protected:Nn \stex_invoke_varinstance:n {
5056   \peek_charcode_remove:NTF ! {
5057     \exp_args:Nnx \use:nn {
5058       \def\comp{\_varcomp}
5059       \use:c{l_stex_varinstance_#1_op_tl}
5060     }{
5061       \_stex_reset:N \comp
5062     }
5063   }{
5064     \_stex_invoke_varinstance:nn {#1}
5065   }
5066 }
5067
5068 \cs_new_protected:Nn \_stex_invoke_instance:nn {
5069   \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
5070     \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
5071   }{
5072     \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
5073     \msg_error:nnxxx{stex}{error/unknownfield}{#2}{#1}{
5074       \prop_to_keyval:N \l_tmpa_prop
5075     }
5076   }
5077 }
5078
5079 \cs_new_protected:Nn \_stex_invoke_varinstance:nn {
5080   \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
5081     \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
5082     \l_tmpa_tl
5083   }{
5084     \msg_error:nnnnn{stex}{error/unknownfield}{#2}{#1}{}
5085   }
5086 }

```

(End definition for \instantiate. This function is documented on page 33.)

\stex\_invoke\_structure:nnn

```

5087 % #1: URI of the instance
5088 % #2: URI of the instantiated module
5089 \cs_new_protected:Nn \stex_invoke_structure:nnn {
5090   \tl_if_empty:nTF{ #3 }{
5091     \prop_set_eq:Nc \l__stex_structures_structure_prop {
5092       c_stex_feature_ #2 _prop
5093     }
5094     \tl_clear:N \l_tmpa_tl
5095     \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
5096     \seq_map_inline:Nn \l_tmpa_seq {
5097       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
5098       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
5099       \cs_if_exist:cT {
5100         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
5101       }{
5102         \tl_if_empty:NF \l_tmpa_tl {
5103           \tl_put_right:Nn \l_tmpa_tl {,}
5104         }

```

```

5105         \tl_put_right:Nx \l_tmpa_tl {
5106             \stex_invoke_symbol:n {#1/\l_tmpa_str}!
5107         }
5108     }
5109 }
5110 \exp_args:No \mathstruct \l_tmpa_tl
5111 }{
5112     \stex_invoke_symbol:n{#1/#3}
5113 }
5114 }

```

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

```

5115 </package>

```

## Chapter 32

# STEX -Statements Implementation

```
5116 <*package>
5117
5118 %%%%%%%%%%% features.dtx %%%%%%%%%%%
5119
5120 <@@=stex_statements>
    Warnings and error messages
5121
\titleemph
5122 \def\titleemph#1{\textbf{#1}}
    (End definition for \titleemph. This function is documented on page ??.)
```

### 32.1 Definitions

#### definiendum

```
5123 \keys_define:nn {stex / definiendum }{
5124   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
5125   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
5126   root     .str_set_x:N    = \l__stex_statements_definiendum_root_str,
5127   gfa      .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
5128 }
5129 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
5130   \str_clear:N \l__stex_statements_definiendum_root_str
5131   \tl_clear:N \l__stex_statements_definiendum_post_tl
5132   \str_clear:N \l__stex_statements_definiendum_gfa_str
5133   \keys_set:nn { stex / definiendum }{ #1 }
5134 }
5135 \NewDocumentCommand \definiendum { O{} m m } {
5136   \__stex_statements_definiendum_args:n { #1 }
5137   \stex_get_symbol:n { #2 }
5138   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5139   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
5140     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```

5141     \tl_set:Nn \l_tmpa_tl { #3 }
5142   } {
5143     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
5144     \tl_set:Nn \l_tmpa_tl {
5145       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
5146     }
5147   }
5148 } {
5149   \tl_set:Nn \l_tmpa_tl { #3 }
5150 }
5151
5152 % TODO root
5153 \stex_html_backend:TF {
5154   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
5155 } {
5156   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
5157 }
5158 }
5159 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

(End definition for definiendum. This function is documented on page 42.)

#### definame

```

5160
5161 \NewDocumentCommand \definame { 0{ } m } {
5162   \__stex_statements_definiendum_args:n { #1 }
5163   % TODO: root
5164   \stex_get_symbol:n { #2 }
5165   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5166   \str_set:Nx \l_tmpa_str {
5167     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5168   }
5169   \str_replace_all:Nnn \l_tmpa_str {-} {~}
5170   \stex_html_backend:TF {
5171     \stex_if_do_html:T {
5172       \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5173         \l_tmpa_str\l__stex_statements_definiendum_post_tl
5174       }
5175     }
5176   } {
5177     \exp_args:Nnx \defemph@uri {
5178       \l_tmpa_str\l__stex_statements_definiendum_post_tl
5179     } { \l_stex_get_symbol_uri_str }
5180   }
5181 }
5182 \stex_deactivate_macro:Nn \definame {definition~environments}
5183
5184 \NewDocumentCommand \Definame { 0{ } m } {
5185   \__stex_statements_definiendum_args:n { #1 }
5186   \stex_get_symbol:n { #2 }
5187   \str_set:Nx \l_tmpa_str {
5188     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5189   }
5190   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}

```

```

5191 \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5192 \stex_html_backend:TF {
5193   \stex_if_do_html:T {
5194     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5195       \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5196     }
5197   }
5198 } {
5199   \exp_args:Nnx \defemph@uri {
5200     \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5201   } { \l_stex_get_symbol_uri_str }
5202 }
5203 }
5204 \stex_deactivate_macro:Nn \Definame {definition-environments}
5205
5206 \NewDocumentCommand \premise { m }{
5207   \noindent\stex_annotate:nnn{ premise }{}{\ignorespaces #1 }
5208 }
5209 \NewDocumentCommand \conclusion { m }{
5210   \noindent\stex_annotate:nnn{ conclusion }{}{\ignorespaces #1 }
5211 }
5212 \NewDocumentCommand \definiens { 0{} m }{
5213   \str_clear:N \l_stex_get_symbol_uri_str
5214   \tl_if_empty:nF {#1} {
5215     \stex_get_symbol:n { #1 }
5216   }
5217   \str_if_empty:NT \l_stex_get_symbol_uri_str {
5218     \int_compare:nNnTF {\clist_count:N \l__stex_statements_sdefinition_for_clist} = 1 {
5219       \str_set:Nx \l_stex_get_symbol_uri_str {\clist_item:Nn \l__stex_statements_sdefinition
5220     }{
5221       % TODO throw error
5222     }
5223   }
5224   \str_if_eq:eeT {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{module}}
5225   {\l_stex_current_module_str}{
5226     \str_if_eq:eeF {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defin
5227   }{true}{
5228     \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5229     \exp_args:Nx \stex_add_to_current_module:n {
5230       \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5231     }
5232   }
5233 }
5234 \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
5235 }
5236
5237 \NewDocumentCommand \varbindforall {m}{
5238   \stex_symbol_or_var:n {#1}
5239   \bool_if:NTF\l_stex_symbol_or_var_bool{
5240     \stex_if_do_html:T {
5241       \stex_annotate_invisible:nnn {bindtype}{forall,\l_stex_get_symbol_uri_str}{}
5242     }
5243   }{
5244     % todo throw error

```

```

5245 }
5246 }
5247
5248 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
5249 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
5250 \stex_deactivate_macro:Nn \definiens {definition~environments}
5251 \stex_deactivate_macro:Nn \varbindforall {definition~or~assertion~environments}
5252

```

(End definition for definame. This function is documented on page 42.)

sdefinition

```

5253
5254 \keys_define:nn {stex / sdefinition }{
5255   type      .str_set_x:N = \sdefinitiontype,
5256   id        .str_set_x:N = \sdefinitionid,
5257   name      .str_set_x:N = \sdefinitionname,
5258   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5259   title     .tl_set:N    = \sdefinitiontitle
5260 }
5261 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
5262   \str_clear:N \sdefinitiontype
5263   \str_clear:N \sdefinitionid
5264   \str_clear:N \sdefinitionname
5265   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5266   \tl_clear:N \sdefinitiontitle
5267   \keys_set:nn { stex / sdefinition }{ #1 }
5268 }
5269
5270 \NewDocumentEnvironment{sdefinition}{0{}}{
5271   \__stex_statements_sdefinition_args:n{ #1 }
5272   \stex_reactivate_macro:N \definiendum
5273   \stex_reactivate_macro:N \definame
5274   \stex_reactivate_macro:N \Definame
5275   \stex_reactivate_macro:N \premise
5276   \stex_reactivate_macro:N \definiens
5277   \stex_reactivate_macro:N \varbindforall
5278   \stex_if_smsmode:F{
5279     \seq_clear:N \l_tmpb_seq
5280     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5281       \tl_if_empty:nF{ ##1 }{
5282         \stex_get_symbol:n { ##1 }
5283         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5284           \l_stex_get_symbol_uri_str
5285         }
5286       }
5287     }
5288     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5289     \exp_args:Nnnx
5290     \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
5291     \str_if_empty:NF \sdefinitiontype {
5292       \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{ }
5293     }
5294     \str_if_empty:NF \sdefinitionname {

```

```

5295     \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5296   }
5297   \clist_set:No \l_tmpa_clist \sdefinitiontype
5298   \tl_clear:N \l_tmpa_tl
5299   \clist_map_inline:Nn \l_tmpa_clist {
5300     \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
5301       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
5302     }
5303   }
5304   \tl_if_empty:NTF \l_tmpa_tl {
5305     \__stex_statements_sdefinition_start:
5306   }{
5307     \l_tmpa_tl
5308   }
5309 }
5310 \stex_ref_new_doc_target:n \sdefinitionid
5311 \stex_smsmode_do:
5312 }{
5313   \stex_suppress_html:n {
5314     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5315   }
5316   \stex_if_smsmode:F {
5317     \clist_set:No \l_tmpa_clist \sdefinitiontype
5318     \tl_clear:N \l_tmpa_tl
5319     \clist_map_inline:Nn \l_tmpa_clist {
5320       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
5321         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
5322       }
5323     }
5324     \tl_if_empty:NTF \l_tmpa_tl {
5325       \__stex_statements_sdefinition_end:
5326     }{
5327       \l_tmpa_tl
5328     }
5329     \end{stex_annotate_env}
5330   }
5331 }

```

### **\stexpatchdefinition**

```

5332 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
5333   \stex_par:\noindent\titllemph{Definition\tl_if_empty:NF \sdefinitiontitle {
5334     ~(\sdefinitiontitle)
5335   }~}
5336 }
5337 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\stex_par:\medskip}
5338
5339 \newcommand\stexpatchdefinition[3]{} {
5340   \str_set:Nx \l_tmpa_str{ #1 }
5341   \str_if_empty:NTF \l_tmpa_str {
5342     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
5343     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
5344   }{
5345     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
5346     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }

```

```

5347     }
5348 }

```

(End definition for `\stexpatchdefinition`. This function is documented on page 48.)

`\inlinedef` inline:

```

5349 \keys_define:nn {stex / inlinedef }{
5350   type      .str_set_x:N = \sdefinitiontype,
5351   id        .str_set_x:N = \sdefinitionid,
5352   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5353   name      .str_set_x:N = \sdefinitionname
5354 }
5355 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
5356   \str_clear:N \sdefinitiontype
5357   \str_clear:N \sdefinitionid
5358   \str_clear:N \sdefinitionname
5359   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5360   \keys_set:nn { stex / inlinedef }{ #1 }
5361 }
5362 \NewDocumentCommand \inlinedef { 0{} m } {
5363   \begingroup
5364   \__stex_statements_inlinedef_args:n{ #1 }
5365   \stex_reactivate_macro:N \definiendum
5366   \stex_reactivate_macro:N \definame
5367   \stex_reactivate_macro:N \Definame
5368   \stex_reactivate_macro:N \premise
5369   \stex_reactivate_macro:N \definiens
5370   \stex_reactivate_macro:N \varbindforall
5371   \stex_ref_new_doc_target:n \sdefinitionid
5372   \stex_if_smsmode:TF{\stex_suppress_html:n {
5373     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5374   }}{
5375     \seq_clear:N \l_tmpb_seq
5376     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5377       \tl_if_empty:nF{ ##1 }{
5378         \stex_get_symbol:n { ##1 }
5379         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5380           \l_stex_get_symbol_uri_str
5381         }
5382       }
5383     }
5384     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5385     \exp_args:Nnx
5386     \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpb_seq {,}}{
5387       \str_if_empty:NF \sdefinitiontype {
5388         \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5389       }
5390       #2
5391       \str_if_empty:NF \sdefinitionname {
5392         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sdefinitionname}}
5393         \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5394       }
5395     }
5396   }

```



```

5397 \endgroup
5398 \stex_smsmode_do:
5399 }

```

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

## 32.2 Assertions

sassertion

```

5400 \keys_define:nn {stex / sassertion }{
5401   type      .str_set_x:N = \sassertiontype,
5402   id        .str_set_x:N = \sassertionid,
5403   title     .tl_set:N     = \sassertiontitle ,
5404   for       .clist_set:N  = \l__stex_statements_sassertion_for_clist ,
5405   name      .str_set_x:N  = \sassertionname
5406 }
5407
5408 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
5409   \str_clear:N \sassertiontype
5410   \str_clear:N \sassertionid
5411   \str_clear:N \sassertionname
5412   \clist_clear:N \l__stex_statements_sassertion_for_clist
5413   \tl_clear:N \sassertiontitle
5414   \keys_set:nn { stex / sassertion }{ #1 }
5415 }
5416
5417 %\tl_new:N \g__stex_statements_aftergroup_tl
5418
5419 \NewDocumentEnvironment{sassertion}{0}{}{
5420   \__stex_statements_sassertion_args:n{ #1 }
5421   \stex_reactivate_macro:N \premise
5422   \stex_reactivate_macro:N \conclusion
5423   \stex_reactivate_macro:N \varbindforall
5424   \stex_if_smsmode:F {
5425     \seq_clear:N \l_tmpb_seq
5426     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5427       \tl_if_empty:nF{ ##1 }{
5428         \stex_get_symbol:n { ##1 }
5429         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5430           \l_stex_get_symbol_uri_str
5431         }
5432       }
5433     }
5434     \exp_args:Nnnx
5435     \begin{sassertion_env}{assertion}{\seq_use:Nn \l_tmpb_seq {},}}
5436     \str_if_empty:NF \sassertiontype {
5437       \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
5438     }
5439     \str_if_empty:NF \sassertionname {
5440       \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
5441     }
5442     \clist_set:Nn \l_tmpa_clist \sassertiontype
5443     \tl_clear:N \l_tmpa_tl

```

```

5444 \clist_map_inline:Nn \l_tmpa_clist {
5445   \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
5446     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
5447   }
5448 }
5449 \tl_if_empty:NTF \l_tmpa_tl {
5450   \__stex_statements_sassertion_start:
5451 }{
5452   \l_tmpa_tl
5453 }
5454 }
5455 \str_if_empty:NTF \sassertionid {
5456   \str_if_empty:NF \sassertionname {
5457     \stex_ref_new_doc_target:n {}
5458   }
5459 } {
5460   \stex_ref_new_doc_target:n \sassertionid
5461 }
5462 \stex_smsmode_do:
5463 ){
5464   \str_if_empty:NF \sassertionname {
5465     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5466     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5467   }
5468   \stex_if_smsmode:F {
5469     \clist_set:No \l_tmpa_clist \sassertiontype
5470     \tl_clear:N \l_tmpa_tl
5471     \clist_map_inline:Nn \l_tmpa_clist {
5472       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
5473         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
5474       }
5475     }
5476     \tl_if_empty:NTF \l_tmpa_tl {
5477       \__stex_statements_sassertion_end:
5478     }{
5479       \l_tmpa_tl
5480     }
5481     \end{stex_annotate_env}
5482   }
5483 }

```

### **\stexpatchassertion**

```

5484
5485 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
5486   \stex_par:\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
5487     (\sassertiontitle)
5488   }~}
5489 }
5490 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\stex_par:\medskip}
5491
5492 \newcommand\stexpatchassertion[3]{} {
5493   \str_set:Nx \l_tmpa_str{ #1 }
5494   \str_if_empty:NTF \l_tmpa_str {
5495     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }

```

```

5496     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
5497   }{
5498     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
5499     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
5500   }
5501 }

```

(End definition for `\stexpatchassertion`. This function is documented on page 48.)

`\inlineass` inline:

```

5502 \keys_define:nn {stex / inlineass }{
5503   type      .str_set_x:N = \sassertiontype,
5504   id        .str_set_x:N = \sassertionid,
5505   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
5506   name      .str_set_x:N = \sassertionname
5507 }
5508 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
5509   \str_clear:N \sassertiontype
5510   \str_clear:N \sassertionid
5511   \str_clear:N \sassertionname
5512   \clist_clear:N \l__stex_statements_sassertion_for_clist
5513   \keys_set:nn { stex / inlineass }{ #1 }
5514 }
5515 \NewDocumentCommand \inlineass { 0{} m } {
5516   \beginngroup
5517   \stex_reactivate_macro:N \premise
5518   \stex_reactivate_macro:N \conclusion
5519   \stex_reactivate_macro:N \varbindforall
5520   \__stex_statements_inlineass_args:n{ #1 }
5521   \str_if_empty:NTF \sassertionid {
5522     \str_if_empty:NF \sassertionname {
5523       \stex_ref_new_doc_target:n {}
5524     }
5525   } {
5526     \stex_ref_new_doc_target:n \sassertionid
5527   }
5528
5529   \stex_if_smsmode:TF{
5530     \str_if_empty:NF \sassertionname {
5531       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sassertionname}}
5532     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5533   }
5534 }{
5535   \seq_clear:N \l_tmpb_seq
5536   \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5537     \tl_if_empty:nF{ ##1 }{
5538       \stex_get_symbol:n { ##1 }
5539       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5540         \l_stex_get_symbol_uri_str
5541       }
5542     }
5543   }
5544   \exp_args:Nnx
5545   \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpb_seq {,}}{

```

```

5546 \str_if_empty:NF \sassertiontype {
5547   \stex_annotate_invisible:nnn{typestrings}{\sassertiontype}{ }
5548 }
5549 #2
5550 \str_if_empty:NF \sassertionname {
5551   \stex_suppress_html:n{\stex_symdecl_do:nn}{\sassertionname}}
5552   \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5553   \stex_annotate_invisible:nnn{statementname}{\sassertionname}{ }
5554 }
5555 }
5556 }
5557 \endgroup
5558 \stex_smsmode_do:
5559 }

```

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

## 32.3 Examples

`sexample`

```

5560
5561 \keys_define:nn {stex / sexample }{
5562   type      .str_set_x:N = \exampletype,
5563   id        .str_set_x:N = \sexampleid,
5564   title     .tl_set:N    = \sexamplename,
5565   name      .str_set_x:N = \sexamplename ,
5566   for       .clist_set:N = \l__stex_statements_sexample_for_clist,
5567 }
5568 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
5569   \str_clear:N \sexampletype
5570   \str_clear:N \sexampleid
5571   \str_clear:N \sexamplename
5572   \tl_clear:N \sexamplename
5573   \clist_clear:N \l__stex_statements_sexample_for_clist
5574   \keys_set:nn { stex / sexample }{ #1 }
5575 }
5576
5577 \NewDocumentEnvironment{sexample}{0{}}{
5578   \__stex_statements_sexample_args:n{ #1 }
5579   \stex_reactivate_macro:N \premise
5580   \stex_reactivate_macro:N \conclusion
5581   \stex_if_smsmode:F {
5582     \seq_clear:N \l_tmpb_seq
5583     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5584       \tl_if_empty:nF{ ##1 }{
5585         \stex_get_symbol:n { ##1 }
5586         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5587           \l_stex_get_symbol_uri_str
5588         }
5589       }
5590     }
5591     \exp_args:Nnnx
5592     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpb_seq {,}}

```

```

5593 \str_if_empty:NF \sexamplotype {
5594   \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{}
5595 }
5596 \str_if_empty:NF \sexamplename {
5597   \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
5598 }
5599 \clist_set:No \l_tmpa_clist \sexamplotype
5600 \tl_clear:N \l_tmpa_tl
5601 \clist_map_inline:Nn \l_tmpa_clist {
5602   \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
5603     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
5604   }
5605 }
5606 \tl_if_empty:NTF \l_tmpa_tl {
5607   \__stex_statements_sexample_start:
5608 }{
5609   \l_tmpa_tl
5610 }
5611 }
5612 \str_if_empty:NF \sexampleid {
5613   \stex_ref_new_doc_target:n \sexampleid
5614 }
5615 \stex_smsmode_do:
5616 }{
5617   \str_if_empty:NF \sexamplename {
5618     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5619   }
5620   \stex_if_smsmode:F {
5621     \clist_set:No \l_tmpa_clist \sexamplotype
5622     \tl_clear:N \l_tmpa_tl
5623     \clist_map_inline:Nn \l_tmpa_clist {
5624       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
5625         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
5626       }
5627     }
5628     \tl_if_empty:NTF \l_tmpa_tl {
5629       \__stex_statements_sexample_end:
5630     }{
5631       \l_tmpa_tl
5632     }
5633     \end{stex_annotate_env}
5634   }
5635 }

```

**\stexpatchexample**

```

5636
5637 \cs_new_protected:Nn \__stex_statements_sexample_start: {
5638   \stex_par:\noindent\titllemph{Example~\tl_if_empty:NF \sexampltitle {
5639     (\sexampltitle)
5640   }~}
5641 }
5642 \cs_new_protected:Nn \__stex_statements_sexample_end: {\stex_par:\medskip}
5643
5644 \newcommand\stexpatchexample[3]{} {

```

```

5645 \str_set:Nx \l_tmpa_str{ #1 }
5646 \str_if_empty:NTF \l_tmpa_str {
5647   \tl_set:Nn \__stex_statements_sexample_start: { #2 }
5648   \tl_set:Nn \__stex_statements_sexample_end: { #3 }
5649 }{
5650   \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
5651   \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
5652 }
5653 }

```

(End definition for `\stexpatchexample`. This function is documented on page 48.)

`\inlineex` inline:

```

5654 \keys_define:nn {stex / inlineex }{
5655   type      .str_set_x:N = \sexamplotype,
5656   id        .str_set_x:N = \sexampleid,
5657   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
5658   name      .str_set_x:N = \sexamplename
5659 }
5660 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
5661   \str_clear:N \sexamplotype
5662   \str_clear:N \sexampleid
5663   \str_clear:N \sexamplename
5664   \clist_clear:N \l__stex_statements_sexample_for_clist
5665   \keys_set:nn { stex / inlineex }{ #1 }
5666 }
5667 \NewDocumentCommand \inlineex { 0{} m } {
5668   \begingroup
5669   \stex_reactivate_macro:N \premise
5670   \stex_reactivate_macro:N \conclusion
5671   \__stex_statements_inlineex_args:n{ #1 }
5672   \str_if_empty:NF \sexampleid {
5673     \stex_ref_new_doc_target:n \sexampleid
5674   }
5675   \stex_if_smsmode:TF{
5676     \str_if_empty:NF \sexamplename {
5677       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sexamplename}}
5678   }
5679 }{
5680   \seq_clear:N \l_tmpb_seq
5681   \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5682     \tl_if_empty:nF{ ##1 }{
5683       \stex_get_symbol:n { ##1 }
5684       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5685         \l_stex_get_symbol_uri_str
5686       }
5687     }
5688   }
5689   \exp_args:Nnx
5690   \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpb_seq {,}}{
5691     \str_if_empty:NF \sexamplotype {
5692       \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{ }
5693     }
5694     #2

```

```

5695     \str_if_empty:NF \sexamplename {
5696       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sexamplename}}
5697       \stex_annotate_invisible:nnn{statementname}{\sexamplename}{ }
5698     }
5699   }
5700 }
5701 \endgroup
5702 \stex_smsmode_do:
5703 }

```

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

## 32.4 Logical Paragraphs

`sparagraph`

```

5704 \keys_define:nn { stex / sparagraph } {
5705   id      .str_set:x:N = \sparagraphid ,
5706   title   .tl_set:N    = \l_stex_sparagraph_title_tl ,
5707   type    .str_set:x:N = \sparagraphtype ,
5708   for     .clist_set:N  = \l__stex_statements_sparagraph_for_clist ,
5709   from    .tl_set:N     = \sparagraphfrom ,
5710   to      .tl_set:N     = \sparagraphto ,
5711   start   .tl_set:N     = \l_stex_sparagraph_start_tl ,
5712   name    .str_set:N    = \sparagraphname ,
5713   imports .tl_set:N     = \l__stex_statements_sparagraph_imports_tl
5714 }
5715
5716 \cs_new_protected:Nn \stex_sparagraph_args:n {
5717   \tl_clear:N \l_stex_sparagraph_title_tl
5718   \tl_clear:N \sparagraphfrom
5719   \tl_clear:N \sparagraphto
5720   \tl_clear:N \l_stex_sparagraph_start_tl
5721   \tl_clear:N \l__stex_statements_sparagraph_imports_tl
5722   \str_clear:N \sparagraphid
5723   \str_clear:N \sparagraphtype
5724   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5725   \str_clear:N \sparagraphname
5726   \keys_set:nn { stex / sparagraph } { #1 }
5727 }
5728 \newif\if@in@omtext\@in@omtextfalse
5729
5730 \NewDocumentEnvironment {sparagraph} { 0{ } } {
5731   \stex_sparagraph_args:n { #1 }
5732   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5733     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
5734   }{
5735     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
5736   }
5737   \@in@omtexttrue
5738   \stex_if_smsmode:F {
5739     \seq_clear:N \l_tmpb_seq
5740     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5741       \tl_if_empty:nF{ ##1 }{

```

```

5742     \stex_get_symbol:n { ##1 }
5743     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5744         \l_stex_get_symbol_uri_str
5745     }
5746 }
5747 }
5748 \exp_args:Nnnx
5749 \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}
5750 \str_if_empty:NF \sparagraphtype {
5751     \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
5752 }
5753 \str_if_empty:NF \sparagraphfrom {
5754     \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5755 }
5756 \str_if_empty:NF \sparagraphto {
5757     \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5758 }
5759 \str_if_empty:NF \sparagraphname {
5760     \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
5761 }
5762 \clist_set:No \l_tmpa_clist \sparagraphtype
5763 \tl_clear:N \l_tmpa_tl
5764 \clist_map_inline:Nn \sparagraphtype {
5765     \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
5766         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
5767     }
5768 }
5769 \stex_csl_to_imports:No \usemodule \l__stex_statements_sparagraph_imports_tl
5770 \tl_if_empty:NTF \l_tmpa_tl {
5771     \__stex_statements_sparagraph_start:
5772 }{
5773     \l_tmpa_tl
5774 }
5775 }
5776 \clist_set:No \l_tmpa_clist \sparagraphtype
5777 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5778     {
5779         \stex_reactivate_macro:N \definiendum
5780         \stex_reactivate_macro:N \definame
5781         \stex_reactivate_macro:N \Definame
5782         \stex_reactivate_macro:N \premise
5783         \stex_reactivate_macro:N \definiens
5784     }
5785 \str_if_empty:NTF \sparagraphid {
5786     \str_if_empty:NTF \sparagraphname {
5787         \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5788             \stex_ref_new_doc_target:n {}
5789         }
5790     } {
5791         \stex_ref_new_doc_target:n {}
5792     }
5793 } {
5794     \stex_ref_new_doc_target:n \sparagraphid
5795 }

```



```

5796 \exp_args:NNx
5797 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5798   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5799     \tl_if_empty:nF{ ##1 }{
5800       \stex_get_symbol:n { ##1 }
5801       \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5802     }
5803   }
5804 }
5805 \stex_smsmode_do:
5806 \ignorespacesandpars
5807 }{
5808   \str_if_empty:NF \sparagraphname {
5809     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5810     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5811   }
5812   \stex_if_smsmode:F {
5813     \clist_set:No \l_tmpa_clist \sparagraphtype
5814     \tl_clear:N \l_tmpa_tl
5815     \clist_map_inline:Nn \l_tmpa_clist {
5816       \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
5817         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
5818       }
5819     }
5820     \tl_if_empty:NTF \l_tmpa_tl {
5821       \__stex_statements_sparagraph_end:
5822     }{
5823       \l_tmpa_tl
5824     }
5825     \end{stex_annotate_env}
5826   }
5827 }

```

## \stexpatchparagraph

```

5828
5829 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
5830   \stex_par:\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5831     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
5832       \titleemph{\l_stex_sparagraph_title_tl}:~
5833     }
5834   }{
5835     \titleemph{\l_stex_sparagraph_start_tl}~
5836   }
5837 }
5838 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\stex_par:\medskip}
5839
5840 \newcommand\stexpatchparagraph[3] [] {
5841   \str_set:Nx \l_tmpa_str{ #1 }
5842   \str_if_empty:NTF \l_tmpa_str {
5843     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
5844     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
5845   }{
5846     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2 }
5847     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }

```

```

5848     }
5849 }
5850
5851 \keys_define:nn { stex / inlinepara } {
5852   id      .str_set:N = \sparagraphid ,
5853   type    .str_set:N = \sparagraphtype ,
5854   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5855   from    .tl_set:N   = \sparagraphfrom ,
5856   to      .tl_set:N   = \sparagraphto ,
5857   name    .str_set:N   = \sparagraphname
5858 }
5859 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
5860   \tl_clear:N \sparagraphfrom
5861   \tl_clear:N \sparagraphto
5862   \str_clear:N \sparagraphid
5863   \str_clear:N \sparagraphtype
5864   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5865   \str_clear:N \sparagraphname
5866   \keys_set:nn { stex / inlinepara }{ #1 }
5867 }
5868 \NewDocumentCommand \inlinepara { O{} m } {
5869   \begingroup
5870     \__stex_statements_inlinepara_args:n{ #1 }
5871     \clist_set:Nn \l_tmpa_clist \sparagraphtype
5872     \str_if_empty:NTF \sparagraphid {
5873       \str_if_empty:NTF \sparagraphname {
5874         \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5875           \stex_ref_new_doc_target:n {}
5876         }
5877       } {
5878         \stex_ref_new_doc_target:n {}
5879       }
5880     } {
5881       \stex_ref_new_doc_target:n \sparagraphid
5882     }
5883     \stex_if_smsmode:TF{
5884       \str_if_empty:NF \sparagraphname {
5885         \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
5886         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5887       }
5888     }{
5889       \seq_clear:N \l_tmpb_seq
5890       \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5891         \tl_if_empty:nF{ ##1 }{
5892           \stex_get_symbol:n { ##1 }
5893           \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5894             \l_stex_get_symbol_uri_str
5895           }
5896         }
5897       }
5898       \exp_args:Nnx
5899       \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}{
5900         \str_if_empty:NF \sparagraphtype {
5901           \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{

```

```

5902     }
5903     \str_if_empty:NF \sparagraphfrom {
5904         \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5905     }
5906     \str_if_empty:NF \sparagraphto {
5907         \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5908     }
5909     \str_if_empty:NF \sparagraphname {
5910         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5911         \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
5912         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5913     }
5914     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5915         \clist_map_inline:Nn \l_tmpb_seq {
5916             \stex_ref_new_sym_target:n {##1}
5917         }
5918     }
5919     #2
5920 }
5921 }
5922 \endgroup
5923 \stex_smsmode_do:
5924 }
5925

```

(End definition for `\stexpatchparagraph`. This function is documented on page 48.)

```

5926 </package>

```

## Chapter 33

# The Implementation

```
5927 <*package>
5928 <@@=stex_sproof>
5929
5930 %%%%%%%%%%    sproof.dtx    %%%%%%%%%%
5931
```

### 33.1 Proofs

We first define some keys for the proof environment.

```
5932 \keys_define:nn { stex / spf } {
5933   id          .str_set_x:N = \spfid,
5934   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,
5935   from        .tl_set:N    = \l__stex_sproof_spf_from_tl ,
5936   proofend    .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
5937   type        .str_set_x:N = \spftype,
5938   title       .tl_set:N    = \spftitle,
5939   continues   .tl_set:N    = \l__stex_sproof_spf_continues_tl,
5940   functions   .tl_set:N    = \l__stex_sproof_spf_functions_tl,
5941   method      .tl_set:N    = \l__stex_sproof_spf_method_tl
5942 }
5943 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
5944   \str_clear:N \spfid
5945   \tl_clear:N \l__stex_sproof_spf_for_tl
5946   \tl_clear:N \l__stex_sproof_spf_from_tl
5947   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
5948   \str_clear:N \spftype
5949   \tl_clear:N \spftitle
5950   \tl_clear:N \l__stex_sproof_spf_continues_tl
5951   \tl_clear:N \l__stex_sproof_spf_functions_tl
5952   \tl_clear:N \l__stex_sproof_spf_method_tl
5953   \bool_set_false:N \l__stex_sproof_inc_counter_bool
5954   \keys_set:nn { stex / spf }{ #1 }
5955 }
```

```
\c__stex_sproof_flow_str We define this macro, so that we can test whether the display key has the value flow
5956 \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.

```

5957 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
5958 \cs_new_protected:Npn \sproofnumber {
5959   \int_set:Nn \l_tmpa_int {1}
5960   \bool_while_do:nn {
5961     \int_compare_p:nNn {
5962       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5963     } > 0
5964   }{
5965     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
5966     \int_incr:N \l_tmpa_int
5967   }
5968 }
5969 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
5970   \int_set:Nn \l_tmpa_int {1}
5971   \bool_while_do:nn {
5972     \int_compare_p:nNn {
5973       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5974     } > 0
5975   }{
5976     \int_incr:N \l_tmpa_int
5977   }
5978   \int_compare:nNnF \l_tmpa_int = 1 {
5979     \int_decr:N \l_tmpa_int
5980   }
5981   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
5982     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1
5983   }
5984 }
5985
5986 \cs_new_protected:Npn \__stex_sproof_add_counter: {
5987   \int_set:Nn \l_tmpa_int {1}
5988   \bool_while_do:nn {
5989     \int_compare_p:nNn {
5990       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5991     } > 0
5992   }{
5993     \int_incr:N \l_tmpa_int
5994   }
5995   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 1 }
5996 }
5997
5998 \cs_new_protected:Npn \__stex_sproof_remove_counter: {
5999   \int_set:Nn \l_tmpa_int {1}
6000   \bool_while_do:nn {

```

```

6001     \int_compare_p:nNn {
6002       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6003     } > 0
6004   }{
6005     \int_incr:N \l_tmpa_int
6006   }
6007   \int_decr:N \l_tmpa_int
6008   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 0 }
6009 }

```

**\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

```

6010 \def\sproof@box{
6011   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
6012 }
6013 \def\sproofend{
6014   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
6015     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
6016   }
6017 }

```

(End definition for \sproofend. This function is documented on page 47.)

**spf@\*@kw**

```

6018 \def\spf@proofsketch@kw{Proof~Sketch}
6019 \def\spf@proof@kw{Proof}
6020 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

6021 \AddToHook{begindocument}{
6022   \ltx@ifpackageloaded{babel}{
6023     \makeatletter
6024     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6025     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6026       \input{sproof-ngerman.ldf}
6027     }
6028     \clist_if_in:NnT \l_tmpa_clist {finnish}{
6029       \input{sproof-finnish.ldf}
6030     }
6031     \clist_if_in:NnT \l_tmpa_clist {french}{
6032       \input{sproof-french.ldf}
6033     }
6034     \clist_if_in:NnT \l_tmpa_clist {russian}{
6035       \input{sproof-russian.ldf}
6036     }
6037     \makeatother
6038   }{}
6039 }

```

**spfsketch**

```

6040 \newcommand\spfsketch[2] [] {
6041   \begin{group}
6042   \let \premise \stex_proof_premise:

```

```

6043 \__stex_sproof_spf_args:n{#1}
6044 \stex_if_smsmode:TF {
6045   \str_if_empty:NF \spfid {
6046     \stex_ref_new_doc_target:n \spfid
6047   }
6048 }{
6049   \seq_clear:N \l_tmpa_seq
6050   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6051     \tl_if_empty:nF{ ##1 }{
6052       \stex_get_symbol:n { ##1 }
6053       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6054         \l_stex_get_symbol_uri_str
6055       }
6056     }
6057   }
6058   \exp_args:Nnx
6059   \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
6060     \str_if_empty:NF \spftype {
6061       \stex_annotate_invisible:nnn{type}{\spftype}{
6062     }
6063     \clist_set:Nn \l_tmpa_clist \spftype
6064     \tl_set:Nn \l_tmpa_tl {
6065       \titleemph{
6066         \tl_if_empty:NTF \spftitle {
6067           \spf@proofsketch@kw
6068         }{
6069           \spftitle
6070         }
6071       }~
6072     }
6073     \clist_map_inline:Nn \l_tmpa_clist {
6074       \exp_args:Nn \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6075         \tl_clear:N \l_tmpa_tl
6076       }
6077     }
6078     \str_if_empty:NF \spfid {
6079       \stex_ref_new_doc_target:n \spfid
6080     }
6081     \l_tmpa_tl #2 \sproofend
6082   }
6083 }
6084 \endgroup
6085 \stex_smsmode_do:
6086 }
6087

```

(End definition for `spfsketch`. This function is documented on page 45.)

**spfeq** This is very similar to `spfsketch`, but uses a computation array<sup>1011</sup>

```

6088 \newenvironment{spfeq}[2][ ]{
6089   \__stex_sproof_spf_args:n{#1}

```

<sup>10</sup>EDNOTE: This should really be more like a tabular with an ensuremath in it. or invoke text on the last column

<sup>11</sup>EDNOTE: document above

```

6090 \let \premise \stex_proof_premise:
6091 \stex_if_smsmode:TF {
6092   \str_if_empty:NF \spfid {
6093     \stex_ref_new_doc_target:n \spfid
6094   }
6095 }{
6096   \seq_clear:N \l_tmpa_seq
6097   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6098     \tl_if_empty:nF{ ##1 }{
6099       \stex_get_symbol:n { ##1 }
6100       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6101         \l_stex_get_symbol_uri_str
6102       }
6103     }
6104   }
6105   \exp_args:Nnnx
6106   \begin{stex_annotate_env}{spfeq}{\seq_use:Nn \l_tmpa_seq {,}}
6107   \str_if_empty:NF \spftype {
6108     \stex_annotate_invisible:nnn{type}{\spftype}{}
6109   }
6110
6111   \clist_set:No \l_tmpa_clist \spftype
6112   \tl_clear:N \l_tmpa_tl
6113   \clist_map_inline:Nn \l_tmpa_clist {
6114     \tl_if_exist:cT {__stex_sproof_spfeq_##1_start:}{
6115       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_start:}}
6116     }
6117     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6118       \tl_set:Nn \l_tmpa_tl {\use:n{}}
6119     }
6120   }
6121   \tl_if_empty:NTF \l_tmpa_tl {
6122     \__stex_sproof_spfeq_start:
6123   }{
6124     \l_tmpa_tl
6125   }{~#2}
6126   \str_if_empty:NF \spfid {
6127     \stex_ref_new_doc_target:n \spfid
6128   }
6129   \begin{displaymath}\begin{array}{rc1l}
6130 \end{array}
6131 \stex_smsmode_do:
6132 }{
6133 \stex_if_smsmode:F {
6134   \end{array}\end{displaymath}
6135   \clist_set:No \l_tmpa_clist \spftype
6136   \tl_clear:N \l_tmpa_tl
6137   \clist_map_inline:Nn \l_tmpa_clist {
6138     \tl_if_exist:cT {__stex_sproof_spfeq_##1_end:}{
6139       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_end:}}
6140     }
6141   }
6142   \tl_if_empty:NTF \l_tmpa_tl {
6143     \__stex_sproof_spfeq_end:

```



```

6144     }{
6145         \l_tmpa_tl
6146     }
6147     \end{stex_annotate_env}
6148 }
6149 }
6150
6151 \cs_new_protected:Nn \__stex_sproof_spfeq_start: {
6152     \titleemph{
6153         \tl_if_empty:NTF \spftitle {
6154             \spf@proof@kw
6155         }{
6156             \spftitle
6157         }
6158     }:
6159 }
6160 \cs_new_protected:Nn \__stex_sproof_spfeq_end: {\sproofend}
6161
6162 \newcommand\stexpatchspfeq[3] [] {
6163     \str_set:Nx \l_tmpa_str{ #1 }
6164     \str_if_empty:NTF \l_tmpa_str {
6165         \tl_set:Nn \__stex_sproof_spfeq_start: { #2 }
6166         \tl_set:Nn \__stex_sproof_spfeq_end: { #3 }
6167     }{
6168         \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_start:\endcsname{ #2 }
6169         \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_end:\endcsname{ #3 }
6170     }
6171 }
6172

```

(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.

```

6173 \newenvironment{sproof}[2] []{
6174     \let \premise \stex_proof_premise:
6175     \intarray_gzero:N \l__stex_sproof_counter_intarray
6176     \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
6177     \__stex_sproof_spf_args:n{#1}
6178     \stex_if_smsmode:TF {
6179         \str_if_empty:NF \spfid {
6180             \stex_ref_new_doc_target:n \spfid
6181         }
6182     }{
6183         \seq_clear:N \l_tmpa_seq
6184         \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6185             \tl_if_empty:nF{ ##1 }{
6186                 \stex_get_symbol:n { ##1 }
6187                 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6188                     \l_stex_get_symbol_uri_str
6189                 }
6190             }
6191         }

```

```

6192 \exp_args:Nnnx
6193 \begin{stex_annotate_env}{sproof}{\seq_use:Nn \l_tmpa_seq {,}}
6194 \str_if_empty:NF \spftype {
6195   \stex_annotate_invisible:nnn{type}{\spftype}{}}
6196 }
6197
6198 \clist_set:No \l_tmpa_clist \spftype
6199 \tl_clear:N \l_tmpa_tl
6200 \clist_map_inline:Nn \l_tmpa_clist {
6201   \tl_if_exist:cT {__stex_sproof_sproof_##1_start:}{
6202     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_start:}}
6203   }
6204   \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6205     \tl_set:Nn \l_tmpa_tl {\use:n{}}
6206   }
6207 }
6208 \tl_if_empty:NTF \l_tmpa_tl {
6209   \__stex_sproof_sproof_start:
6210 }{
6211   \l_tmpa_tl
6212 }{~#2}
6213 \str_if_empty:NF \spfid {
6214   \stex_ref_new_doc_target:n \spfid
6215 }
6216 \begin{description}
6217 }
6218 \stex_smsmode_do:
6219 }{
6220 \stex_if_smsmode:F{
6221   \end{description}
6222   \clist_set:No \l_tmpa_clist \spftype
6223   \tl_clear:N \l_tmpa_tl
6224   \clist_map_inline:Nn \l_tmpa_clist {
6225     \tl_if_exist:cT {__stex_sproof_sproof_##1_end:}{
6226       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_end:}}
6227     }
6228   }
6229   \tl_if_empty:NTF \l_tmpa_tl {
6230     \__stex_sproof_sproof_end:
6231   }{
6232     \l_tmpa_tl
6233   }
6234   \end{stex_annotate_env}
6235 }
6236 }
6237
6238 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
6239   \par\noindent\titleemph{
6240     \tl_if_empty:NTF \spftype {
6241       \spf@proof@kw
6242     }{
6243       \spftype
6244     }
6245   }::

```

```

6246 }
6247 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
6248
6249 \newcommand\stexpatchproof[3] [] {
6250   \str_set:Nx \l_tmpa_str{ #1 }
6251   \str_if_empty:NTF \l_tmpa_str {
6252     \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
6253     \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
6254   }{
6255     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
6256     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
6257   }
6258 }

```

**\spfidea**

```

6259 \newcommand\spfidea[2] []{
6260   \__stex_sproof_spf_args:n{#1}
6261   \titleemph{
6262     \tl_if_empty:NTF \spftype {Proof-Idea}{
6263       \spftype
6264     } :
6265   }~#2
6266   \sproofend
6267 }

```

(End definition for \spfidea. This function is documented on page 45.)

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**

```

6268 \newenvironment{spfstep}[1] []{
6269   \__stex_sproof_spf_args:n{#1}
6270   \stex_if_smsmode:TF {
6271     \str_if_empty:NF \spfid {
6272       \stex_ref_new_doc_target:n \spfid
6273     }
6274   }{
6275     \@in@omtexttrue
6276     \clist_set:Nn \l_tmpa_clist \spftype
6277     \tl_set:Nn \l_tmpa_tl {
6278       \item[\sproofnumber]
6279       \bool_set_true:N \l__stex_sproof_inc_counter_bool
6280     }
6281     \clist_map_inline:Nn \l_tmpa_clist {
6282       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6283         \tl_clear:N \l_tmpa_tl
6284       }
6285     }
6286     \l_tmpa_tl
6287     \seq_clear:N \l_tmpa_seq
6288     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {

```

```

6289     \tl_if_empty:nF{ ##1 }{
6290         \stex_get_symbol:n { ##1 }
6291         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6292             \l_stex_get_symbol_uri_str
6293         }
6294     }
6295 }
6296 \exp_args:Nnnx
6297 \begin{stex_annotate_env}{spfstep}{\seq_use:Nn \l_tmpa_seq {,}}
6298 \str_if_empty:NF \spftype {
6299     \stex_annotate_invisible:nnn{type}{\spftype}{}}
6300 }
6301 \tl_if_empty:NF \spftitle {
6302     {(\titleemph{\spftitle})\enspace}
6303 }
6304 \str_if_empty:NF \spfid {
6305     \stex_ref_new_doc_target:n \spfid
6306 }
6307 }
6308 \stex_smsmode_do:
6309 \ignorespacesandpars
6310 }{
6311     \bool_if:NT \l__stex_sproof_inc_counter_bool {
6312         \__stex_sproof_inc_counter:
6313     }
6314     \stex_if_smsmode:F {
6315         \end{stex_annotate_env}
6316     }
6317 }

```

**spfcomment**

```

6318 \newenvironment{spfcomment}[1][ ]{
6319     \__stex_sproof_spf_args:n{#1}
6320     \clist_set:Nn \l_tmpa_clist \spftype
6321     \tl_set:Nn \l_tmpa_tl {
6322         \item[\sproofnumber]
6323         \bool_set_true:N \l__stex_sproof_inc_counter_bool
6324     }
6325     \clist_map_inline:Nn \l_tmpa_clist {
6326         \exp_args:Nn \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6327             \tl_clear:N \l_tmpa_tl
6328         }
6329     }
6330     \l_tmpa_tl
6331 }{
6332     \bool_if:NT \l__stex_sproof_inc_counter_bool {
6333         \__stex_sproof_inc_counter:
6334     }
6335 }

```

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.

```

6336 \newenvironment{subproof}[2][]{
6337   \_stex_sproof_spf_args:n{#1}
6338   \stex_if_smsmode:TF{
6339     \str_if_empty:NF \spfid {
6340       \stex_ref_new_doc_target:n \spfid
6341     }
6342   }{
6343     \seq_clear:N \l_tmpa_seq
6344     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6345       \tl_if_empty:nF{ ##1 }{
6346         \stex_get_symbol:n { ##1 }
6347         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6348           \l_stex_get_symbol_uri_str
6349         }
6350       }
6351     }
6352     \exp_args:Nnnx
6353     \begin{stex_annotate_env}{subproof}{\seq_use:Nn \l_tmpa_seq {},}
6354     \str_if_empty:NF \spftype {
6355       \stex_annotate_invisible:nnn{type}{\spftype}{}
6356     }
6357
6358     \clist_set:Nn \l_tmpa_clist \spftype
6359     \tl_set:Nn \l_tmpa_tl {
6360       \item[\sproofnumber]
6361       \bool_set_true:N \l__stex_sproof_inc_counter_bool
6362     }
6363     \clist_map_inline:Nn \l_tmpa_clist {
6364       \exp_args:Nn \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6365         \tl_clear:N \l_tmpa_tl
6366       }
6367     }
6368     \l_tmpa_tl
6369     \tl_if_empty:NF \spftitle {
6370       {(\titleemph{\spftitle})\enspace}
6371     }
6372     {~#2}
6373     \str_if_empty:NF \spfid {
6374       \stex_ref_new_doc_target:n \spfid
6375     }
6376   }
6377   \_stex_sproof_add_counter:
6378   \stex_smsmode_do:
6379 }{
6380   \_stex_sproof_remove_counter:
6381   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6382     \_stex_sproof_inc_counter:
6383   }
6384   \stex_if_smsmode:F{
6385     \end{stex_annotate_env}
6386   }
6387 }

```

**spfcases** In the **pfcases** environment, the start text is displayed as the first comment of the proof.

```

6388 \newenvironment{spfcases}[2] [] {
6389   \tl_if_empty:nTF{#1}{
6390     \begin{subproof}[method=by-cases]{#2}
6391   }{
6392     \begin{subproof}[#1,method=by-cases]{#2}
6393   }
6394 }{
6395   \end{subproof}
6396 }

```

**spfcase** In the pfcase environment, the start text is displayed specification of the case after the `\item`

```

6397 \newenvironment{spfcase}[2] [] {
6398   \__stex_sproof_spf_args:n{#1}
6399   \stex_if_smsmode:TF {
6400     \str_if_empty:NF \spfid {
6401       \stex_ref_new_doc_target:n \spfid
6402     }
6403   }{
6404     \seq_clear:N \l_tmpa_seq
6405     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6406       \tl_if_empty:nF{ ##1 }{
6407         \stex_get_symbol:n { ##1 }
6408         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6409           \l_stex_get_symbol_uri_str
6410         }
6411       }
6412     }
6413     \exp_args:Nnnx
6414     \begin{stex_annotate_env}{spfcase}{\seq_use:Nn \l_tmpa_seq {,}}
6415     \str_if_empty:NF \spftype {
6416       \stex_annotate_invisible:nnn{type}{\spftype}{ }
6417     }
6418     \clist_set:Nn \l_tmpa_clist \spftype
6419     \tl_set:Nn \l_tmpa_tl {
6420       \item[\sproofnumber]
6421       \bool_set_true:N \l__stex_sproof_inc_counter_bool
6422     }
6423     \clist_map_inline:Nn \l_tmpa_clist {
6424       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6425         \tl_clear:N \l_tmpa_tl
6426       }
6427     }
6428     \l_tmpa_tl
6429     \tl_if_empty:nF{#2}{
6430       \titleemph{#2}:~
6431     }
6432   }
6433   \__stex_sproof_add_counter:
6434   \stex_smsmode_do:
6435 }{
6436   \__stex_sproof_remove_counter:
6437   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6438     \__stex_sproof_inc_counter:

```

```

6439 }
6440 \stex_if_smsmode:F{
6441   \clist_set:No \l_tmpa_clist \spftype
6442   \tl_set:Nn \l_tmpa_tl{\sproofend}
6443   \clist_map_inline:Nn \l_tmpa_clist {
6444     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6445       \tl_clear:N \l_tmpa_tl
6446     }
6447   }
6448   \l_tmpa_tl
6449   \end{stex_annotate_env}
6450 }
6451 }

```

**spfcase** similar to **spfcase**, takes a third argument.

```

6452 \newcommand\spfcasesketch[3] [] {
6453   \begin{spfcase}[#1]{#2}#3\end{spfcase}
6454 }

```

## 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.

```

6455 \keys_define:nn { stex / just }{
6456   id      .str_set_x:N = \l__stex_sproof_just_id_str,
6457   method  .tl_set:N   = \l__stex_sproof_just_method_tl,
6458   premises .tl_set:N   = \l__stex_sproof_just_premises_tl,
6459   args    .tl_set:N   = \l__stex_sproof_just_args_tl
6460 }

```

The next three environments and macros are purely semantic, so we ignore the keyval arguments for now and only display the content.<sup>12</sup>

**\spfjust**

```

6461 \newcommand\spfjust[1] [] {}

```

(End definition for **\spfjust**. This function is documented on page 46.)

**\premise**

```

6462 \newcommand\stex_proof_premise:[2] [] {#2}

```

(End definition for **\premise**. This function is documented on page 46.)

**\justarg** the **\justarg** macro is purely semantic, so we ignore the keyval arguments for now and only display the content.

```

6463 \newcommand\justarg[2] [] {#2}
6464 \</package>

```

(End definition for **\justarg**. This function is documented on page 46.)

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

---

<sup>12</sup>EdNOTE: need to do something about the premise in draft mode.

## Chapter 34

# STEX -Others Implementation

```
6465 <*package>
6466
6467 %%%%%%%%%%% others.dtx %%%%%%%%%%%
6468
6469 <@@=stex_others>
        Warnings and error messages
6470 % None

\MSC Math subject classifier

6471 \NewDocumentCommand \MSC {m} {
6472 % TODO
6473 }

(End definition for \MSC. This function is documented on page ??.)
        Patching tikzinput, if loaded
6474 \@ifpackageloaded{tikzinput}{
6475 \RequirePackage{stex-tikzinput}
6476 }{}
6477
6478 \bool_if:NT \c_stex_persist_mode_bool {
6479 \let__stex_notation_restore_notation_old:nnnnn
6480 \__stex_notation_restore_notation:nnnnn
6481 \def__stex_notation_restore_notation_new:nnnnn#1#2#3#4#5{
6482 \__stex_notation_restore_notation_old:nnnnn{#1}{#2}{#3}{#4}{#5}
6483 \ExplSyntaxOn
6484 }
6485 \def__stex_notation_restore_notation:nnnnn{
6486 \ExplSyntaxOff
6487 \catcode'\sim10
6488 \__stex_notation_restore_notation_new:nnnnn
6489 }
6490 \input{\jobname.sms}
6491 \let__stex_notation_restore_notation:nnnnn
6492 \__stex_notation_restore_notation_old:nnnnn
6493 \prop_if_exist:NT\c_stex_mathhub_main_manifest_prop{
```



```

6494     \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
6495     \l_tmpa_str
6496     \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str_manifest_prop }
6497     \c_stex_mathhub_main_manifest_prop
6498     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
6499   }
6500 }
6501 </package>

```

## Chapter 35

# STEX -Metatheory Implementation

```
6502 <*package>
6503 <@@=stex_modules>
6504
6505 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
6506
6507 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX/meta}
6508 \begingroup
6509 \stex_module_setup:nn{
6510   ns=\c_stex_metatheory_ns_str,
6511   meta=NONE
6512 }{Metatheory}
6513 \stex_reactivate_macro:N \symdecl
6514 \stex_reactivate_macro:N \notation
6515 \stex_reactivate_macro:N \symdef
6516 \ExplSyntaxOff
6517 \csname stex_suppress_html:n\endcsname{
6518   % is-a (a:A, a \in A, a is an A, etc.)
6519   \symdecl{isa}[args=ai]
6520   \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
6521   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
6522   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
6523
6524   % bind (\forall, \Pi, \lambda etc.)
6525   \symdecl{bind}[args=Bi,assoc=pre]
6526   \notation{bind}[depfun,prec=nobrackets,op={(\cdot)\;\to\;\cdot}]{\comp( #1 \comp{}\;\to\;)}
6527   \notation{bind}[forall]{\comp\forall #1.\;#2}{##1 \comp, ##2}
6528   \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6529
6530   % implicit bind
6531   \symdecl{implicitbind}[args=Bi,assoc=pre]
6532   \notation{implicitbind}[braces,prec=nobrackets,op={(\cdot\;_I\;\cdot)}]{\comp\{ #1 \comp{
6533   \notation{implicitbind}[depfun,prec=nobrackets]{\comp( #1 \comp{}\;\to_I\; } #2}{##1 \comp,
6534   \notation{implicitbind}[Pi]{\comp\prod^I_{#1}#2}{##1\comp,##2}
6535
6536   % dummy variable
```

```

6537 \symdecl{dummyvar}
6538 \notation{dummyvar}[underscore]{\comp\_}
6539 \notation{dummyvar}[dot]{\comp\cdot}
6540 \notation{dummyvar}[dash]{\comp{\rm --}}
6541
6542 %fromto (function space, Hom-set, implication etc.)
6543 \symdecl{fromto}[args=ai]
6544 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
6545 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
6546
6547 % mapto (lambda etc.)
6548 \symdecl{mapto}[args=Bi]
6549 %\notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
6550 %\notation{mapto}[lambda]{\comp\lambda #1 \comp.\; #2}{#1 \comp, #2}
6551 %\notation{mapto}[lambdau]{\comp\lambda_#1 \comp.\; #2}{#1 \comp, #2}
6552
6553 % function/operator application
6554 \symdecl{apply}[args=ia]
6555 \notation{apply}[prec=0;0x\infpres,parens,op=\cdot(\cdot)]{#1 \comp( #2 \comp)}{##1 \comp,
6556 \notation{apply}[prec=0;0x\infpres,lambda]{#1 \; #2 }{##1 \; ; ##2}
6557
6558 % collection of propositions/booleans/truth values
6559 \symdecl{prop}[name=proposition]
6560 \notation{prop}[prop]{\comp{\rm prop}}
6561 \notation{prop}[BOOL]{\comp{\rm BOOL}}
6562
6563 \symdecl{judgmentholds}[args=1]
6564 \notation{judgmentholds}[vdash,op=\vdash]{\comp\vdash\; #1}
6565
6566 % sequences
6567 \symdecl{seqtype}[args=1]
6568 \notation{seqtype}[kleene]{#1^{\comp\ast}}
6569
6570 \symdecl{seqexpr}[args=a]
6571 \notation{seqexpr}[angle,prec=nobrackets]{\comp\langle #1\comp\rangle}{##1\comp,##2}
6572
6573 \symdef{seqmap}[args=abi,setlike]{\comp\{#3 \comp| #2\comp\in \dobrackets{#1} \comp\}}{##1
6574 \symdef{seqprepend}[args=ia]{#1 \comp{::} #2}{##1 \comp, ##2}
6575 \symdef{seqappend}[args=ai]{#1 \comp{::} #2}{##1 \comp, ##2}
6576 \symdef{seqfoldleft}[args=iabbi]{ \comp{foldl}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\comp}
6577 \symdef{seqfoldright}[args=iabbi,op=foldr]{ \comp{foldr}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\comp}
6578 \symdef{seqhead}[args=a]{\comp{head}\dobrackets{#1}}{##1 \comp, ##2}
6579 \symdef{seqtail}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6580 \symdef{seqlast}[args=a]{\comp{last}\dobrackets{#1}}{##1 \comp, ##2}
6581 \symdef{seqinit}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6582
6583 \symdef{sequence-index}[args=2,li,prec=nobrackets]{\comp{#1}_{#2}}
6584 \notation{sequence-index}[ui,prec=nobrackets]{\comp{#1}^{\comp{#2}}}
6585
6586 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{\,\ellipses}}{##1\comp,##2}
6587 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{\,\ellipses,}#2}{##1\comp,##2}
6588 \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{\,\ellipses,}#2\comp{\,\ellipses,}#3}{##1\comp,##2,##3}
6589
6590 % nat literals

```

```

6591 \symdef{natliteral}{\comp{\mathtt{Ord}}}
6592
6593 % letin (''let'', local definitions, variable substitution)
6594 \symdecl{letin}[args=bii]
6595 \notation{letin}{let}{\comp{\rm let}}\;#1\comp{=}\#2\;\comp{\rm in}}\;#3}
6596 \notation{letin}{subst}{#3 \comp[ #1 \comp/ #2 \comp]}
6597 \notation{letin}{frac}{#3 \comp[ \frac{#2}{#1} \comp]}
6598
6599 % structures
6600 \symdecl*{module-type}[args=1]
6601 \notation{module-type}{\comp{\mathtt{MOD}}}\;#1}
6602 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
6603 \notation{mathstruct}[angle,prec=nobrackets]{\comp\angle #1 \comp\rangle}{##1 \comp, ##2}
6604
6605 % objects
6606 \symdecl{object}
6607 \notation{object}{\comp{\mathtt{OBJECT}}}
6608
6609 }
6610
6611 % The following are abbreviations in the sTeX corpus that are left over from earlier
6612 % developments. They will eventually be phased out.
6613
6614 \ExplSyntaxOn
6615 \stex_add_to_current_module:n{
6616   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
6617   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
6618   \def\livar{\csname sequence-index\endcsname[li]}
6619   \def\uivar{\csname sequence-index\endcsname[ui]}
6620   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
6621   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
6622 }
6623 \__stex_modules_end_module:
6624 \endgroup
6625 \</package>

```

## Chapter 36

# Tikzinput Implementation

```
6626 <@@=tikzinput>
6627 <*package>
6628
6629 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
6630
6631 \ProvidesExplPackage{tikzinput}{2022/02/26}{3.0.1}{tikzinput package}
6632 \RequirePackage{l3keys2e}
6633
6634 \keys_define:nn { tikzinput } {
6635   image .bool_set:N = \c_tikzinput_image_bool,
6636   image .default:n = false ,
6637   unknown .code:n = {}
6638 }
6639
6640 \ProcessKeysOptions { tikzinput }
6641
6642 \bool_if:NTF \c_tikzinput_image_bool {
6643   \RequirePackage{graphicx}
6644
6645   \providecommand\usetikzlibrary[]{}
6646   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
6647 }{
6648   \RequirePackage{tikz}
6649   \RequirePackage{standalone}
6650
6651   \newcommand \tikzinput [2] [] {
6652     \setkeys{Gin}{#1}
6653     \ifx \Gin@ewidth \Gin@exclamation
6654       \ifx \Gin@eheight \Gin@exclamation
6655         \input { #2 }
6656       \else
6657         \resizebox{!}{ \Gin@eheight }{
6658           \input { #2 }
6659         }
6660       \fi
6661     \else
6662       \ifx \Gin@eheight \Gin@exclamation
6663         \resizebox{ \Gin@ewidth }{!}{
```

```

6664         \input { #2 }
6665     }
6666     \else
6667         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
6668             \input { #2 }
6669         }
6670     \fi
6671 \fi
6672 }
6673 }
6674
6675 \newcommand \ctikzinput [2] [] {
6676     \begin{center}
6677         \tikzinput [#1] {#2}
6678     \end{center}
6679 }
6680
6681 \@ifpackageloaded{stex}{
6682     \RequirePackage{stex-tikzinput}
6683 }{}
6684
6685 </package>
6686 <*stex>
6687
6688 \ProvidesExplPackage{stex-tikzinput}{2022/02/26}{3.0.1}{stex-tikzinput}
6689 \RequirePackage{stex}
6690 \RequirePackage{tikzinput}
6691
6692 \newcommand\mhtikzinput[2] []{%
6693     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
6694     \stex_in_repository:nn\Gin@mhrepos{
6695         \tikzinput[#1]{\mhp@hpath{##1}{#2}}
6696     }
6697 }
6698
6699 \newcommand\cmhtikzinput[2] []{\begin{center}\mhtikzinput[#1]{#2}\end{center}}
6700
6701 \cs_new_protected:Nn \__tikzinput_usetikzlibrary:nn {
6702     \pgfkeys@spdef\pgf@temp{#1}
6703     \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
6704     \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgf@temp
6705     \expandafter\edef\csname tikz@library@#1@atcode\endcsname{\the\catcode'\@}
6706     \expandafter\edef\csname tikz@library@#1@barcode\endcsname{\the\catcode'\|}
6707     \expandafter\edef\csname tikz@library@#1@dollarcode\endcsname{\the\catcode'\$}
6708     \catcode'\@=11
6709     \catcode'\|=12
6710     \catcode'\$=3
6711     \pgfutil@InputIfFileExists{#2}{-}{-}
6712     \catcode'\@=\csname tikz@library@#1@atcode\endcsname
6713     \catcode'\|=\csname tikz@library@#1@barcode\endcsname
6714     \catcode'\$=\csname tikz@library@#1@dollarcode\endcsname
6715 }
6716
6717 \newcommand\libusetikzlibrary[1]{

```

```

6717 \prop_if_exist:NF \l_stex_current_repository_prop {
6718   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6719 }
6720 \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
6721   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6722 }
6723 \seq_clear:N \l__tikzinput_libinput_files_seq
6724 \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
6725 \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
6726
6727 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
6728   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / tikzlibrary
6729   \IfFileExists{ \l_tmpa_str }{
6730     \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6731   }{
6732     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
6733     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
6734   }
6735
6736   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / tikzlibrary #1 .code.t
6737   \IfFileExists{ \l_tmpa_str }{
6738     \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6739   }{
6740
6741   \seq_if_empty:NTF \l__tikzinput_libinput_files_seq {
6742     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .code.t
6743   }{
6744     \int_compare:nNnTF {\seq_count:N \l__tikzinput_libinput_files_seq} = 1 {
6745       \seq_map_inline:Nn \l__tikzinput_libinput_files_seq {
6746         \__tikzinput_usetikzlibrary:nn{#1}{ ##1 }
6747       }
6748     }{
6749       \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .co
6750     }
6751   }
6752 }
6753 </stex>

```

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

## Chapter 37

# document-structure.sty Implementation

```
6754 \*package>
6755 \@@=document_structure>
6756 \ProvidesExplPackage{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure}
6757 \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).

```
6758
6759 \keys_define:nn{ document-structure }{
6760   class      .str_set_x:N = \c_document_structure_class_str,
6761   topsect    .str_set_x:N = \c_document_structure_topsect_str,,
6762   unknown    .code:n      = {
6763     \PassOptionsToClass{\CurrentOption}{stex}
6764     \PassOptionsToClass{\CurrentOption}{tikzinput}
6765   }
6766   % showignores .bool_set:N = \c_document_structure_showignores_bool,
6767 }
6768 \ProcessKeysOptions{ document-structure }
6769 \str_if_empty:NT \c_document_structure_class_str {
6770   \str_set:Nn \c_document_structure_class_str {article}
6771 }
6772 \str_if_empty:NT \c_document_structure_topsect_str {
6773   \str_set:Nn \c_document_structure_topsect_str {section}
6774 }
```

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

```
6775 \RequirePackage{xspace}
6776 \RequirePackage{comment}
6777 \RequirePackage{stex}
6778 \AddToHook{begindocument}{}
```



```

6779 \ltx@ifpackageloaded{babel}{
6780   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6781   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6782     \makeatletter\input{document-structure-ngerman.ldf}\makeatother
6783   }
6784 }{}
6785 }

```

`\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.

```

6786 \int_new:N \l_document_structure_section_level_int
6787 \str_case:NnF \c_document_structure_topsect_str {
6788   {part}}{
6789     \int_set:Nn \l_document_structure_section_level_int {0}
6790   }
6791   {chapter}{
6792     \int_set:Nn \l_document_structure_section_level_int {1}
6793   }
6794 }{
6795   \str_case:NnF \c_document_structure_class_str {
6796     {book}{
6797       \int_set:Nn \l_document_structure_section_level_int {0}
6798     }
6799     {report}{
6800       \int_set:Nn \l_document_structure_section_level_int {0}
6801     }
6802   }{
6803     \int_set:Nn \l_document_structure_section_level_int {2}
6804   }
6805 }

```

## 37.2 Document Structure

The structure of the document is given by the `sfragment` environment. The hierarchy is adjusted automatically according to the L<sup>A</sup>T<sub>E</sub>X class in effect.

`\currentsectionlevel` For the `\currentsectionlevel` and `\Currentsectionlevel` macros we use an internal macro `\current@section@level` that only contains the keyword (no markup). We initialize it with “document” as a default. In the generated OMDoc, we only generate a text element of class `omdoc_currentsectionlevel`, which will be instantiated by CSS later.<sup>13</sup>

EdN:13

```

6806 \def\current@section@level{document}%
6807 \newcommand\currentsectionlevel{\lowercase\expandafter\current@section@level\xspace}%
6808 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%

```

(End definition for `\currentsectionlevel`. This function is documented on page 53.)

`\skipfragment`

```

6809 \cs_new_protected:Npn \skipfragment {

```

---

<sup>13</sup>EdNOTE: MK: we may have to experiment with the more powerful uppercasing macro from `mfirstuc.sty` once we internationalize.

```

6810 \ifcase\l_document_structure_section_level_int
6811 \or\stepcounter{part}
6812 \or\stepcounter{chapter}
6813 \or\stepcounter{section}
6814 \or\stepcounter{subsection}
6815 \or\stepcounter{subsubsection}
6816 \or\stepcounter{paragraph}
6817 \or\stepcounter{subparagraph}
6818 \fi
6819 }

```

(End definition for `\skipfragment`. This function is documented on page 52.)

**blindfragment**

```

6820 \newcommand\at@begin@blindsfragment[1]{
6821 \newenvironment{blindfragment}
6822 {
6823 \int_incr:N\l_document_structure_section_level_int
6824 \at@begin@blindsfragment\l_document_structure_section_level_int
6825 }{}

```

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

```

6826 \newcommand\sfragment@nonum[2]{
6827 \ifx\hyper@anchor\@undefined\else\phantomsection\fi
6828 \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
6829 }

```

(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.

```

6830 \newcommand\sfragment@num[2]{
6831 \tl_if_empty:NTF \l__document_structure_sfragment_short_tl {
6832 \@nameuse{#1}{#2}
6833 }{
6834 \cs_if_exist:NTF\rdfmata@sectioning{
6835 \@nameuse{rdfmata@#1@old}[\l__document_structure_sfragment_short_tl]{#2}
6836 }{
6837 \@nameuse{#1}[\l__document_structure_sfragment_short_tl]{#2}
6838 }
6839 }
6840 %\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\sfragment@id
6841 }

```

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

**sfragment**

```

6842 \keys_define:nn { document-structure / sfragment }{
6843 id .str_set_x:N = \l__document_structure_sfragment_id_str,
6844 date .str_set_x:N = \l__document_structure_sfragment_date_str,

```

```

6845 creators      .clist_set:N = \l__document_structure_sfragment_creators_clist,
6846 contributors  .clist_set:N = \l__document_structure_sfragment_contributors_clist,
6847 srccite       .tl_set:N     = \l__document_structure_sfragment_srccite_tl,
6848 type         .tl_set:N     = \l__document_structure_sfragment_type_tl,
6849 short        .tl_set:N     = \l__document_structure_sfragment_short_tl,
6850 display      .tl_set:N     = \l__document_structure_sfragment_display_tl,
6851 intro       .tl_set:N     = \l__document_structure_sfragment_intro_tl,
6852 imports     .tl_set:N     = \l__document_structure_sfragment_imports_tl,
6853 loadmodules  .bool_set:N   = \l__document_structure_sfragment_loadmodules_bool
6854 }
6855 \cs_new_protected:Nn \l__document_structure_sfragment_args:n {
6856   \str_clear:N \l__document_structure_sfragment_id_str
6857   \str_clear:N \l__document_structure_sfragment_date_str
6858   \clist_clear:N \l__document_structure_sfragment_creators_clist
6859   \clist_clear:N \l__document_structure_sfragment_contributors_clist
6860   \tl_clear:N \l__document_structure_sfragment_srccite_tl
6861   \tl_clear:N \l__document_structure_sfragment_type_tl
6862   \tl_clear:N \l__document_structure_sfragment_short_tl
6863   \tl_clear:N \l__document_structure_sfragment_display_tl
6864   \tl_clear:N \l__document_structure_sfragment_imports_tl
6865   \tl_clear:N \l__document_structure_sfragment_intro_tl
6866   \bool_set_false:N \l__document_structure_sfragment_loadmodules_bool
6867   \keys_set:nn { document-structure / sfragment } { #1 }
6868 }

```

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.

```

6869 \newif@if@mainmatter\@mainmattertrue
6870 \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.

```

6871 \keys_define:nn { document-structure / sectioning }{
6872   name      .str_set_x:N = \l__document_structure_sect_name_str ,
6873   ref       .str_set_x:N = \l__document_structure_sect_ref_str  ,
6874   clear     .bool_set:N  = \l__document_structure_sect_clear_bool ,
6875   clear     .default:n   = {true} ,
6876   num       .bool_set:N  = \l__document_structure_sect_num_bool  ,
6877   num       .default:n   = {true}
6878 }
6879 \cs_new_protected:Nn \l__document_structure_sect_args:n {
6880   \str_clear:N \l__document_structure_sect_name_str
6881   \str_clear:N \l__document_structure_sect_ref_str
6882   \bool_set_false:N \l__document_structure_sect_clear_bool
6883   \bool_set_false:N \l__document_structure_sect_num_bool
6884   \keys_set:nn { document-structure / sectioning } { #1 }
6885 }
6886 \newcommand\omdoc@sectioning[3][]{
6887   \l__document_structure_sect_args:n {#1 }
6888   \let\omdoc@sect@name\l__document_structure_sect_name_str
6889   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
6890   \if@mainmatter% numbering not overridden by frontmatter, etc.
6891     \bool_if:NTF \l__document_structure_sect_num_bool {

```

```

6892     \sfragment@num{#2}{#3}
6893   }{
6894     \sfragment@nonum{#2}{#3}
6895   }
6896   \def\current@section@level{\omdoc@sect@name}
6897 \else
6898   \sfragment@nonum{#2}{#3}
6899 \fi
6900 }% 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.

```

6901 \newcommand\sfragment@redefine@addtocontents[1]{%
6902   \edef\__document_structureimport{#1}%
6903   \@for\@I:=\__document_structureimport\do{%
6904     \edef\@path{\csname module@\@I @path\endcsname}%
6905     \@ifundefined{tf@toc}\relax%
6906     %    {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
6907     %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
6908     %\def\addcontentsline##1##2##3{%
6909     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6910     %\else% hyperref.sty not loaded
6911     %\def\addcontentsline##1##2##3{%
6912     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6913     %\fi
6914   }% hyperref.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.

```

6915 \newenvironment{sfragment}[2][ ]% keys, title
6916 {
6917   \__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.

```

6918   \stex_csl_to_imports:No \usemodule \l__document_structure_sfragment_imports_tl
6919
6920   \bool_if:NT \l__document_structure_sfragment_loadmodules_bool {
6921     \sfragment@redefine@addtocontents{
6922       %\@ifundefined{module@id}\used@modules%
6923       %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}
6924     }
6925   }

```

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

```

6926
6927   \stex_document_title:n { #2 }
6928
6929   \int_incr:N\l__document_structure_section_level_int
6930   \ifcase\l__document_structure_section_level_int
6931     \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
6932     \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}

```

```

6933 \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
6934 \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
6935 \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
6936 \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
6937 \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
6938 \fi
6939 \at@begin@sfragment[#1]\l_document_structure_section_level_int{#2}
6940 \str_if_empty:NF \l__document_structure_sfragment_id_str {
6941   \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
6942 }
6943 }% for customization
6944 {}

```

and finally, we localize the sections

```

6945 \newcommand\omdoc@part@kw{Part}
6946 \newcommand\omdoc@chapter@kw{Chapter}
6947 \newcommand\omdoc@section@kw{Section}
6948 \newcommand\omdoc@subsection@kw{Subsection}
6949 \newcommand\omdoc@subsubsection@kw{Subsubsection}
6950 \newcommand\omdoc@paragraph@kw{paragraph}
6951 \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`

```

6952 \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).

```

6953 \cs_if_exist:NTF\frontmatter{
6954   \let\__document_structure_orig_frontmatter\frontmatter
6955   \let\frontmatter\relax
6956 }{
6957   \tl_set:Nn\__document_structure_orig_frontmatter{
6958     \clearpage
6959     \@mainmatterfalse
6960     \pagenumbering{roman}
6961   }
6962 }
6963 \cs_if_exist:NTF\backmatter{
6964   \let\__document_structure_orig_backmatter\backmatter
6965   \let\backmatter\relax
6966 }{
6967   \tl_set:Nn\__document_structure_orig_backmatter{
6968     \clearpage
6969     \@mainmatterfalse

```

```

6970 \pagenumbering{roman}
6971 }
6972 }

```

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.

```

6973 \newenvironment{frontmatter}{
6974   \_document_structure_orig_frontmatter
6975 }{
6976   \cs_if_exist:NTF\mainmatter{
6977     \mainmatter
6978   }{
6979     \clearpage
6980     \@mainmattertrue
6981     \pagenumbering{arabic}
6982   }
6983 }

```

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

```

6984 \newenvironment{backmatter}{
6985   \_document_structure_orig_backmatter
6986 }{
6987   \cs_if_exist:NTF\mainmatter{
6988     \mainmatter
6989   }{
6990     \clearpage
6991     \@mainmattertrue
6992     \pagenumbering{arabic}
6993   }
6994 }

```

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

```

6995 \@mainmattertrue\pagenumbering{arabic}

```

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

```

6996 \def \c__document_structure_document_str{document}
6997 \newcommand\afterprematurestop{}
6998 \def\prematurestop@endsfragment{
6999   \unless\ifx\@currenvir\c__document_structure_document_str
7000     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
7001     \expandafter\prematurestop@endsfragment
7002   \fi
7003 }
7004 \providecommand\prematurestop{
7005   \message{Stopping~sTeX~processing~prematurely}
7006   \prematurestop@endsfragment
7007   \afterprematurestop
7008   \end{document}
7009 }

```

(End definition for `\prematurestop`. This function is documented on page 53.)

## 37.4 Global Variables

**\setSGvar** set a global variable

```
7010 \RequirePackage{etoolbox}
7011 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}
```

*(End definition for \setSGvar. This function is documented on page 53.)*

**\useSGvar** use a global variable

```
7012 \newrobustcmd\useSGvar[1]{%
7013   \@ifundefined{sTeX@Gvar@#1}
7014   {\PackageError{document-structure}
7015     {The sTeX Global variable #1 is undefined}
7016     {set it with \protect\setSGvar}}
7017   \@nameuse{sTeX@Gvar@#1}}
```

*(End definition for \useSGvar. This function is documented on page 53.)*

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

```
7018 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
7019   \@ifundefined{sTeX@Gvar@#1}
7020   {\PackageError{document-structure}
7021     {The sTeX Global variable #1 is undefined}
7022     {set it with \protect\setSGvar}}
7023   {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}
```

*(End definition for \ifSGvar. This function is documented on page 53.)*

## 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.

```
7024 \*cls)
7025 \@@=notesslides)
7026 \ProvidesExplClass{notesslides}{2022/02/28}{3.1.0}{notesslides Class}
7027 \RequirePackage{13keys2e}
7028
7029 \keys_define:nn{notesslides / cls}{
7030   class .str_set_x:N = \c__notesslides_class_str,
7031   notes .bool_set:N = \c__notesslides_notes_bool ,
7032   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
7033   docopt .str_set_x:N = \c__notesslides_docopt_str,
7034   unknown .code:n = {
7035     \PassOptionsToPackage{\CurrentOption}{document-structure}
7036     \PassOptionsToClass{\CurrentOption}{beamer}
7037     \PassOptionsToPackage{\CurrentOption}{notesslides}
7038     \PassOptionsToPackage{\CurrentOption}{stex}
7039   }
7040 }
7041 \ProcessKeysOptions{ notesslides / cls }
7042
7043 \str_if_empty:NF \c__notesslides_class_str {
7044   \PassOptionsToPackage{class=\c__notesslides_class_str}{document-structure}
7045 }
7046
7047 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{book}{
7048   \PassOptionsToPackage{defaulttopsect=part}{notesslides}
7049 }
7050 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{report}{
7051   \PassOptionsToPackage{defaulttopsect=part}{notesslides}
7052 }
7053
7054 \RequirePackage{stex}
```



```

7055 \stex_html_backend:T {
7056   \bool_set_true:N\c__notesslides_notes_bool
7057 }
7058
7059 \bool_if:NTF \c__notesslides_notes_bool {
7060   \PassOptionsToPackage{notes=true}{notesslides}
7061   \message{notesslides.cls:~Formatting~course~materials~in~notes~mode}
7062 }{
7063   \PassOptionsToPackage{notes=false}{notesslides}
7064   \message{notesslides.cls:~Formatting~course~materials~in~slides~mode}
7065 }
7066 </cls>

```

now we do the same for the notesslides package.

```

7067 <*package>
7068 \ProvidesExplPackage{notesslides}{2022/02/28}{3.1.0}{notesslides Package}
7069 \RequirePackage{l3keys2e}
7070
7071 \keys_define:nn{notesslides / pkg}{
7072   topsect          .str_set_x:N = \c__notesslides_topsect_str,
7073   defaulttopsect   .str_set_x:N = \c__notesslides_defaulttopsec_str,
7074   notes            .bool_set:N = \c__notesslides_notes_bool ,
7075   slides           .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
7076   sectocframes     .bool_set:N = \c__notesslides_sectocframes_bool ,
7077   frameimages      .bool_set:N = \c__notesslides_frameimages_bool ,
7078   fiboxed          .bool_set:N = \c__notesslides_fiboxed_bool ,
7079   noproblems       .bool_set:N = \c__notesslides_noproblems_bool,
7080   unknown          .code:n      = {
7081     \PassOptionsToClass{\CurrentOption}{stex}
7082     \PassOptionsToClass{\CurrentOption}{tikzinput}
7083   }
7084 }
7085 \ProcessKeysOptions{ notesslides / pkg }
7086
7087 \RequirePackage{stex}
7088 \stex_html_backend:T {
7089   \bool_set_true:N\c__notesslides_notes_bool
7090 }
7091
7092 \newif\ifnotes
7093 \bool_if:NTF \c__notesslides_notes_bool {
7094   \notesttrue
7095 }{
7096   \notesfalse
7097 }
7098

```

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

```

7099 \str_if_empty:NTF \c__notesslides_topsect_str {
7100   \str_set_eq:NN \__notesslides_topsect \c__notesslides_defaulttopsec_str
7101 }{
7102   \str_set_eq:NN \__notesslides_topsect \c__notesslides_topsect_str
7103 }
7104 \PassOptionsToPackage{topsect=\__notesslides_topsect}{document-structure}

```

```
7105 </package>
```

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

```
7106 <*cls>
7107 \bool_if:NTF \c__notesslides_notes_bool {
7108   \str_if_empty:NT \c__notesslides_class_str {
7109     \str_set:Nn \c__notesslides_class_str {article}
7110   }
7111   \exp_after:wN\LoadClass\exp_after:wN[\c__notesslides_docostr]
7112     {\c__notesslides_class_str}
7113 }{
7114   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
7115   \newcounter{Item}
7116   \newcounter{paragraph}
7117   \newcounter{subparagraph}
7118   \newcounter{Hfootnote}
7119 }
7120 \RequirePackage{document-structure}
```

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

```
7121 \RequirePackage{notesslides}
7122 </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.

```
7123 <*package>
7124 \bool_if:NT \c__notesslides_notes_bool {
7125   \RequirePackage{a4wide}
7126   \RequirePackage{marginnote}
7127   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
7128   \RequirePackage{mdframed}
7129   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
7130   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
7131 }
7132 \RequirePackage{stex-tikzinput}
7133 \RequirePackage{etoolbox}
7134 \RequirePackage{amssymb}
7135 \RequirePackage{amsmath}
7136 \RequirePackage{comment}
7137 \RequirePackage{textcomp}
7138 \RequirePackage{url}
7139 \RequirePackage{graphicx}
7140 \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>14</sup>

```

7141 \bool_if:NT \c__notesslides_notes_bool {
7142   \renewcommand\usetheme[2] [] {\usepackage[#1]{beamernotestheme#2}}
7143 }
7144
7145
7146 \NewDocumentCommand \libusetheme {O{} m} {
7147   \bool_if:NTF \c__notesslides_notes_bool {
7148     \libusepackage[#1]{beamernotestheme#2}
7149   }{
7150     \libusepackage[#1]{beamertheme#2}
7151   }
7152 }
7153
```

We define the sizes of slides in the notes. Somehow, we cannot get by with the same here.

```

7154 \newcounter{slide}
7155 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
7156 \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.

```

7157 \bool_if:NTF \c__notesslides_notes_bool {
7158   \renewenvironment{note}{\ignorespaces}{}
7159 }{
7160   \excludecomment{note}
7161 }

```

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.

```

7162 \bool_if:NT \c__notesslides_notes_bool {
7163   \newlength{\slideframewidth}
7164   \setlength{\slideframewidth}{1.5pt}

```

**frame** We first define the keys.

```

7165 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
7166   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
7167     \bool_set_true:N #1
7168   }{
7169     \bool_set_false:N #1
7170   }
7171 }
7172 \keys_define:nn{notesslides / frame}{
7173   label .str_set_x:N = \l__notesslides_frame_label_str,
7174   allowframebreaks .code:n = {
7175     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
7176   },

```

---

<sup>14</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.

```

7177 allowdisplaybreaks .code:n      = {
7178   \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
7179 },
7180 fragile .code:n      = {
7181   \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
7182 },
7183 shrink .code:n      = {
7184   \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
7185 },
7186 squeeze .code:n     = {
7187   \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
7188 },
7189 t .code:n           = {
7190   \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
7191 },
7192 unknown .code:n     = {}
7193 }
7194 \cs_new_protected:Nn \__notesslides_frame_args:n {
7195   \str_clear:N \l__notesslides_frame_label_str
7196   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
7197   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
7198   \bool_set_true:N \l__notesslides_frame_fragile_bool
7199   \bool_set_true:N \l__notesslides_frame_shrink_bool
7200   \bool_set_true:N \l__notesslides_frame_squeeze_bool
7201   \bool_set_true:N \l__notesslides_frame_t_bool
7202   \keys_set:nn { notesslides / frame }{ #1 }
7203 }

```

We define the environment, read them, and construct the slide number and label.

```

7204 \renewenvironment{frame}[1][]{
7205   \__notesslides_frame_args:n{#1}
7206   \sffamily
7207   \stepcounter{slide}
7208   \def\@currentlabel{\theslide}
7209   \str_if_empty:NF \l__notesslides_frame_label_str {
7210     \label{\l__notesslides_frame_label_str}
7211   }

```

We redefine the `itemize` environment so that it looks more like the one in `beamer`.

```

7212 \def\itemize@level{outer}
7213 \def\itemize@outer{outer}
7214 \def\itemize@inner{inner}
7215 \renewcommand\newpage{\addtocounter{framenum}{1}}
7216 %\newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
7217 \renewenvironment{itemize}{
7218   \ifx\itemize@level\itemize@outer
7219     \def\itemize@label{$\rhd$}
7220   \fi
7221   \ifx\itemize@level\itemize@inner
7222     \def\itemize@label{$\scriptstyle\rhd$}
7223   \fi
7224   \begin{list}
7225     {\itemize@label}
7226     {\setlength{\labelsep}{.3em}
7227      \setlength{\labelwidth}{.5em}

```

```

7228     \setlength{\leftmargin}{1.5em}
7229   }
7230   \edef\itemize@level{\itemize@inner}
7231 }{
7232   \end{list}
7233 }

```

We create the box with the `mdframed` environment from the `equinymous` package.

```

7234   \stex_html_backend:TF {
7235     \begin{stex_annotate_env}{frame}{}\vbox\bgroup
7236       \mdf@patchamsthm
7237   }{
7238     \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwid
7239   }
7240 }{
7241   \stex_html_backend:TF {
7242     \miko@slidelabel\egroup\end{stex_annotate_env}
7243   }{\medskip\miko@slidelabel\end{mdframed}}
7244 }

```

Now, we need to redefine the `frametitle` (we are still in course notes mode).

`\frametitle`

```

7245   \renewcommand{\frametitle}[1]{
7246     \stex_document_title:n { #1 }
7247     {\Large\bf\sf\color{blue}{#1}}\medskip
7248   }
7249 }

```

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

EdN:15

`\pause`

```

15
7250 \bool_if:NT \c__notesslides_notes_bool {
7251   \newcommand\pause{}
7252 }

```

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

`nparagraph`

```

7253 \bool_if:NTF \c__notesslides_notes_bool {
7254   \newenvironment{nparagraph}[1] []{\begin{sparagraph}[#1]}\end{sparagraph}}
7255 }{
7256   \excludecomment{nparagraph}
7257 }

```

`nfragment`

```

7258 \bool_if:NTF \c__notesslides_notes_bool {
7259   \newenvironment{nfragment}[2] []{\begin{sfragment}[#1]{#2}}\end{sfragment}}
7260 }{
7261   \excludecomment{nfragment}
7262 }

```

---

<sup>15</sup>EdNOTE: MK: fake it in notes mode for now

ndefinition

```
7263 \bool_if:NTF \c__notesslides_notes_bool {
7264   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}\end{sdefinition}}
7265 }{
7266   \excludecomment{ndefinition}
7267 }
```

nassertion

```
7268 \bool_if:NTF \c__notesslides_notes_bool {
7269   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}\end{sassertion}}
7270 }{
7271   \excludecomment{nassertion}
7272 }
```

nsproof

```
7273 \bool_if:NTF \c__notesslides_notes_bool {
7274   \newenvironment{nproof}[2] [] {\begin{sproof}[#1]{#2}}\end{sproof}}
7275 }{
7276   \excludecomment{nproof}
7277 }
```

nexample

```
7278 \bool_if:NTF \c__notesslides_notes_bool {
7279   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}\end{sexample}}
7280 }{
7281   \excludecomment{nexample}
7282 }
```

\inputref@\*skip We customize the hooks for in \inputref.

```
7283 \def\inputref@preskip{\smallskip}
7284 \def\inputref@postskip{\medskip}
```

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

**\inputref\***

```
7285 \let\orig@inputref\inputref
7286 \def\inputref{\@ifstar\ninputref\orig@inputref}
7287 \newcommand\ninputref[2] [] {
7288   \bool_if:NT \c__notesslides_notes_bool {
7289     \orig@inputref[#1]{#2}
7290   }
7291 }
```

(End definition for \inputref\*. This function is documented on page 55.)

## 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.

**\setslideologo** The default logo is the  $\text{\TeX}$  logo. Customization can be done by `\setslideologo{<logo name>}`.

```

7292 \newlength{\slideologoheight}
7293
7294 \RequirePackage{graphicx}
7295
7296 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7297 \providecommand\mhgraphics[2][]{
7298   \def\Gin@mhrepos{}\setkeys{Gin}{#1}
7299   \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}
7300 }
7301
7302
7303
7304 \bool_if:NTF \c__notesslides_notes_bool {
7305   \setlength{\slideologoheight}{.4cm}
7306 }{
7307   \setlength{\slideologoheight}{.25cm}
7308 }
7309 \ifcsname slideologo\endcsname\else
7310   \newsavebox{\slideologo}
7311   \sbox{\slideologo}{\text{\TeX}}
7312 \fi
7313 \newrobustcmd{\setslideologo}[2][]{
7314   \tl_if_empty:nTF{#1}{
7315     \sbox{\slideologo}{\includegraphics[height=\slideologoheight]{#2}}
7316   }{
7317     \sbox{\slideologo}{\mhgraphics[height=\slideologoheight,mhrepos=#1]{#2}}
7318   }
7319 }

```

(End definition for `\setslideologo`. This function is documented on page 55.)

**\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.

```

7320 \def\source{Michael Kohlhase}% customize locally
7321 \newrobustcmd{\setsource}[1]{\def\source{#1}}

```

(End definition for `\setsource`. This function is documented on page 55.)

**\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.

```

7322 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
7323 \newsavebox{\cclogo}
7324 \sbox{\cclogo}{\includegraphics[height=\slideologoheight]{stex-cc_somerights}}
7325 \newif\ifcchref\cchreffalse
7326 \AtBeginDocument{
7327   \@ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
7328 }
7329 \def\licensing{
7330   \ifcchref
7331     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}

```

```

7332 \else
7333   {\usebox{\cclogo}}
7334 \fi
7335 }
7336 \newrobustcmd{\setlicensing}[2][]{
7337   \def\@url{\#1}
7338   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{\#2}}
7339   \ifx\@url\@empty
7340     \def\licensing{\usebox{\cclogo}}
7341   \else
7342     \def\licensing{
7343       \ifcchref
7344         \href{\#1}{\usebox{\cclogo}}
7345       \else
7346         {\usebox{\cclogo}}
7347       \fi
7348     }
7349   \fi
7350 }

```

(End definition for `\setlicensing`. This function is documented on page 55.)

EdN:16

`\slidelabel` Now, we set up the slide label for the article mode.<sup>16</sup>

```

7351 \newrobustcmd\miko@slidelabel{
7352   \vbox to \slidelogoheight{
7353     \vss\hbox to \slidewidth
7354     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
7355   }
7356 }

```

(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.

```

7357 \def\Gin@mhrefpos{}
7358 \define@key{Gin}{mhrepos}{\def\Gin@mhrefpos{\#1}}
7359 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{\#1}}
7360 \newrobustcmd\frameimage[2][]{
7361   \stepcounter{slide}
7362   \bool_if:NT \c__notesslides_frameimages_bool {
7363     \def\Gin@ewidth{}\setkeys{Gin}{\#1}
7364     \bool_if:NF \c__notesslides_notes_bool { \vfill }
7365     \begin{center}
7366       \bool_if:NTF \c__notesslides_fboxed_bool {
7367         \fbox{
7368           \ifx\Gin@ewidth\@empty
7369             \ifx\Gin@mhrefpos\@empty
7370               \mhgraphics[width=\slidewidth,\#1]{\#2}
7371             \else
7372               \mhgraphics[width=\slidewidth,\#1,mhrefpos=\Gin@mhrefpos]{\#2}

```

<sup>16</sup>EdNOTE: see that we can use the themes for the slides some day. This is all fake.



```

7373         \fi
7374     \else% Gin@ewidth empty
7375         \ifx\Gin@mhrepos\@empty
7376             \mhgraphics[#1]{#2}
7377         \else
7378             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7379         \fi
7380     \fi% Gin@ewidth empty
7381 }
7382 }{
7383     \ifx\Gin@ewidth\@empty
7384     \ifx\Gin@mhrepos\@empty
7385         \mhgraphics[width=\slidewidth,#1]{#2}
7386     \else
7387         \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7388     \fi
7389     \ifx\Gin@mhrepos\@empty
7390         \mhgraphics[#1]{#2}
7391     \else
7392         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7393     \fi
7394     \fi% Gin@ewidth empty
7395 }
7396 \end{center}
7397 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
7398 \bool_if:NF \c__notesslides_notes_bool { \vfill }
7399 }
7400 } % ifmks@sty@frameimages

```

(End definition for `\frameimage`. This function is documented on page 56.)

## 38.5 Colors and Highlighting

We first specify sans serif fonts as the default.

```

7401 \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

```

7402 \AddToHook{begindocument}{
7403     \definecolor{green}{rgb}{0,.5,0}
7404     \definecolor{purple}{cmyk}{.3,1,0,.17}
7405 }

```

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`.

```

7406 % \def\STpresent#1{\textcolor{blue}{#1}}
7407 \def\defemph#1{\textcolor{magenta}{#1}}
7408 \def\symrefemph#1{\textcolor{cyan}{#1}}
7409 \def\compemph#1{\textcolor{blue}{#1}}
7410 \def\titleemph#1{\textcolor{blue}{#1}}
7411 \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.

```

7412 \pgfdeclareimage[width=.8em]{miko@small@dbend}{stex-dangerous-bend}
7413 \def\smalltextwarning{
7414   \pgfuseimage{miko@small@dbend}
7415   \xspace
7416 }
7417 \pgfdeclareimage[width=1.2em]{miko@dbend}{stex-dangerous-bend}
7418 \newrobustcmd\textwarning{
7419   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
7420   \xspace
7421 }
7422 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{stex-dangerous-bend}
7423 \newrobustcmd\bigtextwarning{
7424   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
7425   \xspace
7426 }

```

(End definition for `\textwarning`. This function is documented on page 56.)

```

7427 \newrobustcmd\putgraphicsat[3]{
7428   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
7429 }
7430 \newrobustcmd\putat[2]{
7431   \begin{picture}(0,0)\put(#1){#2}\end{picture}
7432 }

```

## 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`.

```

7433 \stex_html_backend:F {
7434   \bool_if:NT \c__notesslides_sectocframes_bool {
7435     \str_if_eq:VnTF \__notesslidesstopsect{part}{
7436       \newcounter{chapter}\counterwithin*{section}{chapter}
7437     }{
7438       \str_if_eq:VnT \__notesslidesstopsect{chapter}{
7439         \newcounter{chapter}\counterwithin*{section}{chapter}
7440       }
7441     }
7442   }
7443 }

```

`\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
7444 \def\part@prefix{}
7445 \@ifpackageloaded{document-structure}{\{
7446   \str_case:VnF \__notesslidesstopsect {
7447     {part}{

```

```

7448     \int_set:Nn \l_document_structure_section_level_int {0}
7449     \def\thesection{\arabic{chapter}.\arabic{section}}
7450     \def\part@prefix{\arabic{chapter}.}
7451   }
7452   {chapter}{
7453     \int_set:Nn \l_document_structure_section_level_int {1}
7454     \def\thesection{\arabic{chapter}.\arabic{section}}
7455     \def\part@prefix{\arabic{chapter}.}
7456   }
7457 }{
7458   \int_set:Nn \l_document_structure_section_level_int {2}
7459   \def\part@prefix{}
7460 }
7461 }
7462
7463 \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

```

7464 \renewenvironment{sfragment}[2][]{
7465   \__document_structure_sfragment_args:n { #1 }
7466   \int_incr:N \l_document_structure_section_level_int
7467   \bool_if:NT \c__notesslides_sectocframes_bool {
7468     \stepcounter{slide}
7469     \begin{frame}[noframenumbering]
7470     \vfill\Large\centering
7471     \red{
7472       \ifcase\l_document_structure_section_level_int\or
7473         \stepcounter{part}
7474         \def\__notesslideslabel{\omdoc@part@kw}~\Roman{part}}
7475       \def\currentsectionlevel{\omdoc@part@kw}
7476     \or
7477       \stepcounter{chapter}
7478       \def\__notesslideslabel{\omdoc@chapter@kw}~\arabic{chapter}}
7479       \def\currentsectionlevel{\omdoc@chapter@kw}
7480     \or
7481       \stepcounter{section}
7482       \def\__notesslideslabel{\part@prefix\arabic{section}}
7483       \def\currentsectionlevel{\omdoc@section@kw}
7484     \or
7485       \stepcounter{subsection}
7486       \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
7487       \def\currentsectionlevel{\omdoc@subsection@kw}
7488     \or
7489       \stepcounter{subsubsection}
7490       \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
7491       \def\currentsectionlevel{\omdoc@subsubsection@kw}
7492     \or
7493       \stepcounter{paragraph}
7494       \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}.\arabic{paragraph}}
7495       \def\currentsectionlevel{\omdoc@paragraph@kw}

```

```

7496         \else
7497             \def\__notesslideslabel{}
7498             \def\currentsectionlevel{\omdoc@paragraph@kw}
7499             \fi% end ifcase
7500             \__notesslideslabel%\sref@label@id\__notesslideslabel
7501             \quad #2%
7502         }%
7503         \vfill%
7504         \end{frame}%
7505     }
7506     \str_if_empty:NF \l__document_structure_sfragment_id_str {
7507         \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
7508     }
7509 }{}
7510 }

```

We set up a `beamer` template for theorems like `ams` style, but without a block environment.

```

7511 \def\inserttheorembodyfont{\normalfont}
7512 %\bool_if:NF \c__notesslides_notes_bool {
7513 % \defbeamertemplate{theorem begin}{miko}
7514 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
7515 % \ifx\inserttheoremaddition@empty\else\ (\inserttheoremaddition)\fi%
7516 % \inserttheorempunctuation\inserttheorembodyfont\xspace}
7517 % \defbeamertemplate{theorem end}{miko}{}}

```

and we set it as the default one.

```

7518 % \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.

```

7519 % \expandafter\def\csname Parent2\endcsname{}
7520 %}
7521
7522 \AddToHook{begindocument}{ % this does not work for some reasons
7523     \setbeamertemplate{theorems}[ams style]
7524 }
7525 \bool_if:NT \c__notesslides_notes_bool {
7526     \renewenvironment{columns}[1][ ]{%
7527         \par\noindent%
7528         \begin{minipage}%
7529             \slidewidth\centering\leavevmode%
7530     }{%
7531         \end{minipage}\par\noindent%
7532     }%
7533     \newsavebox\columnbox%
7534     \renewenvironment<>{column}[2][ ]{%
7535         \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
7536     }{%
7537         \end{minipage}\end{lrbox}\usebox\columnbox%
7538     }%
7539 }
7540 \bool_if:NFT \c__notesslides_noproblems_bool {
7541     \newenvironment{problems}{}{}

```

```

7542 }{
7543   \excludacomment{problems}
7544 }

```

## 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.

```

7545 \gdef\printexcursions{}
7546 \newcommand\excursionref[2]{% label, text
7547   \bool_if:NT \c__notesslides_notes_bool {
7548     \begin{sparagraph}[title=Excursion]
7549       #2 \sref[fallback=the appendix]{#1}.
7550     \end{sparagraph}
7551   }
7552 }
7553 \newcommand\activate@excursion[2][]{
7554   \gappto\printexcursions{\inputref[#1]{#2}}
7555 }
7556 \newcommand\excursion[4][]{% repos, label, path, text
7557   \bool_if:NT \c__notesslides_notes_bool {
7558     \activate@excursion[#1]{#3}\excursionref{#2}{#4}
7559   }
7560 }

```

(End definition for `\excursion`. This function is documented on page 56.)

**\excursiongroup**

```

7561 \keys_define:nn{notesslides / excursiongroup }{
7562   id          .str_set_x:N = \l__notesslides_excursion_id_str,
7563   intro       .tl_set:N   = \l__notesslides_excursion_intro_tl,
7564   mhrepos     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
7565 }
7566 \cs_new_protected:Nn \__notesslides_excursion_args:n {
7567   \tl_clear:N \l__notesslides_excursion_intro_tl
7568   \str_clear:N \l__notesslides_excursion_id_str
7569   \str_clear:N \l__notesslides_excursion_mhrepos_str
7570   \keys_set:nn {notesslides / excursiongroup }{ #1 }
7571 }
7572 \newcommand\excursiongroup[1][]{
7573   \__notesslides_excursion_args:n{ #1 }
7574   \ifdefempty\printexcursions{}% only if there are excursions
7575   {\begin{note}
7576     \begin{sfragment}[#1]{Excursions}%
7577     \ifdefempty\l__notesslides_excursion_intro_tl}{
7578       \inputref[\l__notesslides_excursion_mhrepos_str]{
7579         \l__notesslides_excursion_intro_tl
7580       }
7581     }
7582     \printexcursions%
7583     \end{sfragment}
7584   \end{note}}

```

```

7585 }
7586 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
7587 \endpackage

```

*(End definition for \excursiongroup. This function is documented on page 57.)*

## 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.

```
7588 <*package>
7589 <@@=problems>
7590 \ProvidesExplPackage{problem}{2022/02/26}{3.0.1}{Semantic Markup for Problems}
7591 \RequirePackage{l3keys2e,stex}
7592
7593 \keys_define:nn { problem / pkg }{
7594   notes      .default:n    = { true },
7595   notes      .bool_set:N   = \c__problems_notes_bool,
7596   gnotes     .default:n    = { true },
7597   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
7598   hints      .default:n    = { true },
7599   hints      .bool_set:N   = \c__problems_hints_bool,
7600   solutions  .default:n    = { true },
7601   solutions  .bool_set:N   = \c__problems_solutions_bool,
7602   pts        .default:n    = { true },
7603   pts        .bool_set:N   = \c__problems_pts_bool,
7604   min        .default:n    = { true },
7605   min        .bool_set:N   = \c__problems_min_bool,
7606   boxed      .default:n    = { true },
7607   boxed      .bool_set:N   = \c__problems_boxed_bool,
7608   unknown    .code:n       = {}
7609 }
7610 \newif\ifsolutions
7611
7612 \ProcessKeysOptions{ problem / pkg }
7613 \bool_if:NTF \c__problems_solutions_bool {
7614   \solutionstrue
7615 }{
7616   \solutionsfalse
7617 }
```

Then we make sure that the necessary packages are loaded (in the right versions).

```
7618 \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.

```
7619 \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.

```
7620 \def\prob@problem@kw{Problem}
7621 \def\prob@solution@kw{Solution}
7622 \def\prob@hint@kw{Hint}
7623 \def\prob@note@kw{Note}
7624 \def\prob@gnote@kw{Grading}
7625 \def\prob@pt@kw{pt}
7626 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
7627 \AddToHook{begindocument}{
7628   \ltx@ifpackageloaded{babel}{
7629     \makeatletter
7630     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7631     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7632       \input{problem-ngerman.ldf}
7633     }
7634     \clist_if_in:NnT \l_tmpa_clist {finnish}{
7635       \input{problem-finnish.ldf}
7636     }
7637     \clist_if_in:NnT \l_tmpa_clist {french}{
7638       \input{problem-french.ldf}
7639     }
7640     \clist_if_in:NnT \l_tmpa_clist {russian}{
7641       \input{problem-russian.ldf}
7642     }
7643     \makeatother
7644   }{}
7645 }
```

## 39.2 Problems and Solutions

We now prepare the KeyVal support for problems. The key macros just set appropriate internal macros.

```
7646 \keys_define:nn{ problem / problem }{
7647   id      .str_set_x:N = \l__problems_prob_id_str,
7648   pts     .tl_set:N    = \l__problems_prob_pts_tl,
7649   min     .tl_set:N    = \l__problems_prob_min_tl,
7650   title   .tl_set:N    = \l__problems_prob_title_tl,
7651   type    .tl_set:N    = \l__problems_prob_type_tl,
7652   imports .tl_set:N    = \l__problems_prob_imports_tl,
7653   name    .str_set_x:N = \l__problems_prob_name_str,
7654   refnum  .int_set:N   = \l__problems_prob_refnum_int
```



```

7655 }
7656 \cs_new_protected:Nn \__problems_prob_args:n {
7657   \str_clear:N \l__problems_prob_id_str
7658   \str_clear:N \l__problems_prob_name_str
7659   \tl_clear:N \l__problems_prob_pts_tl
7660   \tl_clear:N \l__problems_prob_min_tl
7661   \tl_clear:N \l__problems_prob_title_tl
7662   \tl_clear:N \l__problems_prob_type_tl
7663   \tl_clear:N \l__problems_prob_imports_tl
7664   \int_zero_new:N \l__problems_prob_refnum_int
7665   \keys_set:nn { problem / problem }{ #1 }
7666   \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
7667     \let\l__problems_prob_refnum_int\undefined
7668   }
7669 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

7670 \newcounter{problem}[section]
7671 \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.

```

7672 \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

```

7673 \newcommand\prob@number{
7674   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
7675     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
7676   }{
7677     \int_if_exist:NTF \l__problems_prob_refnum_int {
7678       \prob@label{\int_use:N \l__problems_prob_refnum_int }
7679     }{
7680       \prob@label\theproblem
7681     }
7682   }
7683 }

```

(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.

```

7684 \newcommand\prob@title[3]{%
7685   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
7686     #2 \l__problems_inclprob_title_tl #3
7687   }{
7688     \tl_if_exist:NTF \l__problems_prob_title_tl {
7689       #2 \l__problems_prob_title_tl #3
7690     }{
7691       #1

```

```

7692     }
7693   }
7694 }

```

(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.

```

7695 \def\prob@heading{
7696   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
7697   %\sref@label@id{\prob@problem@kw~\prob@number}{~}
7698 }

```

(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`

```

7699 \newenvironment{sproblem}[1][{}]{
7700   \__problems_prob_args:n{#1}%\sref@target%
7701   \@in@omtexttrue% we are in a statement (for inline definitions)
7702   \stepcounter{problem}\record@problem
7703   \def\current@section@level{\prob@problem@kw}
7704
7705   \str_if_empty:NT \l__problems_prob_name_str {
7706     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
7707     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
7708     \seq_get_left:NN \l_tmpa_seq \l__problems_prob_name_str
7709   }
7710
7711   \stex_if_do_html:T{
7712     \tl_if_empty:NF \l__problems_prob_title_tl {
7713       \exp_args:No \stex_document_title:n \l__problems_prob_title_tl
7714     }
7715   }
7716
7717   \exp_args:Nno\stex_module_setup:nn{type=problem}\l__problems_prob_name_str
7718
7719   \stex_reactivate_macro:N \STEXexport
7720   \stex_reactivate_macro:N \importmodule
7721   \stex_reactivate_macro:N \symdecl
7722   \stex_reactivate_macro:N \notation
7723   \stex_reactivate_macro:N \symdef
7724
7725   \stex_if_do_html:T{
7726     \begin{stex_annotate_env} {problem} {
7727       \l_stex_module_ns_str ? \l_stex_module_name_str
7728     }
7729
7730     \stex_annotate_invisible:nnn{header}{} {
7731       \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}

```

```

7732     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
7733     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
7734         \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
7735     }
7736 }
7737 }
7738
7739 \stex_csl_to_imports:No \importmodule \l__problems_prob_imports_tl
7740
7741
7742 \tl_if_exist:NTF \l__problems_inclprob_type_tl {
7743     \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
7744 }{
7745     \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
7746 }
7747 \str_if_exist:NTF \l__problems_inclprob_id_str {
7748     \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
7749 }{
7750     \str_set_eq:NN \sproblemid \l__problems_prob_id_str
7751 }
7752
7753
7754 \stex_if_smsmode:F {
7755     \clist_set:No \l_tmpa_clist \sproblemtype
7756     \tl_clear:N \l_tmpa_tl
7757     \clist_map_inline:Nn \l_tmpa_clist {
7758         \tl_if_exist:cT {__problems_sproblem_##1_start:}{
7759             \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_start:}}
7760         }
7761     }
7762     \tl_if_empty:NTF \l_tmpa_tl {
7763         \__problems_sproblem_start:
7764     }{
7765         \l_tmpa_tl
7766     }
7767 }
7768 \stex_ref_new_doc_target:n \sproblemid
7769 \stex_smsmode_do:
7770 }{
7771     \__stex_modules_end_module:
7772     \stex_if_smsmode:F{
7773         \clist_set:No \l_tmpa_clist \sproblemtype
7774         \tl_clear:N \l_tmpa_tl
7775         \clist_map_inline:Nn \l_tmpa_clist {
7776             \tl_if_exist:cT {__problems_sproblem_##1_end:}{
7777                 \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_end:}}
7778             }
7779         }
7780         \tl_if_empty:NTF \l_tmpa_tl {
7781             \__problems_sproblem_end:
7782         }{
7783             \l_tmpa_tl
7784         }
7785     }

```

```

7786 \stex_if_do_html:T{
7787   \end{stex_annotate_env}
7788 }
7789
7790 \smallskip
7791 }
7792
7793 \seq_put_right:Nx\g_stex_smsmode_allowedenvs_seq{\tl_to_str:n{sproblem}}
7794
7795
7796
7797 \cs_new_protected:Nn \__problems_sproblem_start: {
7798   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
7799 }
7800 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
7801
7802 \newcommand\stexpatchproblem[3][] {
7803   \str_set:Nx \l_tmpa_str{ #1 }
7804   \str_if_empty:NTF \l_tmpa_str {
7805     \tl_set:Nn \__problems_sproblem_start: { #2 }
7806     \tl_set:Nn \__problems_sproblem_end: { #3 }
7807   }{
7808     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
7809     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
7810   }
7811 }
7812
7813
7814 \bool_if:NT \c__problems_boxed_bool {
7815   \surroundwithmdframed{problem}
7816 }

```

**\record@problem** This macro records information about the problems in the \*.aux file.

```

7817 \def\record@problem{
7818   \protected@write\@auxout{}
7819   {
7820     \string\@problem{\prob@number}
7821     {
7822       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7823         \l__problems_inclprob_pts_tl
7824       }{
7825         \l__problems_prob_pts_tl
7826       }
7827     }%
7828     {
7829       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7830         \l__problems_inclprob_min_tl
7831       }{
7832         \l__problems_prob_min_tl
7833       }
7834     }
7835   }
7836 }

```

(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).

```
7837 \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.

```
7838 \keys_define:nn { problem / solution }{
7839   id          .str_set_x:N = \l__problems_solution_id_str ,
7840   for         .tl_set:N   = \l__problems_solution_for_tl ,
7841   height      .dim_set:N  = \l__problems_solution_height_dim ,
7842   creators    .clist_set:N = \l__problems_solution_creators_clist ,
7843   contributors .clist_set:N = \l__problems_solution_contributors_clist ,
7844   srccite     .tl_set:N   = \l__problems_solution_srccite_tl
7845 }
7846 \cs_new_protected:Nn \__problems_solution_args:n {
7847   \str_clear:N \l__problems_solution_id_str
7848   \tl_clear:N \l__problems_solution_for_tl
7849   \tl_clear:N \l__problems_solution_srccite_tl
7850   \clist_clear:N \l__problems_solution_creators_clist
7851   \clist_clear:N \l__problems_solution_contributors_clist
7852   \dim_zero:N \l__problems_solution_height_dim
7853   \keys_set:nn { problem / solution }{ #1 }
7854 }
```

the next step is to define a helper macro that does what is needed to start a solution.

```
7855 \newcommand\@startsolution[1][]{
7856   \__problems_solution_args:n { #1 }
7857   \@in@omtexttrue% we are in a statement.
7858   \bool_if:NF \c__problems_boxed_bool { \hrule }
7859   \smallskip\noindent
7860   {\textbf{\prob@solution@kw :}\enspace}
7861   \begin{small}
7862   \def\current@section@level{\prob@solution@kw}
7863   \ignorespacesandpars
7864 }
```

`\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.

```
7865 \box_new:N \l__problems_solution_box
7866 \newenvironment{solution}[1][]{
7867   \stex_html_backend:TF{
7868     \stex_if_do_html:T{
7869       \begin{stex_annotate_env}{solution}{}}
7870   }
7871   }{
7872     \setbox\l__problems_solution_box\vbox\bgroup
7873     \par\smallskip\hrule\smallskip
7874     \noindent\textbf{Solution:}~
7875   }
7876   }{
7877     \stex_html_backend:TF{
```

```

7878 \stex_if_do_html:T{
7879 \end{stex_annotate_env}
7880 }
7881 }{
7882 \smallskip\hrule
7883 \egroup
7884 \bool_if:NT \c__problems_solutions_bool {
7885 \box\l__problems_solution_box
7886 }
7887 }
7888 }
7889
7890 \newcommand\startsolutions{
7891 \bool_set_true:N \c__problems_solutions_bool
7892 % \specialcomment{solution}{\@startsolution}{
7893 % \bool_if:NF \c__problems_boxed_bool {
7894 % \hrule\medskip
7895 % }
7896 % \end{small}}%
7897 % }
7898 % \bool_if:NT \c__problems_boxed_bool {
7899 % \surroundwithmdframed{solution}
7900 % }
7901 }

```

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

## **\stopsolutions**

```

7902 \newcommand\stopsolutions{\bool_set_false:N \c__problems_solutions_bool}%\excludecomment{sol

```

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

so it only remains to start/stop solutions depending on what option was specified.

```

7903 \ifsolutions
7904 \startsolutions
7905 \else
7906 \stopsolutions
7907 \fi

```

## **exnote**

```

7908 \bool_if:NTF \c__problems_notes_bool {
7909 \newenvironment{exnote}[1][{}]{
7910 \par\smallskip\hrule\smallskip
7911 \noindent\textbf{\prob@note@kw :~ }\small
7912 }{
7913 \smallskip\hrule
7914 }
7915 }{
7916 \excludecomment{exnote}
7917 }

```

## **hint**

```

7918 \bool_if:NTF \c__problems_notes_bool {
7919 \newenvironment{hint}[1][{}]{
7920 \par\smallskip\hrule\smallskip

```

```

7921 \noindent\textbf{\prob@hint@kw :~ }\small
7922 }{
7923 \smallskip\hrule
7924 }
7925 \newenvironment{exhint}[1][]{
7926 \par\smallskip\hrule\smallskip
7927 \noindent\textbf{\prob@hint@kw :~ }\small
7928 }{
7929 \smallskip\hrule
7930 }
7931 }{
7932 \excludecomment{hint}
7933 \excludecomment{exhint}
7934 }

```

**gnote**

```

7935 \bool_if:NTF \c__problems_notes_bool {
7936 \newenvironment{gnote}[1][]{
7937 \par\smallskip\hrule\smallskip
7938 \noindent\textbf{\prob@gnote@kw :~ }\small
7939 }{
7940 \smallskip\hrule
7941 }
7942 }{
7943 \excludecomment{gnote}
7944 }

```

## 39.3 Multiple Choice Blocks

EdN:17

**mcb** 17

```

7945 \newenvironment{mcb}{
7946 \begin{enumerate}
7947 }{
7948 \end{enumerate}
7949 }

```

we define the keys for the mcb macro

```

7950 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
7951 \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
7952 \bool_set_true:N #1
7953 }{
7954 \bool_set_false:N #1
7955 }
7956 }
7957 \keys_define:nn { problem / mcb }{
7958 id .str_set_x:N = \l__problems_mcc_id_str ,
7959 feedback .tl_set:N = \l__problems_mcc_feedback_tl ,
7960 T .default:n = { false } ,
7961 T .bool_set:N = \l__problems_mcc_t_bool ,
7962 F .default:n = { false } ,
7963 F .bool_set:N = \l__problems_mcc_f_bool ,

```

---

<sup>17</sup>EdNOTE: MK: maybe import something better here from a dedicated MC package

```

7964 Ttext      .tl_set:N      = \l__problems_mcc_Ttext_str ,
7965 Ftext      .tl_set:N      = \l__problems_mcc_Ftext_str
7966 }
7967 \cs_new_protected:Nn \l__problems_mcc_args:n {
7968   \str_clear:N \l__problems_mcc_id_str
7969   \tl_clear:N \l__problems_mcc_feedback_tl
7970   \bool_set_false:N \l__problems_mcc_t_bool
7971   \bool_set_false:N \l__problems_mcc_f_bool
7972   \tl_clear:N \l__problems_mcc_Ttext_tl
7973   \tl_clear:N \l__problems_mcc_Ftext_tl
7974   \str_clear:N \l__problems_mcc_id_str
7975   \keys_set:nn { problem / mcc }{ #1 }
7976 }

```

**\mcc**

```

7977 \def\mccTrueText{\textbf{(true)}~}}
7978 \def\mccFalseText{\textbf{(false)}~}}
7979 \newcommand\mcc[2][]{}
7980   \l__problems_mcc_args:n{ #1 }
7981   \item[{$\Box$}] #2
7982   \ifsolutions
7983     \\\
7984     \bool_if:NT \l__problems_mcc_t_bool {
7985       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccTrueText\l__problems_mcc_Ttext_tl
7986     }
7987     \bool_if:NT \l__problems_mcc_f_bool {
7988       \tl_if_empty:NTF\l__problems_mcc_Ftext_tl\mccFalseText\l__problems_mcc_Ftext_tl
7989     }
7990     \tl_if_empty:NF \l__problems_mcc_feedback_tl {
7991       \emph{(\l__problems_mcc_feedback_tl)}
7992     }
7993   \fi
7994 } %solutions

```

(End definition for \mcc. This function is documented on page 59.)

## 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.

```

7995
7996 \keys_define:nn{ problem / inclproblem }{
7997   id      .str_set_x:N = \l__problems_inclprob_id_str,
7998   pts     .tl_set:N    = \l__problems_inclprob_pts_tl,
7999   min     .tl_set:N    = \l__problems_inclprob_min_tl,
8000   title   .tl_set:N    = \l__problems_inclprob_title_tl,
8001   refnum  .int_set:N    = \l__problems_inclprob_refnum_int,
8002   type    .tl_set:N    = \l__problems_inclprob_type_tl,
8003   mhrepos .str_set_x:N = \l__problems_inclprob_mhrepos_str
8004 }
8005 \cs_new_protected:Nn \__problems_inclprob_args:n {
8006   \str_clear:N \l__problems_prob_id_str

```



```

8007 \tl_clear:N \l__problems_inclprob_pts_tl
8008 \tl_clear:N \l__problems_inclprob_min_tl
8009 \tl_clear:N \l__problems_inclprob_title_tl
8010 \tl_clear:N \l__problems_inclprob_type_tl
8011 \int_zero_new:N \l__problems_inclprob_refnum_int
8012 \str_clear:N \l__problems_inclprob_mhrepos_str
8013 \keys_set:nn { problem / inclproblem }{ #1 }
8014 \tl_if_empty:NT \l__problems_inclprob_pts_tl {
8015   \let\l__problems_inclprob_pts_tl\undefined
8016 }
8017 \tl_if_empty:NT \l__problems_inclprob_min_tl {
8018   \let\l__problems_inclprob_min_tl\undefined
8019 }
8020 \tl_if_empty:NT \l__problems_inclprob_title_tl {
8021   \let\l__problems_inclprob_title_tl\undefined
8022 }
8023 \tl_if_empty:NT \l__problems_inclprob_type_tl {
8024   \let\l__problems_inclprob_type_tl\undefined
8025 }
8026 \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
8027   \let\l__problems_inclprob_refnum_int\undefined
8028 }
8029 }
8030
8031 \cs_new_protected:Nn \__problems_inclprob_clear: {
8032   \let\l__problems_inclprob_id_str\undefined
8033   \let\l__problems_inclprob_pts_tl\undefined
8034   \let\l__problems_inclprob_min_tl\undefined
8035   \let\l__problems_inclprob_title_tl\undefined
8036   \let\l__problems_inclprob_type_tl\undefined
8037   \let\l__problems_inclprob_refnum_int\undefined
8038   \let\l__problems_inclprob_mhrepos_str\undefined
8039 }
8040 \__problems_inclprob_clear:
8041
8042 \newcommand\includeproblem[2][ ]{
8043   \__problems_inclprob_args:n{ #1 }
8044   \exp_args:No \stex_in_repository:nn\l__problems_inclprob_mhrepos_str{
8045     \stex_html_backend:TF {
8046       \str_clear:N \l_tmpa_str
8047       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
8048         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
8049       }
8050       \stex_annotate_invisible:nnn{includeproblem}{
8051         \l_tmpa_str / #2
8052       }{}
8053     }{
8054       \begingroup
8055         \inputreftrue
8056         \tl_if_empty:nTF{ ##1 }{
8057           \input{#2}
8058         }{
8059           \input{ \c_stex_mathhub_str / ##1 / source / #2 }
8060         }

```

```

8061     \endgroup
8062   }
8063 }
8064 \__problems_inclprob_clear:
8065 }

```

(End definition for `\includeproblem`. This function is documented on page 60.)

## 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.

```

8066 \AddToHook{enddocument}{
8067   \bool_if:NT \c__problems_pts_bool {
8068     \message{Total:~\arabic{pts}~points}
8069   }
8070   \bool_if:NT \c__problems_min_bool {
8071     \message{Total:~\arabic{min}~minutes}
8072   }
8073 }

```

The margin pars are reader-visible, so we need to translate

```

8074 \def\pts#1{
8075   \bool_if:NT \c__problems_pts_bool {
8076     \marginpar{#1~\prob@pt@kw}
8077   }
8078 }
8079 \def\min#1{
8080   \bool_if:NT \c__problems_min_bool {
8081     \marginpar{#1~\prob@min@kw}
8082   }
8083 }

```

`\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.

```

8084 \newcounter{pts}
8085 \def\show@pts{
8086   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
8087     \bool_if:NT \c__problems_pts_bool {
8088       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
8089       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
8090     }
8091   }{
8092     \tl_if_exist:NT \l__problems_prob_pts_tl {
8093       \bool_if:NT \c__problems_pts_bool {
8094         \tl_if_empty:NTF \l__problems_prob_pts_tl{
8095           \tl_set:Nn \l__problems_prob_pts_tl {0}
8096         }
8097         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
8098         \addtocounter{pts}{\l__problems_prob_pts_tl}
8099       }
8100     }

```

```

8101 }
8102 }

```

(End definition for \show@pts. This function is documented on page ??.)  
and now the same for the minutes

\show@min

```

8103 \newcounter{min}
8104 \def\show@min{
8105   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
8106     \bool_if:NT \c__problems_min_bool {
8107       \marginpar{\l__problems_inclprob_pts_tl\ min}
8108       \addtocounter{min}{\l__problems_inclprob_min_tl}
8109     }
8110   }{
8111     \tl_if_exist:NT \l__problems_prob_min_tl {
8112       \bool_if:NT \c__problems_min_bool {
8113         \tl_if_empty:NT\l__problems_prob_min_tl{
8114           \tl_set:Nn \l__problems_prob_min_tl {0}
8115         }
8116         \marginpar{\l__problems_prob_min_tl\ min}
8117         \addtocounter{min}{\l__problems_prob_min_tl}
8118       }
8119     }
8120   }
8121 }
8122 \</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.

```
8123 \*package>
8124 \ProvidesExplPackage{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
8125 \RequirePackage{13keys2e}
8126
8127 \newif\iftest\testfalse
8128 \DeclareOption{test}{\testtrue}
8129 \newif\ifmultiple\multiplefalse
8130 \DeclareOption{multiple}{\multipletrue}
8131 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
8132 \ProcessOptions
```

Then we make sure that the necessary packages are loaded (in the right versions).

```
8133 \RequirePackage{keyval}[1997/11/10]
8134 \RequirePackage{problem}
```

`\hwexam@*kw` For multilinguality, we define internal macros for keywords that can be specialized in `*.ldf` files.

```
8135 \newcommand\hwexam@assignment@kw{Assignment}
8136 \newcommand\hwexam@given@kw{Given}
8137 \newcommand\hwexam@due@kw{Due}
8138 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
8139 blank~for~extra~space}
8140 \def\hwexam@minutes@kw{minutes}
8141 \newcommand\correction@probs@kw{prob.}
8142 \newcommand\correction@pts@kw{total}
8143 \newcommand\correction@reached@kw{reached}
8144 \newcommand\correction@sum@kw{Sum}
8145 \newcommand\correction@grade@kw{grade}
8146 \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

```

8147 \AddToHook{begindocument}{
8148 \ltx@ifpackageloaded{babel}{
8149 \makeatletter
8150 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
8151 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
8152 \input{hwexam-ngerman.ldf}
8153 }
8154 \clist_if_in:NnT \l_tmpa_clist {finnish}{
8155 \input{hwexam-finnish.ldf}
8156 }
8157 \clist_if_in:NnT \l_tmpa_clist {french}{
8158 \input{hwexam-french.ldf}
8159 }
8160 \clist_if_in:NnT \l_tmpa_clist {russian}{
8161 \input{hwexam-russian.ldf}
8162 }
8163 \makeatother
8164 }{}
8165 }
8166

```

## 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.

```

8167 \newcounter{assignment}
8168 %\numberproblemsin{assignment}

```

We will prepare the keyval support for the `assignment` environment.

```

8169 \keys_define:nn { hwexam / assignment } {
8170 id .str_set:N = \l_@@_assign_id_str,
8171 number .int_set:N = \l_@@_assign_number_int,
8172 title .tl_set:N = \l_@@_assign_title_tl,
8173 type .tl_set:N = \l_@@_assign_type_tl,
8174 given .tl_set:N = \l_@@_assign_given_tl,
8175 due .tl_set:N = \l_@@_assign_due_tl,
8176 loadmodules .code:n = {
8177 \bool_set_true:N \l_@@_assign_loadmodules_bool
8178 }
8179 }
8180 \cs_new_protected:Nn \_@@_assignment_args:n {
8181 \str_clear:N \l_@@_assign_id_str
8182 \int_set:Nn \l_@@_assign_number_int {-1}
8183 \tl_clear:N \l_@@_assign_title_tl
8184 \tl_clear:N \l_@@_assign_type_tl
8185 \tl_clear:N \l_@@_assign_given_tl
8186 \tl_clear:N \l_@@_assign_due_tl
8187 \bool_set_false:N \l_@@_assign_loadmodules_bool
8188 \keys_set:nn { hwexam / assignment }{ #1 }
8189 }

```

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.

```

8190 \newcommand\given@due[2]{
8191 \bool_lazy_all:nF {
8192 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
8193 { \tl_if_empty_p:V \l_@@_assign_given_tl }
8194 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
8195 { \tl_if_empty_p:V \l_@@_assign_due_tl }
8196 }{ #1 }
8197
8198 \tl_if_empty:NTF \l_@@_inclasssign_given_tl {
8199 \tl_if_empty:NF \l_@@_assign_given_tl {
8200 \hwexam@given@kw\xspace\l_@@_assign_given_tl
8201 }
8202 }{
8203 \hwexam@given@kw\xspace\l_@@_inclasssign_given_tl
8204 }
8205
8206 \bool_lazy_or:nnF {
8207 \bool_lazy_and_p:nn {
8208 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
8209 }{
8210 \tl_if_empty_p:V \l_@@_assign_due_tl
8211 }
8212 }{
8213 \bool_lazy_and_p:nn {
8214 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
8215 }{
8216 \tl_if_empty_p:V \l_@@_assign_due_tl
8217 }
8218 }{ ,~ }
8219
8220 \tl_if_empty:NTF \l_@@_inclasssign_due_tl {
8221 \tl_if_empty:NF \l_@@_assign_due_tl {
8222 \hwexam@due@kw\xspace \l_@@_assign_due_tl
8223 }
8224 }{
8225 \hwexam@due@kw\xspace \l_@@_inclasssign_due_tl
8226 }
8227
8228 \bool_lazy_all:nF {
8229 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
8230 { \tl_if_empty_p:V \l_@@_assign_given_tl }
8231 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
8232 { \tl_if_empty_p:V \l_@@_assign_due_tl }
8233 }{ #2 }
8234 }

```

`\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.

```

8235 \newcommand\assignment@title[3]{
8236 \tl_if_empty:NTF \l_@@_inclasssign_title_tl {
8237 \tl_if_empty:NTF \l_@@_assign_title_tl {
8238 #1
8239 }{
8240 #2\l_@@_assign_title_tl#3
8241 }
8242 }{
8243 #2\l_@@_inclasssign_title_tl#3
8244 }
8245 }

```

(End definition for \assignment@title. This function is documented on page ??.)

**\assignment@number** Like \assignment@title only for the number, and no around part.

```

8246 \newcommand\assignment@number{
8247 \int_compare:nNnTF \l_@@_inclasssign_number_int = {-1} {
8248 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8249 \arabic{assignment}
8250 } {
8251 \int_use:N \l_@@_assign_number_int
8252 }
8253 }{
8254 \int_use:N \l_@@_inclasssign_number_int
8255 }
8256 }

```

(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.

```

8257 \newenvironment{assignment}[1][]{
8258 \_@@_assignment_args:n { #1 }
8259 %\sref@target
8260 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8261 \global\stepcounter{assignment}
8262 }{
8263 \global\setcounter{assignment}{\int_use:N\l_@@_assign_number_int}
8264 }
8265 \setcounter{problem}{0}
8266 \renewcommand\prob@label[1]{\assignment@number.##1}
8267 \def\current@section@level{\document@hwexamtype}
8268 %\sref@label{id}{\document@hwexamtype \thesection}
8269 \begin{@assignment}
8270 }{
8271 \end{@assignment}
8272 }

```

In the multi-assignment case we just use the omdoc environment for suitable sectioning.

```

8273 \def\ass@title{
8274 {\protect\document@hwexamtype}\arabic{assignment}
8275 \assignment@title{}\;\;{}{}\; -- \given@due{}\;}
8276 }
8277 \ifmultiple
8278 \newenvironment{@assignment}{
8279 \bool_if:NTF \l_@@_assign_loadmodules_bool {
8280 \begin{sfragment}[loadmodules]{\ass@title}
8281 }{
8282 \begin{sfragment}{\ass@title}
8283 }
8284 }{
8285 \end{sfragment}
8286 }

```

for the single-page case we make a title block from the same components.

```

8287 \else
8288 \newenvironment{@assignment}{
8289 \begin{center}\bf
8290 \Large@title\strut\
8291 \document@hwexamtype\arabic{assignment}\assignment@title{}\;\;{}{}\;
8292 \large\given@due{--\;\;{}{}\;}\;
8293 \end{center}
8294 }{}
8295 \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.

```

8296 \keys_define:nn { hwexam / inclassignment } {
8297 %id .str_set_x:N = \l_@@_assign_id_str,
8298 number .int_set:N = \l_@@_inclassign_number_int,
8299 title .tl_set:N = \l_@@_inclassign_title_tl,
8300 type .tl_set:N = \l_@@_inclassign_type_tl,
8301 given .tl_set:N = \l_@@_inclassign_given_tl,
8302 due .tl_set:N = \l_@@_inclassign_due_tl,
8303 mhrepos .str_set_x:N = \l_@@_inclassign_mhrepos_str
8304 }
8305 \cs_new_protected:Nn \l_@@_inclassignment_args:n {
8306 \int_set:Nn \l_@@_inclassign_number_int {-1}
8307 \tl_clear:N \l_@@_inclassign_title_tl
8308 \tl_clear:N \l_@@_inclassign_type_tl
8309 \tl_clear:N \l_@@_inclassign_given_tl
8310 \tl_clear:N \l_@@_inclassign_due_tl
8311 \str_clear:N \l_@@_inclassign_mhrepos_str
8312 \keys_set:nn { hwexam / inclassignment }{ #1 }
8313 }
8314 \l_@@_inclassignment_args:n {}
8315
8316 \newcommand\inputassignment[2][{}]{

```



```

8317 \_@@_inclassassignment_args:n { #1 }
8318 \str_if_empty:NTF \l_@@_inclasssign_mhrepos_str {
8319 \input{#2}
8320 }{
8321 \stex_in_repository:nn{\l_@@_inclasssign_mhrepos_str}{
8322 \input{\mhpath{\l_@@_inclasssign_mhrepos_str}{#2}}
8323 }
8324 }
8325 \_@@_inclassassignment_args:n {}
8326 }
8327 \newcommand\includeassignment[2][]{
8328 \newpage
8329 \inputassignment[#1]{#2}
8330 }

```

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

## 40.4 Typesetting Exams

\quizheading

```

8331 \ExplSyntaxOff
8332 \newcommand\quizheading[1]{%
8333 \def\@tas{#1}%
8334 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
8335 \ifx\@tas\@empty\else%
8336 \noindent TA:~\@for\@I:=\@tas\do{{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
8337 \fi%
8338 }
8339 \ExplSyntaxOn

```

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

\testheading

```

8340
8341 \def\hwexamheader{\input{hwexam-default.header}}
8342
8343 \def\hwexamminutes{
8344 \tl_if_empty:NTF \testheading@duration {
8345 {\testheading@min}~\hwexam@minutes@kw
8346 }{
8347 \testheading@duration
8348 }
8349 }
8350
8351 \keys_define:nn { hwexam / testheading } {
8352 min .tl_set:N = \testheading@min,
8353 duration .tl_set:N = \testheading@duration,
8354 reqpts .tl_set:N = \testheading@reqpts,
8355 tools .tl_set:N = \testheading@tools
8356 }
8357 \cs_new_protected:Nn \_@@_testheading_args:n {
8358 \tl_clear:N \testheading@min
8359 \tl_clear:N \testheading@duration

```

```

8360 \tl_clear:N \testheading@reqpts
8361 \tl_clear:N \testheading@tools
8362 \keys_set:nn { hwexam / testheading }{ #1 }
8363 }
8364 \newenvironment{testheading}[1][]{
8365 \_@@_testheading_args:n{ #1 }
8366 \newcount\check@time\check@time=\testheading@min
8367 \advance\check@time by -\theassignment@totalmin
8368 \newif\if@bonuspoints
8369 \tl_if_empty:NTF \testheading@reqpts {
8370 \@bonuspointsfalse
8371 }{
8372 \newcount\bonus@pts
8373 \bonus@pts=\theassignment@totalpts
8374 \advance\bonus@pts by -\testheading@reqpts
8375 \edef\bonus@pts{\the\bonus@pts}
8376 \@bonuspointstrue
8377 }
8378 \edef\check@time{\the\check@time}
8379
8380 \makeatletter\hwexamheader\makeatother
8381 }{
8382 \newpage
8383 }

```

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

\testspace

```

8384 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

8385 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

8386 \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.

```

8387 <@=problems>
8388 \renewcommand\@problem[3]{
8389 \stepcounter{assignment@probs}
8390 \def\__problemspts{#2}
8391 \ifx\__problemspts\@empty\else
8392 \addtocounter{assignment@totalpts}{#2}
8393 \fi
8394 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
8395 \xdef\correction@probs{\correction@probs & #1}%
8396 \xdef\correction@pts{\correction@pts & #2}
8397 \xdef\correction@reached{\correction@reached &}

```

```

8398 }
8399 <@@=hwexam>

```

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

`\correction@table` This macro generates the correction table

```

8400 \newcounter{assignment@probs}
8401 \newcounter{assignment@totalpts}
8402 \newcounter{assignment@totalmin}
8403 \def\correction@probs{\correction@probs@kw}
8404 \def\correction@pts{\correction@pts@kw}
8405 \def\correction@reached{\correction@reached@kw}
8406 \stepcounter{assignment@probs}
8407 \newcommand\correction@table{
8408 \resizebox{\textwidth}{!}{%
8409 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
8410 &\multicolumn{\theassignment@probs}{c|}|%|
8411 {\footnotesize\correction@forgrading@kw} &\\ \hline
8412 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
8413 \correction@pts & \theassignment@totalpts & \\ \hline
8414 \correction@reached & & \[.7cm]\hline
8415 \end{tabular}}
8416 </package>

```

(End definition for \correction@table. This function is documented on page ??.)

## 40.5 Leftovers

at some point, we may want to reactivate the logos font, then we use

here we define the logos that characterize the assignment

```

\font\bierfont=../assignments/bierglas
\font\denkerfont=../assignments/denker
\font\uhrfont=../assignments/uhr
\font\warnschildfont=../assignments/achtung

\newcommand\bierglas{{\bierfont\char65}}
\newcommand\denker{{\denkerfont\char65}}
\newcommand\uhr{{\uhrfont\char65}}
\newcommand\warnschild{{\warnschildfont\char 65}}
\newcommand\hardA{\warnschild}
\newcommand\longA{\uhr}
\newcommand\thinkA{\denker}
\newcommand\discussA{\bierglas}

```

# Chapter 41

## References

- [Bus+04] Stephen Buswell et al. *The Open Math Standard, Version 2.0*. Tech. rep. The OpenMath Society, 2004. URL: <http://www.openmath.org/standard/om20>.
- [CR99] David Carlisle and Sebastian Rathz. *The graphicx package*. Part of the T<sub>E</sub>X distribution. The Comprehensive T<sub>E</sub>X Archive Network. 1999. URL: <https://www.tug.org/texlive/devsrc/Master/texmf-dist/doc/latex/graphics/graphicx.pdf>.
- [DCM03] The DCMI Usage Board. *DCMI Metadata Terms*. DCMI Recommendation. Dublin Core Metadata Initiative, 2003. URL: <http://dublincore.org/documents/dcmi-terms/>.
- [Koh06] Michael Kohlhase. *OMDoc – An open markup format for mathematical documents [Version 1.2]*. LNAI 4180. Springer Verlag, Aug. 2006. URL: <http://omdoc.org/pubs/omdoc1.2.pdf>.
- [LMH] *LMH Scripts*. URL: <https://github.com/sLaTeX/lmhtools>.
- [MMT] *MMT – Language and System for the Uniform Representation of Knowledge*. Project web site. URL: <https://uniformal.github.io/> (visited on 01/15/2019).
- [MRK18] Dennis Müller, Florian Rabe, and Michael Kohlhase. “Theories as Types”. In: *9th International Joint Conference on Automated Reasoning*. Ed. by Didier Galmiche, Stephan Schulz, and Roberto Sebastiani. Springer Verlag, 2018. URL: <https://kwarc.info/kohlhase/papers/ijcar18-records.pdf>.
- [Rab15] Florian Rabe. “The Future of Logic: Foundation-Independence”. In: *Logica Universalis* 10.1 (2015). 10.1007/s11787-015-0132-x; Winner of the Contest “The Future of Logic” at the World Congress on Universal Logic, pp. 1–20.
- [RK13] Florian Rabe and Michael Kohlhase. “A Scalable Module System”. In: *Information & Computation* 0.230 (2013), pp. 1–54. URL: <https://kwarc.info/frabe/Research/mmt.pdf>.
- [RT] *sLaTeX/RusTeX*. URL: <https://github.com/sLaTeX/RusTeX> (visited on 04/22/2022).

---

<sup>18</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).