

# The sTeX3 Package \*

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

2022-04-21

## Abstract

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

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

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

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

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

---

\*Version 3.0 (last revised 2022-04-21)

# Contents

<b>I</b>	<b>Manual</b>	<b>1</b>
<b>1</b>	<b>What is sTeX?</b>	<b>2</b>
<b>2</b>	<b>Quickstart</b>	<b>3</b>
2.1	Setup	3
2.1.1	The sTeX IDE	3
2.1.2	Manual Setup	3
2.2	A First sTeX Document	4
2.2.1	OMDoc/xhtml Conversion	7
<b>3</b>	<b>Creating sTeX Content</b>	<b>9</b>
3.1	How Knowledge is Organized in sTeX	9
3.2	sTeX Archives	10
3.2.1	The Local MathHub-Directory	10
3.2.2	The Structure of sTeX Archives	10
3.2.3	MANIFEST.MF-Files	11
3.2.4	Using Files in sTeX Archives Directly	12
3.3	Module, Symbol and Notation Declarations	13
3.3.1	The smodule-Environment	13
3.3.2	Declaring New Symbols and Notations	14
	Operator Notations	18
3.3.3	Argument Types	18
	b-Type Arguments	19
	a-Type Arguments	19
	B-Type Arguments	21
3.3.4	Type and Definiens Components	21
3.3.5	Precedences and Automated Bracketing	22
3.3.6	Variables	24
3.3.7	Variable Sequences	25
3.4	Module Inheritance and Structures	27
3.4.1	Multilinguality and Translations	27
3.4.2	Simple Inheritance and Namespaces	28
3.4.3	The mathstructure Environment	29
3.4.4	The copymodule Environment	32
3.4.5	The interpretmodule Environment	33
3.5	Primitive Symbols (The sTeX Metatheory)	34
<b>4</b>	<b>Using sTeX Symbols</b>	<b>35</b>
4.1	\symref and its variants	35
4.2	Marking Up Text and On-the-Fly Notations	36
4.3	Referencing Symbols and Statements	38
<b>5</b>	<b>sTeX Statements</b>	<b>39</b>
5.1	Definitions, Theorems, Examples, Paragraphs	39
5.2	Proofs	41
<b>6</b>	<b>Highlighting and Presentation Customizations</b>	<b>42</b>

<b>7</b>	<b>Additional Packages</b>	<b>44</b>
7.1	Modular Document Structuring . . . . .	44
7.2	Slides and Course Notes . . . . .	44
7.3	Homework, Problems and Exams . . . . .	44
<b>II</b>	<b>Documentation</b>	<b>45</b>
<b>8</b>	<b>sTeX-Basics</b>	<b>46</b>
8.1	Macros and Environments . . . . .	46
8.1.1	HTML Annotations . . . . .	46
8.1.2	Babel Languages . . . . .	47
8.1.3	Auxiliary Methods . . . . .	47
<b>9</b>	<b>sTeX-MathHub</b>	<b>48</b>
9.1	Macros and Environments . . . . .	48
9.1.1	Files, Paths, URIs . . . . .	48
9.1.2	MathHub Archives . . . . .	49
9.1.3	Using Content in Archives . . . . .	50
<b>10</b>	<b>sTeX-References</b>	<b>51</b>
10.1	Macros and Environments . . . . .	51
10.1.1	Setting Reference Targets . . . . .	51
10.1.2	Using References . . . . .	52
<b>11</b>	<b>sTeX-Modules</b>	<b>53</b>
11.1	Macros and Environments . . . . .	53
11.1.1	The <code>smodule</code> environment . . . . .	55
<b>12</b>	<b>sTeX-Module Inheritance</b>	<b>57</b>
12.1	Macros and Environments . . . . .	57
12.1.1	SMS Mode . . . . .	57
12.1.2	Imports and Inheritance . . . . .	58
<b>13</b>	<b>sTeX-Symbols</b>	<b>60</b>
13.1	Macros and Environments . . . . .	60
<b>14</b>	<b>sTeX-Terms</b>	<b>62</b>
14.1	Macros and Environments . . . . .	62
<b>15</b>	<b>sTeX-Structural Features</b>	<b>64</b>
15.1	Macros and Environments . . . . .	64
15.1.1	Structures . . . . .	64
<b>16</b>	<b>sTeX-Statements</b>	<b>65</b>
16.1	Macros and Environments . . . . .	65

<b>17</b>	<b>STeX-Proofs: Structural Markup for Proofs</b>	<b>66</b>
17.1	Introduction	68
17.2	The User Interface	69
17.2.1	Package Options	69
17.2.2	Proofs and Proof steps	69
17.2.3	Justifications	69
17.2.4	Proof Structure	71
17.2.5	Proof End Markers	71
17.2.6	Configuration of the Presentation	71
17.3	Limitations	72
<b>18</b>	<b>STeX-Metatheory</b>	<b>73</b>
18.1	Symbols	73
<b>III</b>	<b>Extensions</b>	<b>74</b>
<b>19</b>	<b>Tikzinput</b>	<b>75</b>
19.1	Macros and Environments	75
<b>20</b>	<b>document-structure: Semantic Markup for Open Mathematical Documents in L<sup>A</sup>T<sub>E</sub>X</b>	<b>76</b>
20.1	Introduction	76
20.2	The User Interface	77
20.2.1	Package and Class Options	77
20.2.2	Document Structure	77
20.2.3	Ignoring Inputs	79
20.2.4	Structure Sharing	79
20.2.5	Global Variables	79
20.2.6	Colors	80
20.3	Limitations	80
<b>21</b>	<b>NotesSlides – Slides and Course Notes</b>	<b>81</b>
21.1	Introduction	81
21.2	The User Interface	81
21.2.1	Package Options	81
21.2.2	Notes and Slides	82
21.2.3	Header and Footer Lines of the Slides	83
21.2.4	Frame Images	83
21.2.5	Colors and Highlighting	84
21.2.6	Front Matter, Titles, etc.	84
21.2.7	Excursions	84
21.2.8	Miscellaneous	85
21.3	Limitations	85

<b>22</b>	<b>problem.sty: An Infrastructure for formatting Problems</b>	<b>86</b>
22.1	Introduction . . . . .	86
22.2	The User Interface . . . . .	86
22.2.1	Package Options . . . . .	86
22.2.2	Problems and Solutions . . . . .	87
22.2.3	Multiple Choice Blocks . . . . .	88
22.2.4	Including Problems . . . . .	88
22.2.5	Reporting Metadata . . . . .	88
22.3	Limitations . . . . .	88
<b>23</b>	<b>hwexam.sty/cls: An Infrastructure for formatting Assignments and Exams</b>	<b>90</b>
23.1	Introduction . . . . .	91
23.2	The User Interface . . . . .	91
23.2.1	Package and Class Options . . . . .	91
23.2.2	Assignments . . . . .	91
23.2.3	Typesetting Exams . . . . .	91
23.2.4	Including Assignments . . . . .	92
23.3	Limitations . . . . .	92
<b>IV</b>	<b>Implementation</b>	<b>94</b>
<b>24</b>	<b>gTeX-Basics Implementation</b>	<b>95</b>
24.1	The gTeXDocument Class . . . . .	95
24.2	Preliminaries . . . . .	95
24.3	Messages and logging . . . . .	96
24.4	HTML Annotations . . . . .	97
24.5	Babel Languages . . . . .	98
24.6	Persistence . . . . .	99
24.7	Auxiliary Methods . . . . .	100
<b>25</b>	<b>gTeX-MathHub Implementation</b>	<b>102</b>
25.1	Generic Path Handling . . . . .	102
25.2	PWD and kpsewhich . . . . .	104
25.3	File Hooks and Tracking . . . . .	105
25.4	MathHub Repositories . . . . .	106
25.5	Using Content in Archives . . . . .	111
<b>26</b>	<b>gTeX-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>gTeX-Modules Implementation</b>	<b>122</b>
27.1	The smodule environment . . . . .	126
27.2	Invoking modules . . . . .	132
<b>28</b>	<b>gTeX-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>143</b>
29.1	Symbol Declarations . . . . .	143
29.2	Notations . . . . .	150
29.3	Variables . . . . .	158
<b>30</b>	<b>STeX-Terms Implementation</b>	<b>165</b>
30.1	Symbol Invocations . . . . .	165
30.2	Terms . . . . .	172
30.3	Notation Components . . . . .	176
30.4	Variables . . . . .	178
30.5	Sequences . . . . .	180
<b>31</b>	<b>STeX-Structural Features Implementation</b>	<b>181</b>
31.1	Imports with modification . . . . .	182
31.2	The feature environment . . . . .	190
31.3	Structure . . . . .	190
<b>32</b>	<b>STeX-Statements Implementation</b>	<b>200</b>
32.1	Definitions . . . . .	200
32.2	Assertions . . . . .	205
32.3	Examples . . . . .	209
32.4	Logical Paragraphs . . . . .	211
<b>33</b>	<b>The Implementation</b>	<b>217</b>
33.1	Package Options . . . . .	217
33.2	Proofs . . . . .	217
33.3	Justifications . . . . .	228
<b>34</b>	<b>STeX-Others Implementation</b>	<b>230</b>
<b>35</b>	<b>STeX-Metatheory Implementation</b>	<b>231</b>
<b>36</b>	<b>Tikzinput Implementation</b>	<b>234</b>
<b>37</b>	<b>document-structure.sty Implementation</b>	<b>237</b>
37.1	Package Options . . . . .	237
37.2	Document Structure . . . . .	238
37.3	Front and Backmatter . . . . .	242
37.4	Global Variables . . . . .	244
<b>38</b>	<b>NotesSlides – Implementation</b>	<b>245</b>
38.1	Class and Package Options . . . . .	245
38.2	Notes and Slides . . . . .	247
38.3	Header and Footer Lines . . . . .	251
38.4	Frame Images . . . . .	253
38.5	Colors and Highlighting . . . . .	254
38.6	Sectioning . . . . .	255
38.7	Excursions . . . . .	257

<b>39 The Implementation</b>	<b>259</b>
39.1 Package Options . . . . .	259
39.2 Problems and Solutions . . . . .	260
39.3 Multiple Choice Blocks . . . . .	267
39.4 Including Problems . . . . .	268
39.5 Reporting Metadata . . . . .	270
<b>40 Implementation: The hwexam Package</b>	<b>272</b>
40.1 Package Options . . . . .	272
40.2 Assignments . . . . .	273
40.3 Including Assignments . . . . .	276
40.4 Typesetting Exams . . . . .	277
40.5 Leftovers . . . . .	279

# 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 easiyl be skipped on a first read.



Boxes like this one explain how some  $\text{\texttt{STEX}}$  concept relates to the  $\text{\texttt{MMT/OMDoc}}$  system, philosophy or language.



# Chapter 1

## What is sTeX?

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

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

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

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

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

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

# Chapter 2

## Quickstart

### 2.1 Setup

#### 2.1.1 The sTeX IDE

TODO: VSCode Plugin

#### 2.1.2 Manual Setup

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

- **The sTeX-Package** available [here](#).  
sTeX is also available on CTAN and in TeXLive.
- 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](#)).

- **The Mmt System** available [here](#)<sup>1</sup>. We recommend following the setup routine documented [here](#).

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

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

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

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

---

<sup>1</sup>EdNOTE: For now, we require the sTeX-branch, requiring manually compiling the MMT sources

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

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

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

```

1 \documentclass{article}
2 \usepackage{stex,xcolor,stexthm}
3
4 \begin{document}
5 \begin{smodule}{GeometricSeries}
6   \importmodule[smglom/calculus]{series}
7   \importmodule[smglom/arithmetics]{realarith}
8
9   \symdef{geometricSeries}[name=geometric-series]{\comp{S}}
10
11   \begin{sdefinition}[for=geometricSeries]
12     The \definame{geometricSeries} is the \symname{?series}
13     \[\defeq{\geometricSeries}{\definiens{
14       \infinitesum{\svar{n}}{1}{
15         \realdivide[frac]{1}{
16           \realpower{2}{\svar{n}}
17         }
18       }}
19     \].\]
20   \end{sdefinition}
21
22   \begin{sassertion}[name=geometricSeriesConverges,type=theorem]
23     The \symname{geometricSeries} \symname{converges} towards $1$.
24   \end{sassertion}
25 \end{smodule}
26 \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.

Feel free to 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 now:

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

**smodule** First, we open a new *module* called `GeometricSeries`. This module is assigned a *globally unique* identifier (URI), which (depending on your pdf viewer) should pop up in a tooltip if you hover over the word **geometric series**.

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

---

**\importmodule** Next, we *import* two modules – `series` in the `smglom/calculus`-archive, and `realarith` in the `smglom/arithmetics`-archive. 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  $\text{\LaTeX}$  symbols and associated semantic macros (e.g. `\infinitesum`, `\realdive`, `\realpower`) in the desired module available. Additionally, they “export” these symbols to all further modules which include the *current* module – i.e. if in some future module we would put `\importmodule{GeometricSeries}`, we would also have `\infinitesum` etc. at our disposal.

---

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

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

---

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

---

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

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

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

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

The `\define{geometricSeries}` is the `\symname{?series}`

<u><code>\symname</code></u>	The <code>\symname</code> -command prints the name of a symbol, highlights it (based on customizable settings) and associates the text printed with the corresponding symbol. If you hover over the word <code>series</code> in the pdf output, you should see a tooltip showing the full URI of the symbol used.
<u><code>\symref</code></u>	The <code>\symname</code> -command is a special case of the more general <code>\symref</code> -command, which allows customizing the precise text associated with a symbol.
<u><code>\define</code></u> <u><code>\definiendum</code></u>	<p>The <code>sdefinition</code>-environment provides two additional macros, <code>\define</code> and <code>\definiendum</code> which behave similar to <code>\symname</code> and <code>\symref</code>, but explicitly mark the symbols as <i>being defined</i> in this environment, to allow for special highlighting.</p> <pre> \[\defeq{\geometricSeries}{\definiens{   \infinitesum{svar{n}}{1}{     \realdivide[frac]{1}{       \realpower{2}{svar{n}}     }   }} }\].\]</pre> <p>The next snippet – set in a math environment – uses several semantic macros imported from (or recursively via) <code>series</code> and <code>realarithmetics</code>, such as <code>\defeq</code>, <code>\infinitesum</code>, 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. <code>\realdivide[frac]{a}{b}</code> will use the explicit notation named <code>frac</code> of the semantic macro <code>\realdivide</code>, which yields <math>\frac{a}{b}</math> instead of <math>a/b</math>.</p>
<u><code>\svar</code></u>	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> .
<u><code>\definiens</code></u>	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  $\text{\LaTeX}$  yields pretty colors and tooltips<sup>1</sup>. But  $\text{\LaTeX}$  becomes a lot more powerful if we additionally convert our document to `xhtml`.

**TODO VSCode Plugin**

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

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

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

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

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

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

**Remark 2.2.2:**

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

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

## Chapter 3

# Creating sTeX Content

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

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

**lang** (*<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.

**sms** (*<boolean>*) use *persisted* mode (not yet implemented).

**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:

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

 M

 M

 T

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

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





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

- Symbol declarations induce OMDOC/MMT *constants*, with optional (formal) *type* and *definiens* components.
- Finally,  $\text{\texttt{\textit{STEX}}}$  expressions are converted to OMDOC/MMT terms, which use the syntax of `OPENMATH`.

## 3.2 $\text{\texttt{\textit{STEX}}}$ Archives

### 3.2.1 The Local MathHub-Directory

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

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

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

### 3.2.2 The Structure of $\text{\texttt{\textit{STEX}}}$ Archives

An  $\text{\texttt{\textit{STEX}}}$  archive `group/name` needs to be stored in the directory `/path/to/mathhub/group/name`; e.g. assuming your local **MathHub**-directory is set as `/user/foo/MathHub`, then in order for the `smglom/calculus`-archive to be found by the  $\text{\texttt{\textit{STEX}}}$  system, it needs to be in `/user/foo/MathHub/smgom/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 this 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 “encyclopedic” 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, instructing  $\text{\TeX}$  (and associated software) of various properties of an archive. For example, the `MANIFEST.MF` of the `smglom/calculus`-archive looks like this:

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

---

---

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

---

---

`\inputref` `\inputref`[Some/Archive]{some/file} behaves like `\mhinput`, but wraps the input in a `\begingroup ... \endgroup`. When converting to `xhtml`, the file is not input at all, and instead an `html`-annotation is inserted that references the file.

In the majority of cases `\inputref` is likely to be preferred over `\mhinput`.

---

---

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

---

---

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

---

---

`\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`.

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`.

Will throw 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.

## 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 optional arguments, all of which are optional:

`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.

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

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

#### Example 1

Input:

```

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

```

Output:

Hello World

#### \stexpatchmodule

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

For example:

#### Example 2

Input:

```

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

```

Output:

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

### 3.3.2 Declaring New Symbols and Notations

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

`\symdecl`

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

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

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

`\notation`

In that case, 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:  $b$

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

`\comp`

Unfortunately, we have no highlighting whatsoever now. That is because we need to tell  $\text{\TeX}$  explicitly which parts of the notation are *notation components* which *should* be highlighted. 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  $\text{\LaTeX}$  users), and might want to give those another thought - quite likely, the macro you aim to implement does not actually represent a semantically meaningful mathematical concept, and you will want to use `\def` and similar native  $\text{\LaTeX}$  macro definitions rather than semantic macros.

## `\symdef`

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

### Example 7

Input:

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

Output:

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

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



---

`\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}`.

### 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,
2 op={\text{a:}\cdot\text{; b:}\cdot}]
3 {\comp{\text{a:}}#1\comp{\text{; b:}}#2}
4 \symname{newbinarysymbol} is also occasionally written
5 $\newbinarysymbol![ab]$
```

Output:

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

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

### 3.3.3 Argument Types

The notations so far used *simple* arguments which we call *i-type* 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 *i-type* arguments. However, there are three more argument types which we will investigate now, namely *b-type*, *a-type* and *B-type* arguments.

## b-Type Arguments

A **b-type** 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.

$\hookrightarrow$  **b-type** arguments behave exactly like **i-type** arguments within  $\text{\TeX}$ , but applications of binding operators, i.e. symbols with **b-type** arguments, are translated to  $\text{\OMBIND}$ -terms in  $\text{\OMDOC}$ / $\text{\OMT}$ , rather than  $\text{\OMA}$ .

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

### Example 9

Input:

```
1 \symdef{summation}[args=biil]
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  $\text{\summation}$ -symbol in the expression.

## a-Type Arguments

**a-type** 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. **a-type** arguments allow us to write e.g.  $\text{\addition}\{a,b,c,d,e\}$  rather than having to write something like  $\text{\addition}\{a\}\{\text{\addition}\{b\}\{\text{\addition}\{c\}\{\text{\addition}\{d\}\{e\}\}\}\}$ !

$\text{\notation}$  (and consequently  $\text{\symdef}$ , too) take one additional argument for each **a-type** 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.  $\text{\ascendingchain}\{S\}\{a,b,c,d,e\}\{t\}$  should yield  $\forall a <_S b <_S c <_S d <_S e. t$ . The "base"-notation for this operator is simply  $\{\text{\comp}\{\text{\forallall}\} \#2 \text{\comp}\{.,\}\#3\}$ , where  $\#2$  represents the full notation fragment *accumulated* from  $\{a,b,c,d,e\}$ .

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

### 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:  $\forall a <_S b <_{Sc} <_{sd} <_{se} t$

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

### Example 11

Input:

```

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

```

Output:

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

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

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

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

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

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

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

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

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

## B-Type Arguments

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

### Example 12

Input:

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

Output:

$\forall x,y,z.P$

## 3.3.4 Type and Definiens Components

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

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

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

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

### Example 13

Input:

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

Output:

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

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

#### Example 14

Input:

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

Output:

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

### 3.3.5 Precedences and Automated Bracketing

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

#### Example 15

Input:

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

Output:

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

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

#### Example 16

Input:

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

Output:

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

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

### Example 17

Input:

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

Output:

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

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

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

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

### Example 18

Input:

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

Output:

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

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

---

`\infprec`  
`\neginfprec`

---

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

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

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

### 3.3.6 Variables

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

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

---

`\svar`

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

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

---

**\vardef**

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

### Example 19

Input:

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

Output:

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

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

**TODO: bind=forall/exists**

### 3.3.7 Variable Sequences

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

---

**\varseq**

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

This is best shown by example:

### Example 20

Input:

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

Output:

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



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

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

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

#### Example 21

Input:

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

Output:

$$a_1 + \dots + a_n$$

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

#### Example 22

Input:

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

Output:

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

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

### 3.4.1 Multilinguality and Translations

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

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

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

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

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

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

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

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

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

### 3.4.2 Simple Inheritance and Namespaces

---

`\importmodule`  
`\usemodule`

---

`\importmodule`[Some/Archive]{path?ModuleName} is only allowed within an `smodule`-environment and makes the symbols declared therein available. Additionally the content 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  $\langle top-directory \rangle / some/path/Foo[. \langle lang \rangle].tex$ , or in  $\langle top-directory \rangle / some/path[. \langle lang \rangle].tex$  (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 code from a module, you can use the command `\STEXexport{<code>}` in your module. Then `<code>` is executed (both immediately and) every time the current module is opened via `\importmodule` or `\usemodule`.



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

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

### 3.4.3 The `mathstructure` Environment

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

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

This phenomenon is important and common enough to warrant special support, in particular because it requires being able to *instantiate* such structures (or, ratherer, 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  $\text{monoid}$  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  $\text{monoid}$ .

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}{
2   universe = Int ,
3   op = addition ,
4   unit = zero
5 }{monoid}{\mathbb{Z}_{+,0}}
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: `mathstructure{<name>}` does in fact simply create 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`.  
 $\hookrightarrow M$   $\hookrightarrow M$   $\rightsquigarrow T$  `\instantiate` appropriately generates a constant whose definiens is a record term of type `Mod(<name>-structure)`, with the fields assigned appropriately based on the 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 $\varM!:=\mathstrut{\varM{universe},\varM{op}!,\varM{unit}}{\varM{op}!:\funtype{\varM{universe},\varM{universe}}{\varM{universe}}}$
5 such that
6 $\varM{op}!:\funtype{\varM{universe},\varM{universe}}{\varM{universe}}$
7 and...
8
9 \varinstantiate{varMb}{universe = Int}{monoid}{M_2}
10
11 \noindent Let $\varMb!:=\mathstrut{\varMb{universe},\varMb{op}!,\varMb{unit}}{\varMb{op}!:\funtype{\varMb{universe},\varMb{universe}}{\varMb{universe}}}$
12 a \symname{monoid} on $\Int$...

```

Output:

A **monoid** is a structure  $M := \langle U, \circ, e \rangle$  such that  $\circ : U \times U \rightarrow U$  and...  
 Let  $M_2 := \langle \mathbb{Z}, \circ, e \rangle$  a **monoid** on  $\mathbb{Z}$ ...

We will return to this 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 28

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 29

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 30

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)

TODO: metatheory documentation

## 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 31

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 32

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`, then  $\text{\TeX}$  checks all symbols currently in scope until it finds one, whose full URI ends with `string`. 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.

However, this also means that if we have symbols `foo` and e.g. `miraculous-foo`, then  $\text{\TeX}$  might resolve `\symname{foo}` to `miraculous-foo` if it finds this symbol first. It is therefore a good idea to prefix symbol names with a `?`, thus ensuring that  $\text{\TeX}$  will find the symbol `...?foo` rather than `...?miraculous-foo`.

## 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 33

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  
 $\hookrightarrow$  M  $\rightarrow$  OMA with `<OMS name="...?addition"/>` as head and `<OMV name="n"/>` and  
 $\hookrightarrow$  T  $\hookrightarrow$  `<OMV name="m"/>` as arguments.

---

**\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 34

Input:

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

Output:

Addition is...

In deed, `\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)

#### Example 35

Input:

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

Output:

adding  $k$  yields...

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

The same syntax can be used in math mode, too, which allows us to spontaneously introduce new notations on the fly. We can activate it using the starred variants of semantic macros:

### Example 36

Input:

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

Output:

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

## 4.3 Referencing Symbols and Statements

TODO: references documentation

## 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
- `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), `title=`, and `for=`.

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

#### Example 37

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`  
`\definiens`  
`\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.

`\definiens`[<optional symbolname>]{<code>} marks up <code> as being the explicit *definiens* of <optional symbolname> (in case `for=` has multiple symbols).

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

All four environments 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:

### Example 38

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
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...

Now the **mathstructure monoid** 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}{  
2   universe = Int ,  
3   op = addition ,  
4   unit = zero  
5 }{monoid}{\mathbb{Z}_{+,0}}
```

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

## 5.2 Proofs

TODO

---

<sup>2</sup>Of course, **STEX** 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**



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

Therefore we introduced the separate environments `sdefinition` etc. instead of using `definition` directly, and 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 `gTEX` 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 that 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`-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 `sdefinitions` to use a predefined `definition`-environment, we can do

```

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 `STEX` 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

```

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

TODO: tikzinput documentation

### 7.1 Modular Document Structuring

TODO: document-structure documentation

### 7.2 Slides and Course Notes

TODO: notesslides documentation

### 7.3 Homework, Problems and Exams

TODO: problem documentation

TODO: hwexam documentation

## 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> X <sub>ML</sub>
--------------------------	---

---

---

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

---

---

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

---

---

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

---

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

---

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

---

Annotates the HTML generated by `⟨content⟩` with

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

### 8.1.2 Babel Languages

---

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

---

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

### 8.1.3 Auxiliary Methods

---

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

---

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

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

---

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

---

## Chapter 9

# STEX-MathHub

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

### 9.1 Macros and Environments

---

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

---

#### 9.1.1 Files, Paths, URIs

---

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

---

---

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

---

---

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

---

---

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

---

---

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

---

---

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

---



---

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

---

### 9.1.2 MathHub Archives

---

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

---

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

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

---

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

---

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

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

---

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

---

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

---

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

---



---

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

---



### 9.1.3 Using Content in Archives

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

# Chapter 10

## STEX-References

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

### 10.1 Macros and Environments

---

---

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

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

---

---

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

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

---

---

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

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

---

---

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

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

---

---

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

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

#### 10.1.1 Setting Reference Targets

---

---

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

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

---

---

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

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

### 10.1.2 Using References

---

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

---

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

---

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

---

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

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

---

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

---

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

# Chapter 11

## STEX-Modules

This sub package contains code related to Modules

### 11.1 Macros and Environments

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

---

---

`\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** `\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.



---

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

---

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

---

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

---

## 12.1.2 Imports and Inheritance

---

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

---

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

---

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

---

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

---

 $\backslash\text{stex\_import\_module\_uri:nn}$ 


---

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

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

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

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

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

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

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

2. Otherwise:

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

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

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

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

---

 $\backslash\text{l\_stex\_import\_name\_str}$   
 $\backslash\text{l\_stex\_import\_archive\_str}$   
 $\backslash\text{l\_stex\_import\_path\_str}$   
 $\backslash\text{l\_stex\_import\_ns\_str}$ 


---

stores the result in these four variables.

---

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


---

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

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

# Chapter 13

## STEX-Symbols

Code related to symbol declarations and notations

### 13.1 Macros and Environments

---

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

---

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

- **name**: An (OMDOC) name. By default equal to  $\langle\text{macroname}\rangle$ .
- **type**: An (ideally semantic) term, representing a *type*. Not used by STEX, but passed on to MMT for semantic services.
- **def**: An (ideally semantic) term, representing a *definiens*. Not used by STEX, but passed on to MMT for semantic services.
- **local**: A boolean (by default false). If set, this declaration will not be added to the module content, i.e. importing the current module will not make this declaration available.
- **args**: Specifies the “signature” of the semantic macro. Can be either an integer  $0 \leq n \leq 9$ , or a (more precise) sequence of the following characters:

- i** a “normal” argument, e.g.  $\backslash\text{symdecl}\{\text{plus}\}[\text{args}=\text{ii}]$  allows for  $\backslash\text{plus}\{2\}\{2\}$ .
- a** an *associative* argument; i.e. a sequence of arbitrarily many arguments provided as a comma-separated list, e.g.  $\backslash\text{symdecl}\{\text{plus}\}[\text{args}=\text{a}]$  allows for  $\backslash\text{plus}\{2,2,2\}$ .
- b** a *variable* argument. Is treated by STEX like an *i*-argument, but an application is turned into an `OMBind` in OMDOC, binding the provided variable in the subsequent arguments of the operator; e.g.  $\backslash\text{symdecl}\{\text{forall}\}[\text{args}=\text{bi}]$  allows for  $\backslash\text{forall}\{x\in\text{Nat}\}\{x\geq 0\}$ .

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

# Chapter 14

## STEX-Terms

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

### 14.1 Macros and Environments

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

<hr/> <hr/>	
<code>\infprec</code> <code>\neginfprec</code>	Maximal and minimal notation precedences.
<hr/> <hr/>	
<code>\dobrackets</code>	<code>\dobrackets {⟨body⟩}</code>
	Puts $\langle body \rangle$ in parentheses; scaled if in display mode unscaled otherwise. Uses the current $\text{\SIX}$ brackets (by default ( and )), which can be changed temporarily using <code>\withbrackets</code> .
<hr/> <hr/>	
<code>\withbrackets</code>	<code>\withbrackets ⟨left⟩ ⟨right⟩ {⟨body⟩}</code>
	Temporarily (i.e. within $\langle body \rangle$ ) sets the brackets used by $\text{\SIX}$ for automated bracketing (by default ( and )) to $\langle left \rangle$ and $\langle right \rangle$ . Note that $\langle left \rangle$ and $\langle right \rangle$ need to be allowed after <code>\left</code> and <code>\right</code> in display-mode.
<hr/> <hr/>	
<code>\stex_term_custom:nn</code>	<code>\stex_term_custom:nn{⟨URI⟩}{⟨args⟩}</code>
	Implements custom one-time notation. Invoked by <code>\stex_invoke_symbol:n</code> in text mode, or if followed by <code>*</code> in math mode, or whenever followed by <code>!</code> .
<hr/> <hr/>	
<code>\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

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

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

## Contents

## 17.1 Introduction

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

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

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

Example 1: A very explicit proof, marked up semantically

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

<sup>2</sup>EdNOTE: talk a bit more about proofs and their structure,... maybe copy from OMDoc spec.

## 17.2 The User Interface

### 17.2.1 Package Options

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

### 17.2.2 Proofs and Proof steps

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

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

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

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

### 17.2.3 Justifications

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

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

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

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

1. For the induction we have to consider the following cases:
  - 1.1.  $n = 1$ : then we compute  $1 = 1^2$  □
  - 1.2.  $n = 2$ : 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. □

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

17.2.4 Proof Structure

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

17.2.5 Proof End Markers

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

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

17.2.6 Configuration of the Presentation

Finally, we provide configuration hooks in Figure 1 for the keywords in proofs. These are mainly intended for package authors building on `statements`, e.g. for multi-language support.<sup>3</sup>. The proof step labels can be customized via the `\pstlabelstyle` macro:

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

Figure 1: Configuration Hooks for Semantic Proof Markup

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

<sup>3</sup>EdNOTE: we might want to develop an extension `sproof-babel` in the future.

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

## 17.3 Limitations

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

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

## Chapter 18

# sTeX-Metatheory

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

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

### 18.1 Symbols



**Part III**  
**Extensions**

## Chapter 19

# Tikzinput

### 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

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

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

### 20.1 Introduction

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

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

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

source and the formatter does the copying during document formatting/presentation.<sup>4</sup>

## 20.2 The User Interface

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

### 20.2.1 Package and Class Options

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

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

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

### 20.2.2 Document Structure

<code>document</code>	The top-level <code>document</code> environment can be given key/value information by the
<code>\documentkeys</code>	<code>\documentkeys</code> macro in the preamble <sup>3</sup> . This can be used to give metadata about the
<code>id</code>	document. For the moment only the <code>id</code> key is used to give an identifier to the <code>omdoc</code>
<code>sfragment</code>	element resulting from the L <sup>A</sup> T <sub>E</sub> XML transformation.
	The structure of the document is given by the <code>omgroup</code> environment just like in OM-
	DOC. In the L <sup>A</sup> T <sub>E</sub> X route, the <code>omgroup</code> environment is flexibly mapped to sectioning com-
	mands, inducing the proper sectioning level from the nesting of <code>omgroup</code> environments.
	Correspondingly, the <code>omgroup</code> environment takes an optional key/value argument for
	metadata followed by a regular argument for the (section) title of the <code>omgroup</code> . The op-
<code>id</code>	tional metadata argument has the keys <code>id</code> for an identifier, <code>creators</code> and <code>contributors</code>
<code>creators</code>	for the Dublin Core metadata [DCM03]; see [Koh20a] for details of the format. The
<code>contributors</code>	<code>short</code> allows to give a short title for the generated section. If the title contains semantic
<code>short</code>	macros, they need to be protected by <code>\protect</code> , and we need to give the <code>loadmodules</code>
<code>loadmodules</code>	key it needs no value. For instance we would have

```

\begin{smodule}{foo}
\symdef{bar}{B^a_r}
...
\begin{sfragment}[id=sec.barderv,loadmodules]{Introducing $\protect\bar$ Derivation

```

<sup>4</sup>EdNOTE: integrate with latexml's XMRef in the Math mode.

<sup>3</sup>We cannot patch the document environment to accept an optional argument, since other packages we load already do; pity.

`blindfragment`

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

- The outer one groups the introductory parts of the book (which we assume to have a sectioning hierarchy topping at the part level). This `blindomgroup` makes sure that the introductory remarks become a “chapter” instead of a “part”.
- The inner one groups the frontmatter<sup>4</sup> and makes the preface of the book a section-level construct. Note that here the `display=flow` on the `omgroup` environment prevents numbering as is traditional for prefaces.

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

Example 3: A typical Document Structure of a Book

`\skipomgroup`

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

`\currentsectionlevel`  
`\CurrentSectionLevel`

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

---

<sup>4</sup>We shied away from redefining the `frontmatter` to induce a `blindomgroup`, but this may be the “right” way to go in the future.

### 20.2.3 Ignoring Inputs

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

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

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

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

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

### 20.2.4 Structure Sharing

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

`\STRcopy` The `\STRlabel` macro has a variant `\STRsemantics`, where the label argument is optional, and which takes a third argument, which is ignored in  $\text{\LaTeX}$ . This allows to specify the meaning of the content (whatever that may mean) in cases, where the source document is not formatted for presentation, but is transformed into some content markup format.<sup>5</sup>

`\STRsemantics`

### 20.2.5 Global Variables

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

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

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

<sup>5</sup>EdNOTE: document LMID und LMXRef here if we decide to keep them.

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

### 20.2.6 Colors

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

## 20.3 Limitations

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

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

# Chapter 21

## NotesSlides – Slides and Course Notes

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

### 21.1 Introduction

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

### 21.2 The User Interface

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

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

#### 21.2.1 Package Options

The `notesslides` class takes a variety of class options:<sup>6</sup>


- |   |  |
|---|--|
| <code>slides</code><br><code>notes</code> | <ul style="list-style-type: none"><li>• The options <code>slides</code> and <code>notes</code> switch between slides mode and notes mode (see Section 21.2.2).</li></ul> |
|---|--|



<code>sectocframes</code>	<ul style="list-style-type: none"> <li>If the option <code>sectocframes</code> is given, then for the <code>omgroups</code>, special frames with the <code>omgroup</code> title (and number) are generated.</li> </ul>
<code>showmeta</code>	<ul style="list-style-type: none"> <li><code>showmeta</code>. If this is set, then the metadata keys are shown (see [Koh20b] for details and customization options).</li> </ul>
<code>frameimages</code> <code>fiboxed</code>	<ul style="list-style-type: none"> <li>If the option <code>frameimages</code> is set, then slide mode also shows the <code>\frameimage</code>-generated frames (see section 21.2.4). If also the <code>fiboxed</code> option is given, the slides are surrounded by a box.</li> </ul>
<code>topsect</code>	<ul style="list-style-type: none"> <li><code>topsect=&lt;sect&gt;</code> can be used to specify the top-level sectioning level; the default for <code>&lt;sect&gt;</code> is <code>section</code>.</li> </ul>

## 21.2.2 Notes and Slides

`frame` Slides are represented with the `frame` just like in the `beamer` class, see [Tanb] for details.  
`note` The `notesslides` class adds the `note` environment for encapsulating the course note fragments.<sup>5</sup>

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

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

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

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

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

Example 4: A typical Course Notes File

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

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

<sup>6</sup>EDNOTE: leaving out `noproblems` for the moment until we decide what to do with it.

<sup>5</sup>MK: it would be very nice, if we did not need this environment, and this should be possible in principle, but not without intensive L<sup>A</sup>T<sub>E</sub>X trickery. Hints to the author are welcome.

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

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

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

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

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

`\inputref*`  
`nparagraph`  
`nfragment`  
`ndefinition`  
`nexample`  
`nsproof`  
`nassertion`

### 21.2.3 Header and Footer Lines of the Slides

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

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

`\setslidelogo`  
`\setsource`  
`\setlicensing`

### 21.2.4 Frame Images

Sometimes, we want to integrate slides as images after all – e.g. because we already have a PowerPoint presentation, to which we want to add  $\text{\TeX}$ notes. In this case we can use `\frameimage[<opt>]{<path>}`, where `<opt>` are the options of `\includegraphics` from the `graphicx` package [CR99] and `<path>` is the file path (extension can be left off like in `\includegraphics`). We have added the `label` key that allows to give a frame label that can be referenced like a regular `beamer` frame.<sup>7</sup>

`\frameimage`  
`\mhframeimage`

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

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

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

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

<sup>7</sup>EdNOTE: MK: the `hyperref` link does not seem to work yet. I wonder why but do not have the time to fix it.

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

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

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

## 21.2.5 Colors and Highlighting

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

## 21.2.6 Front Matter, Titles, etc.

### 21.2.7 Excursions

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

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

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

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

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

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

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

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

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

### 21.2.8 Miscellaneous

## 21.3 Limitations

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

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

## Chapter 22

# problem.sty: An Infrastructure for formatting Problems

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

### 22.1 Introduction

The `problem` package supplies an infrastructure that allows specify problem. Problems are text fragments that come with auxiliary functions: hints, notes, and solutions<sup>6</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.

### 22.2 The User Interface

#### 22.2.1 Package Options

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

---

<sup>6</sup>for the moment multiple choice problems are not supported, but may well be in a future version

## 22.2.2 Problems and Solutions

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

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

Example 5: A marked up Problem

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

```
Problem 22.2.1 (Fitting Elefants)
How many Elefants can you fit into a Volkswagen beetle?


---


Hint: Think positively, this is simple!


---


Note:Justify your answer


---


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


---


```

Example 6: The Formatted Problem from Figure 5

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

**exnote**

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

may arise in grading.

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

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

### 22.2.3 Multiple Choice Blocks

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

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

See Figure ?? for an example

### 22.2.4 Including Problems

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

### 22.2.5 Reporting Metadata

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

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

## 22.3 Limitations

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

1. none reported yet

```

\begin{sproblem}[title=Functions,name=functions1]
  What is the keyword to introduce a function definition in python?
  \begin{mcb}
    \mcc[T]{def}
    \mcc[F,feedback=that is for C and C++){function}
    \mcc[F,feedback=that is for Standard ML]{fun}
    \mcc[F,Ftext=Noooooooooooo,feedback=that is for Java]{public static void}
  \end{mcb}
\end{sproblem}

```

---

**Problem 22.2.2 (Functions)**

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

- ☐ def
- ☐ function
- ☐ fun
- ☐ public static void

---

**Problem 22.2.3 (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*)

Example 7: A Problem with a multiple choice block



## Chapter 23

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

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

## Contents

## 23.1 Introduction

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

## 23.2 The User Interface

### 23.2.1 Package and Class Options

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

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

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

### 23.2.2 Assignments

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

### 23.2.3 Typesetting Exams

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

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

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

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

### 23.2.4 Including Assignments

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

### 23.3 Limitations

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

1. none reported yet.

---

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

2022-04-21

Write the solutions to the sheet.

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

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

[illegible]

Example 8: A generated test heading.

Part IV

# Implementation

## Chapter 24

# sTeX -Basics Implementation

### 24.1 The sTeXDocument 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 </cls>
```

### 24.2 Preliminaries

```
23 <*package>
24
25 %%%%%%%%% basics.dtx %%%%%%%%%
26
```

```

27 \RequirePackage{expl3,l3keys2e,ltxcmds}
28 \ProvidesExplPackage{stex}{2022/03/03}{3.1.0}{sTeX package}
29
30 \bool_if_exist:NF \c_stex_document_class_bool {
31   \bool_set_false:N \c_stex_document_class_bool
32   \RequirePackage{standalone}
33 }
34
35 \message{^^J
36   *****^^J
37   *~This~is~sTeX~version~3.1.0~*^^J
38   *****^^J
39 ^^J}
40
41 %\RequirePackage{morewrites}
42 %\RequirePackage{amsmath}
43
44 Package options:
45 \keys_define:nn { stex } {
46   debug      .clist_set:N = \c_stex_debug_clist ,
47   lang       .clist_set:N = \c_stex_languages_clist ,
48   mathhub    .tl_set_x:N = \mathhub ,
49   usesms     .bool_set:N = \c_stex_persist_mode_bool ,
50   writesms   .bool_set:N = \c_stex_persist_write_mode_bool ,
51   image      .bool_set:N = \c_tikzinput_image_bool ,
52   unknown    .code:n      = {}
53 }
54 \ProcessKeysOptions { stex }

```

**\stex** The sTeX logo:

**\sTeX**

```

54 \RequirePackage{xspace}
55 \protected\def\stex{
56   \@ifundefined{texorpdfstring}{\let\texorpdfstring\@firstoftwo}{}
57   \texorpdfstring{\raisebox{-.5ex}{S\kern-.5ex\TeX}}{sTeX}\xspace
58 }
59 \let\sTeX\stex

```

(End definition for \stex and \sTeX. These functions are documented on page 46.)

## 24.3 Messages and logging

```

60 <@@=stex_log>

```

Warnings and error messages

```

61 \msg_new:nnn{stex}{error/unknownlanguage}{
62   Unknown~language:~#1
63 }
64 \msg_new:nnn{stex}{warning/nomathhub}{
65   MATHHUB~system~variable~not~found~and~no~
66   \detokenize{\mathhub}-value~set!
67 }
68 \msg_new:nnn{stex}{error/deactivated-macro}{
69   The~\detokenize{#1}~command~is~only~allowed~in~#2!
70 }

```

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

```

71 \cs_new_protected:Nn \stex_debug:nn {
72   \clist_if_in:NnTF \c_stex_debug_clist { all } {
73     \msg_set:nnn{stex}{debug / #1}{
74       \\Debug~#1:~#2\\
75     }
76     \msg_none:nn{stex}{debug / #1}
77   }{
78     \clist_if_in:NnT \c_stex_debug_clist { #1 } {
79       \msg_set:nnn{stex}{debug / #1}{
80         \\Debug~#1:~#2\\
81       }
82       \msg_none:nn{stex}{debug / #1}
83     }
84   }
85 }

```

(End definition for \stex\_debug:nn. This function is documented on page 46.)

Redirecting messages:

```

86 \clist_if_in:NnTF \c_stex_debug_clist {all} {
87   \msg_redirect_module:nnn{ stex }{ none }{ term }
88 }{
89   \clist_map_inline:Nn \c_stex_debug_clist {
90     \msg_redirect_name:nnn{ stex }{ debug / ##1 }{ term }
91   }
92 }
93
94 \stex_debug:nn{log}{debug~mode~on}

```

## 24.4 HTML Annotations

95 <@@=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**

96 \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**

```

97 \cs_new_protected:Nn \_stex_html_checkempty:n {
98   \tl_set:Nn \l_stex_html_arg_tl { #1 }
99   \tl_if_empty:NT \l_stex_html_arg_tl {
100     \tl_set_eq:NN \l_stex_html_arg_tl \c_stex_html_emptyarg_tl
101   }
102 }

```

(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**

```

103 \bool_new:N \_stex_html_do_output_bool
104 \bool_set_true:N \_stex_html_do_output_bool
105

```



```

106 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
107   \bool_if:nTF \_stex_html_do_output_bool
108     \prg_return_true: \prg_return_false:
109 }

```

(End definition for `\stex_if_do_html:TF`. This function is documented on page 46.)

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

```

110 \cs_new_protected:Nn \stex_suppress_html:n {
111   \exp_args:Nne \use:nn {
112     \bool_set_false:N \_stex_html_do_output_bool
113     #1
114   }{
115     \stex_if_do_html:T {
116       \bool_set_true:N \_stex_html_do_output_bool
117     }
118   }
119 }

```

(End definition for `\stex_suppress_html:n`. This function is documented on page 46.)

`\stex_annotate:enw`

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

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

```

120 \tl_if_exist:NF\stex@backend{
121   \ifcsname if@rustex\endcsname
122     \def\stex@backend{rustex}
123   \else
124     \ifcsname if@latexml\endcsname
125       \def\stex@backend{latexml}
126     \else
127       \def\stex@backend{pdflatex}
128     \fi
129   \fi
130 }
131 \input{stex-backend-\stex@backend.cfg}

```

(End definition for `\stex_annotate:nnn`, `\stex_annotate_invisible:n`, and `\stex_annotate_invisible:nnn`. These functions are documented on page 47.)

## 24.5 Babel Languages

```

132 <@@=stex_language>

```

`\c_stex_languages_prop`

`\c_stex_language_abbrevs_prop`

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

```

133 \prop_const_from_keyval:Nn \c_stex_languages_prop {
134   en = english ,
135   de = ngerman ,
136   ar = arabic ,
137   bg = bulgarian ,
138   ru = russian ,
139   fi = finnish ,
140   ro = romanian ,

```

```

141   tr = turkish ,
142   fr = french
143 }
144
145 \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop {
146   english   = en ,
147   ngerman   = de ,
148   arabic    = ar ,
149   bulgarian = bg ,
150   russian   = ru ,
151   finnish   = fi ,
152   romanian  = ro ,
153   turkish   = tr ,
154   french    = fr
155 }
156 % todo: chinese simplified (zhs)
157 %       chinese traditional (zht)

```

(End definition for `\c_stex_languages_prop` and `\c_stex_language_abbrevs_prop`. These variables are documented on page 47.)

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

```

158 \clist_if_empty:NF \c_stex_languages_clist {
159   \clist_clear:N \l_tmpa_clist
160   \clist_map_inline:Nn \c_stex_languages_clist {
161     \prop_get:NnNTF \c_stex_languages_prop { #1 } \l_tmpa_str {
162       \clist_put_right:No \l_tmpa_clist \l_tmpa_str
163     } {
164       \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
165     }
166   }
167   \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
168   \RequirePackage[\clist_use:Nn \l_tmpa_clist,]{babel}
169 }
170
171 \AtBeginDocument{
172   \stex_html_backend:T {
173     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
174     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
175     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
176     \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
177     \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
178       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
179       \stex_debug:nn{basics} {Language~\l_tmpa_str~
180         inferred~from~file~name}
181       \stex_annotate_invisible:nnn{language}{ \l_tmpa_str }{}
182     }
183   }
184 }

```

## 24.6 Persistence

```

185 <@@=stex_persist>
186 \bool_if:NTF \c_stex_persist_mode_bool {

```

```

187 \def \stex_persist:n #1 {}
188 \def \stex_persist:x #1 {}
189 }{
190 \bool_if:NTF \c_stex_persist_write_mode_bool {
191 \iow_new:N \c__stex_persist_iow
192 \iow_open:Nn \c__stex_persist_iow{\jobname.sms}
193 \AtEndDocument{
194 \iow_close:N \c__stex_persist_iow
195 }
196 \cs_new_protected:Nn \stex_persist:n {
197 \tl_set:Nn \l_tmpa_tl { #1 }
198 \regex_replace_all:nnN { \cP\# } { \cO\# } \l_tmpa_tl
199 \exp_args:NNo \iow_now:Nn \c__stex_persist_iow \l_tmpa_tl
200 }
201 \cs_generate_variant:Nn \stex_persist:n {x}
202 }{
203 \def \stex_persist:n #1 {}
204 \def \stex_persist:x #1 {}
205 }
206 }

```

## 24.7 Auxiliary Methods

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

```

207 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
208 \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
209 \def#1{
210 \msg_error:nnnn{stex}{error/deactivated-macro}{\detokenize{#1}}{#2}
211 }
212 }

```

(End definition for \stex\_deactivate\_macro:Nn. This function is documented on page 47.)

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

```

213 \cs_new_protected:Nn \stex_reactivate_macro:N {
214 \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname
215 }

```

(End definition for \stex\_reactivate\_macro:N. This function is documented on page 47.)

**\ignorespacesandpars**

```

216 \protected\def\ignorespacesandpars{
217 \begingroup\catcode13=10\relax
218 \@ifnextchar\par{
219 \endgroup\expandafter\ignorespacesandpars\@gobble
220 }{
221 \endgroup
222 }
223 }
224
225 \cs_new_protected:Nn \stex_copy_control_sequence:NNN {
226 \tl_set:Nx \_tmp_args_tl {\cs_argument_spec:N #2}
227 \exp_args:NNo \tl_remove_all:Nn \_tmp_args_tl \c_hash_str
228 \int_set:Nn \l_tmpa_int {\tl_count:N \_tmp_args_tl}

```

```

229
230 \tl_clear:N \_tmp_args_tl
231 \int_step_inline:nn \l_tmpa_int {
232   \tl_put_right:Nx \_tmp_args_tl {{\exp_not:n{####}\exp_not:n{##1}}}
233 }
234
235 \tl_set:Nn #3 {\cs_generate_from_arg_count:NNnn #1 \cs_set:Npn}
236 \tl_put_right:Nx #3 { {\int_use:N \l_tmpa_int}{
237   \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
238   \exp_after:wN\exp_after:wN\exp_after:wN {
239     \exp_after:wN #2 \_tmp_args_tl
240   }
241 }}
242 }
243 \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {cNN}
244 \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {NcN}
245 \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {ccN}

```

(End definition for `\ignorespacesandpars`. This function is documented on page 47.)

`\MMTrule`

```

246 \NewDocumentCommand \MMTrule {m m}{
247   \seq_set_split:Nnn \l_tmpa_seq , {#2}
248   \int_zero:N \l_tmpa_int
249   \stex_annotate_invisible:nnn{mmtrule}{scala://#1}{
250     $\seq_map_inline:Nn \l_tmpa_seq {
251       \int_incr:N \l_tmpa_int
252       \stex_annotate:nnn{arg}{i\int_use:N \l_tmpa_int}{##1}
253     }$
254   }
255 }
256
257 \NewDocumentCommand \MMTinclude {m}{
258   \stex_annotate_invisible:nnn{import}{#1}{ }
259 }
260 \endpackage

```

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

## Chapter 25

# STEX -MathHub Implementation

```
261 <*package>
262
263 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
264
265 <@@=stex_path>
266
267 Warnings and error messages
268 \msg_new:nnn{stex}{error/norepository}{
269   No~archive~#1~found~in~#2
270 }
271 \msg_new:nnn{stex}{error/notinarchive}{
272   Not~currently~in~an~archive,~but~\detokenize{#1}~
273   needs~one!
274 }
275 \msg_new:nnn{stex}{error/nofile}{
276   \detokenize{#1}~could~not~find~file~#2
277 }
278 \msg_new:nnn{stex}{error/twofiles}{
279   \detokenize{#1}~found~two~candidates~for~#2
280 }
```

### 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`

```
279 \cs_new_protected:Nn \stex_path_from_string:Nn {
280   \str_set:Nx \l_tmpa_str { #2 }
281   \str_if_empty:NTF \l_tmpa_str {
282     \seq_clear:N #1
283   }{
284     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
285     \sys_if_platform_windows:T{
286       \seq_clear:N \l_tmpa_tl
```

```

287 \seq_map_inline:Nn #1 {
288   \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
289   \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
290 }
291 \seq_set_eq:NN #1 \l_tmpa_tl
292 }
293 \stex_path_canonicalize:N #1
294 }
295 }
296

```

(End definition for `\stex_path_from_string:Nn`. This function is documented on page 48.)

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

```

297 \cs_new_protected:Nn \stex_path_to_string:NN {
298   \exp_args:Nne \str_set:Nn #2 { \seq_use:Nn #1 / }
299 }
300
301 \cs_new:Nn \stex_path_to_string:N {
302   \seq_use:Nn #1 /
303 }

```

(End definition for `\stex_path_to_string:NN` and `\stex_path_to_string:N`. These functions are documented on page 48.)

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

```

304 \str_const:Nn \c__stex_path_dot_str {.}
305 \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.

```

306 \cs_new_protected:Nn \stex_path_canonicalize:N {
307   \seq_if_empty:NF #1 {
308     \seq_clear:N \l_tmpa_seq
309     \seq_get_left:NN #1 \l_tmpa_tl
310     \str_if_empty:NT \l_tmpa_tl {
311       \seq_put_right:Nn \l_tmpa_seq {}
312     }
313     \seq_map_inline:Nn #1 {
314       \str_set:Nn \l_tmpa_tl { ##1 }
315       \str_if_eq:NNF \l_tmpa_tl \c__stex_path_dot_str {
316         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
317           \seq_if_empty:NNTF \l_tmpa_seq {
318             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
319               \c__stex_path_up_str
320             }
321           }{
322             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
323             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
324               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
325                 \c__stex_path_up_str
326               }
327             }{

```

```

328         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
329     }
330 }
331 }{
332     \str_if_empty:NF \l_tmpa_tl {
333         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
334     }
335 }
336 }
337 }
338 \seq_gset_eq:NN #1 \l_tmpa_seq
339 }
340 }

```

(End definition for `\stex_path_canonicalize:N`. This function is documented on page 48.)

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

```

341 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
342     \seq_if_empty:NTF #1 {
343         \prg_return_false:
344     }{
345         \seq_get_left:NN #1 \l_tmpa_tl
346         \sys_if_platform_windows:TF{
347             \str_if_in:NnTF \l_tmpa_tl {:}{
348                 \prg_return_true:
349             }{
350                 \prg_return_false:
351             }
352         }{
353             \str_if_empty:NTF \l_tmpa_tl {
354                 \prg_return_true:
355             }{
356                 \prg_return_false:
357             }
358         }
359     }
360 }

```

(End definition for `\stex_path_if_absolute:N $\underline{TF}$` . This function is documented on page 48.)

## 25.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

361 \str_new:N\l_stex_kpsewhich_return_str
362 \cs_new_protected:Nn \stex_kpsewhich:n {
363     \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
364     \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
365     \tl_trim_spaces:N \l_stex_kpsewhich_return_str
366 }

```

(End definition for `\stex_kpsewhich:n`. This function is documented on page 48.)

We determine the PWD

`\c_stex_pwd_seq`  
`\c_stex_pwd_str`

```

367 \sys_if_platform_windows:TF{
368   \begingroup\escapechar=-1\catcode'\=12
369   \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
370   \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
371   \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_
372   }}{
373   \stex_kpsewhich:n{-var-value~PWD}
374   }
375
376 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
377 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
378 \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 48.)

## 25.3 File Hooks and Tracking

```

379 <@@=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

```

380 \seq_gclear_new:N\g__stex_files_stack

```

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`  
`\c_stex_mainfile_str`

```

381 \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
382 \stex_path_from_string:Nn \c_stex_mainfile_seq
383   \c_stex_mainfile_str

```

(End definition for `\c_stex_mainfile_seq` and `\c_stex_mainfile_str`. These variables are documented on page 48.)

`\g_stex_currentfile_seq`

```

384 \seq_gclear_new:N\g_stex_currentfile_seq

```

(End definition for `\g_stex_currentfile_seq`. This variable is documented on page 49.)

`\stex_filestack_push:n`

```

385 \cs_new_protected:Nn \stex_filestack_push:n {
386   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
387   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
388     \stex_path_from_string:Nn\g_stex_currentfile_seq{
389       \c_stex_pwd_str/#1
390     }
391   }
392   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
393   \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
394 }

```



(End definition for `\stex_filestack_push:n`. This function is documented on page 49.)

`\stex_filestack_pop:`

```

395 \cs_new_protected:Nn \stex_filestack_pop: {
396   \seq_if_empty:NF\g__stex_files_stack{
397     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
398   }
399   \seq_if_empty:NTF\g__stex_files_stack{
400     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
401   }{
402     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
403     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
404   }
405 }
```

(End definition for `\stex_filestack_pop:`. This function is documented on page 49.)

Hooks for the current file:

```

406 \AddToHook{file/before}{
407   \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
408 }
409 \AddToHook{file/after}{
410   \stex_filestack_pop:
411 }
```

## 25.4 MathHub Repositories

412 `<@@=stex_mathhub>`

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

```

413 \str_if_empty:NTF\mathhub{
414   \sys_if_platform_windows:TF{
415     \begingroup\escapechar=-1\catcode'\=12
416     \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str MATHHUB\c_percent_str}
417     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
418     \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_kpsewhich_return_str}}
419   }{
420     \stex_kpsewhich:n{-var-value-MATHHUB}
421   }
422   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
423 }
424 \str_if_empty:NT \c_stex_mathhub_str {
425   \sys_if_platform_windows:TF{
426     \begingroup\escapechar=-1\catcode'\=12
427     \exp_args:Nx\stex_kpsewhich:n{-var-value-HOME}
428     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
429     \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_kpsewhich_return_str}}
430   }{
431     \stex_kpsewhich:n{-var-value-HOME}
432   }
433   \ior_open:NnT \l_tmpa_ior{\l_stex_kpsewhich_return_str / .stex / mathhub.path}{
434     \begingroup\escapechar=-1\catcode'\=12
435     \ior_str_get:NN \l_tmpa_ior \l_tmpa_str
```

```

436     \sys_if_platform_windows:T{
437         \exp_args:NNx\str_replace_all:Nnn\l_tmpa_str{\c_backslash_str}/
438     }
439     \str_gset_eq:NN \c_stex_mathhub_str\l_tmpa_str
440     \endgroup
441     \ior_close:N \l_tmpa_ior
442 }
443 }
444 \str_if_empty:NTF\c_stex_mathhub_str{
445     \msg_warning:nn{stex}{warning/nomathhub}
446 }{
447     \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
448     \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
449 }
450 }{
451     \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
452     \stex_path_if_absolute:NF \c_stex_mathhub_seq {
453         \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
454             \c_stex_pwd_str/\mathhub
455         }
456     }
457     \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
458     \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
459 }

```

(End definition for `\mathhub`, `\c_stex_mathhub_seq`, and `\c_stex_mathhub_str`. These variables are documented on page 49.)

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

```

460 \cs_new_protected:Nn \_stex_mathhub_do_manifest:n {
461     \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
462         \str_set:Nx \l_tmpa_str { #1 }
463         \prop_new:c { c_stex_mathhub_#1_manifest_prop }
464         \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
465         \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
466         \_stex_mathhub_find_manifest:N \l_tmpa_seq
467         \seq_if_empty:NTF \l_stex_mathhub_manifest_file_seq {
468             \msg_error:nnxx{stex}{error/norepository}{#1}{
469                 \stex_path_to_string:N \c_stex_mathhub_str
470             }
471         } {
472             \exp_args:No \_stex_mathhub_parse_manifest:n { \l_tmpa_str }
473         }
474     }
475 }

```

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

`\l_stex_mathhub_manifest_file_seq`

```

476 \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`:

```

477 \cs_new_protected:Nn \_stex_mathhub_find_manifest:N {
478   \seq_set_eq:NN \l_tmpa_seq #1
479   \bool_set_true:N \l_tmpa_bool
480   \bool_while_do:Nn \l_tmpa_bool {
481     \seq_if_empty:NTF \l_tmpa_seq {
482       \bool_set_false:N \l_tmpa_bool
483     }{
484       \file_if_exist:nTF{
485         \stex_path_to_string:N \l_tmpa_seq/MANIFEST.MF
486       }{
487         \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
488         \bool_set_false:N \l_tmpa_bool
489       }{
490         \file_if_exist:nTF{
491           \stex_path_to_string:N \l_tmpa_seq/META-INF/MANIFEST.MF
492         }{
493           \seq_put_right:Nn \l_tmpa_seq{META-INF}
494           \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
495           \bool_set_false:N \l_tmpa_bool
496         }{
497           \file_if_exist:nTF{
498             \stex_path_to_string:N \l_tmpa_seq/meta-inf/MANIFEST.MF
499           }{
500             \seq_put_right:Nn \l_tmpa_seq{meta-inf}
501             \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
502             \bool_set_false:N \l_tmpa_bool
503           }{
504             \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
505           }
506         }
507       }
508     }
509   }
510   \seq_set_eq:NN \l__stex_mathhub_manifest_file_seq \l_tmpa_seq
511 }

```

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

`\c_stex_mathhub_manifest_ior` File variable used for MANIFEST-files

```

512 \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:

```

513 \cs_new_protected:Nn \_stex_mathhub_parse_manifest:n {
514   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
515   \ior_open:Nn \c_stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
516   \ior_map_inline:Nn \c_stex_mathhub_manifest_ior {
517     \str_set:Nn \l_tmpa_str {##1}
518     \exp_args:NNoo \seq_set_split:Nnn
519       \l_tmpb_seq \c_colon_str \l_tmpa_str
520     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {

```

```

521 \exp_args:NNe \str_set:Nn \l_tmpb_tl {
522 \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
523 }
524 \exp_args:No \str_case:nnTF \l_tmpa_tl {
525 {id} {
526 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
527 { id } \l_tmpb_tl
528 }
529 {narration-base} {
530 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
531 { narr } \l_tmpb_tl
532 }
533 {url-base} {
534 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
535 { docurl } \l_tmpb_tl
536 }
537 {source-base} {
538 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
539 { ns } \l_tmpb_tl
540 }
541 {ns} {
542 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
543 { ns } \l_tmpb_tl
544 }
545 {dependencies} {
546 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
547 { deps } \l_tmpb_tl
548 }
549 }{}{}
550 }{}
551 }
552 \ior_close:N \c__stex_mathhub_manifest_ior
553 \stex_persist:x {
554 \prop_set_from_keyval:cn{ c_stex_mathhub_#1_manifest_prop }{
555 \exp_after:wN \prop_to_keyval:N \csname c_stex_mathhub_#1_manifest_prop\endcsname
556 }
557 }
558 }

```

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

`\stex_set_current_repository:n`

```

559 \cs_new_protected:Nn \stex_set_current_repository:n {
560 \stex_require_repository:n { #1 }
561 \prop_set_eq:Nc \l_stex_current_repository_prop {
562 c_stex_mathhub_#1_manifest_prop
563 }
564 }

```

(End definition for `\stex_set_current_repository:n`. This function is documented on page 49.)

`\stex_require_repository:n`

```

565 \cs_new_protected:Nn \stex_require_repository:n {
566 \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
567 \stex_debug:nn{mathhub}{Opening~archive:~#1}

```

```

568   \_stex_mathhub_do_manifest:n { #1 }
569   }
570 }

```

(End definition for `\stex_require_repository:n`. This function is documented on page 49.)

`\l_stex_current_repository_prop` Current MathHub repository

```

571 %\prop_new:N \l_stex_current_repository_prop
572 \bool_if:NF \c_stex_persist_mode_bool {
573   \_stex_mathhub_find_manifest:N \c_stex_pwd_seq
574   \seq_if_empty:NTF \l_stex_mathhub_manifest_file_seq {
575     \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
576   } {
577     \_stex_mathhub_parse_manifest:n { main }
578     \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
579     \l_tmpa_str
580     \prop_set_eq:cN { c_stex_mathhub\_l_tmpa_str_manifest_prop }
581     \c_stex_mathhub_main_manifest_prop
582     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
583     \stex_debug:nn{mathhub}{Current~repository:~
584     \prop_item:Nn \l_stex_current_repository_prop {id}
585   }
586 }
587 }

```

(End definition for `\l_stex_current_repository_prop`. This variable is documented on page 49.)

`\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.

```

588 \cs_new_protected:Nn \stex_in_repository:nn {
589   \str_set:Nx \l_tmpa_str { #1 }
590   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
591   \str_if_empty:NTF \l_tmpa_str {
592     \prop_if_exist:NTF \l_stex_current_repository_prop {
593       \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
594       \exp_args:Ne \l_tmpa_cs{
595         \prop_item:Nn \l_stex_current_repository_prop { id }
596       }
597     }{
598       \l_tmpa_cs{}
599     }
600   }{
601     \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}
602     \stex_require_repository:n \l_tmpa_str
603     \str_set:Nx \l_tmpa_str { #1 }
604     \exp_args:Nne \use:nn {
605       \stex_set_current_repository:n \l_tmpa_str
606       \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
607     }{
608       \stex_debug:nn{mathhub}{switching~back~to:~
609       \prop_if_exist:NTF \l_stex_current_repository_prop {
610         \prop_item:Nn \l_stex_current_repository_prop { id }::~
611       \meaning\l_stex_current_repository_prop
612     }{

```

```

613         no~repository
614     }
615 }
616 \prop_if_exist:NTF \l_stex_current_repository_prop {
617     \stex_set_current_repository:n {
618         \prop_item:Nn \l_stex_current_repository_prop { id }
619     }
620 }{
621     \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
622 }
623 }
624 }
625 }

```

(End definition for `\stex_in_repository:nn`. This function is documented on page 49.)

## 25.5 Using Content in Archives

`\mhpath`

```

626 \def \mhpath #1 #2 {
627     \exp_args:Ne \tl_if_empty:nTF{#1}{
628         \c_stex_mathhub_str /
629         \prop_item:Nn \l_stex_current_repository_prop { id }
630         / source / #2
631     }{
632         \c_stex_mathhub_str / #1 / source / #2
633     }
634 }

```

(End definition for `\mhpath`. This function is documented on page 50.)

`\inputref`

`\mhinput`

```

635 \newif \ifinputref \inputreffalse
636
637 \cs_new_protected:Nn \__stex_mathhub_mhinput:nn {
638     \stex_in_repository:nn {#1} {
639         \ifinputref
640             \input{ \c_stex_mathhub_str / ##1 / source / #2 }
641         \else
642             \inputreftrue
643             \input{ \c_stex_mathhub_str / ##1 / source / #2 }
644             \inputreffalse
645         \fi
646     }
647 }
648 \NewDocumentCommand \mhinput { 0{} m }{
649     \__stex_mathhub_mhinput:nn{ #1 }{ #2 }
650 }
651
652 \cs_new_protected:Nn \__stex_mathhub_inputref:nn {
653     \stex_in_repository:nn {#1} {
654         \stex_html_backend:TF {
655             \str_clear:N \l_tmpa_str

```

```

656     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
657       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
658     }
659     \stex_annotate_invisible:nnn{inputref}{
660       \l_tmpa_str / #2
661     }{}
662   }{
663     \begingroup
664     \inputreftrue
665     \tl_if_empty:nTF{ ##1 }{
666       \input{#2}
667     }{
668       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
669     }
670     \endgroup
671   }
672 }
673 }
674 \NewDocumentCommand \inputref { 0{ } m }{
675   \__stex_mathhub_inputref:nn{ #1 }{ #2 }
676 }

```

(End definition for `\inputref` and `\mhinput`. These functions are documented on page 50.)

#### `\addmhbibresource`

```

677 \cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {
678   \stex_in_repository:nn {#1} {
679     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
680   }
681 }
682 \newcommand\addmhbibresource[2][]{
683   \__stex_mathhub_mhbibresource:nn{ #1 }{ #2 }
684 }

```

(End definition for `\addmhbibresource`. This function is documented on page 50.)

#### `\libinput`

```

685 \cs_new_protected:Npn \libinput #1 {
686   \prop_if_exist:NF \l_stex_current_repository_prop {
687     \msg_error:nnn{stex}{error/notinarchive}\libinput
688   }
689   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
690     \msg_error:nnn{stex}{error/notinarchive}\libinput
691   }
692   \seq_clear:N \l__stex_mathhub_libinput_files_seq
693   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
694   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
695
696   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
697     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}
698     \IfFileExists{ \l_tmpa_str }{
699       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
700     }{}
701     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
702     \seq_put_right:No \l_tmpa_seq \l_tmpa_str

```

```

703 }
704
705 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
706 \IfFileExists{ \l_tmpa_str }{
707   \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
708 }{}
709
710 \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
711   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
712 }{
713   \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
714     \input{ ##1 }
715   }
716 }
717 }

```

(End definition for `\libinput`. This function is documented on page 50.)

## `\libusepackage`

```

718 \NewDocumentCommand \libusepackage {0{ } m} {
719   \prop_if_exist:NF \l_stex_current_repository_prop {
720     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
721   }
722   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
723     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
724   }
725   \seq_clear:N \l__stex_mathhub_libinput_files_seq
726   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
727   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
728
729   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
730     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2}
731     \IfFileExists{ \l_tmpa_str.sty }{
732       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
733     }{}
734     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
735     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
736   }
737
738   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2}
739   \IfFileExists{ \l_tmpa_str.sty }{
740     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
741   }{}
742
743   \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
744     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
745   }{
746     \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
747       \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
748         \usepackage[##1]{ ##1 }
749       }
750     }{
751       \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
752     }

```



```

753 }
754 }

```

(End definition for `\libusepackage`. This function is documented on page 50.)

```

\mhgraphics
\cmhgraphics

```

```

755
756 \AddToHook{begindocument}{
757 \ltx@ifpackageloaded{graphicx}{
758   \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
759   \newcommand\mhgraphics[2][]{\%
760     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
761     \includegraphics[#1]{\mhp@th\Gin@mhrepos{#2}}}
762   \newcommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
763 }{}

```

(End definition for `\mhgraphics` and `\cmhgraphics`. These functions are documented on page 50.)

```

\lstinputmhlisting
\clstinputmhlisting

```

```

764 \ltx@ifpackageloaded{listings}{
765   \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
766   \newcommand\lstinputmhlisting[2][]{\%
767     \def\lst@mhrepos{}\setkeys{lst}{#1}%
768     \lstinputlisting[#1]{\mhp@th\lst@mhrepos{#2}}}
769   \newcommand\clstinputmhlisting[2][]{\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
770 }{}
771 }
772
773 \</package>

```

(End definition for `\lstinputmhlisting` and `\clstinputmhlisting`. These functions are documented on page 50.)

## Chapter 26

# STEX -References Implementation

```
774 <*package>
775
776 %%%%%%%%%% references.dtx %%%%%%%%%%
777
778 <@@=stex_refs>
779
780 Warnings and error messages
```

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

```
780 %\iow_new:N \c__stex_refs_refs_iow
781 \AtBeginDocument{
782 % \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
783 }
784 \AtEndDocument{
785 % \iow_close:N \c__stex_refs_refs_iow
786 }
```

`\STEXreftitle`

```
787 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
788
789 \NewDocumentCommand \STEXreftitle { m } {
790 \tl_gset:Nx \g__stex_refs_title_tl { #1 }
791 }
```

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

### 26.1 Document URIs and URLs

`\l_stex_current_docns_str`

```
792 \str_new:N \l_stex_current_docns_str
```

*(End definition for `\l_stex_current_docns_str`. This variable is documented on page 51.)*

`\stex_get_document_uri:`

```
793 \cs_new_protected:Nn \stex_get_document_uri: {
794   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
795   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
796   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
797   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
798   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
799
800   \str_clear:N \l_tmpa_str
801   \prop_if_exist:NT \l_stex_current_repository_prop {
802     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
803       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
804     }
805   }
806
807   \str_if_empty:NTF \l_tmpa_str {
808     \str_set:Nx \l_stex_current_docns_str {
809       file:/\stex_path_to_string:N \l_tmpa_seq
810     }
811   }{
812     \bool_set_true:N \l_tmpa_bool
813     \bool_while_do:Nn \l_tmpa_bool {
814       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
815       \exp_args:No \str_case:nnTF { \l_tmpb_str } {
816         {source} { \bool_set_false:N \l_tmpa_bool }
817       }{}{
818         \seq_if_empty:NT \l_tmpa_seq {
819           \bool_set_false:N \l_tmpa_bool
820         }
821       }
822     }
823
824     \seq_if_empty:NTF \l_tmpa_seq {
825       \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
826     }{
827       \str_set:Nx \l_stex_current_docns_str {
828         \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
829       }
830     }
831   }
832 }
```

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

`\l_stex_current_docurl_str`

```
833 \str_new:N \l_stex_current_docurl_str
```

(End definition for `\l_stex_current_docurl_str`. This variable is documented on page 51.)

`\stex_get_document_url:`

```
834 \cs_new_protected:Nn \stex_get_document_url: {
835   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
836   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
837   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
```

```

838 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
839 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
840
841 \str_clear:N \l_tmpa_str
842 \prop_if_exist:NT \l_stex_current_repository_prop {
843   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
844     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
845       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
846     }
847   }
848 }
849
850 \str_if_empty:NTF \l_tmpa_str {
851   \str_set:Nx \l_stex_current_docurl_str {
852     file:/\stex_path_to_string:N \l_tmpa_seq
853   }
854 }{
855   \bool_set_true:N \l_tmpa_bool
856   \bool_while_do:Nn \l_tmpa_bool {
857     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
858     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
859       {source} { \bool_set_false:N \l_tmpa_bool }
860     }{}{
861       \seq_if_empty:NT \l_tmpa_seq {
862         \bool_set_false:N \l_tmpa_bool
863       }
864     }
865   }
866
867   \seq_if_empty:NTF \l_tmpa_seq {
868     \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
869   }{
870     \str_set:Nx \l_stex_current_docurl_str {
871       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
872     }
873   }
874 }
875 }

```

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

## 26.2 Setting Reference Targets

```

876 \str_const:Nn \c__stex_refs_url_str{URL}
877 \str_const:Nn \c__stex_refs_ref_str{REF}
878 \str_new:N \l__stex_refs_curr_label_str
879 % @currentlabel -> number
880 % @currentlabelname -> title
881 % @currentHref -> name.number <- id of some kind
882 % \theH# -> \arabic{section}
883 % \the# -> number
884 % \hyper@makecurrent{#}
885 \int_new:N \l__stex_refs_unnamed_counter_int

```

`\stex_ref_new_doc_target:n`

```

886 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
887   \stex_get_document_uri:
888   \str_clear:N \l__stex_refs_curr_label_str
889   \str_set:Nx \l_tmpa_str { #1 }
890   \str_if_empty:NT \l_tmpa_str {
891     \int_incr:N \l__stex_refs_unnamed_counter_int
892     \str_set:Nx \l_tmpa_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
893   }
894   \str_set:Nx \l__stex_refs_curr_label_str {
895     \l_stex_current_docns_str?\l_tmpa_str
896   }
897   \seq_if_exist:cF{g__stex_refs_labels_\l_tmpa_str_seq}{
898     \seq_new:c {g__stex_refs_labels_\l_tmpa_str_seq}
899   }
900   \seq_if_in:coF{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str {
901     \seq_gput_right:co{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str
902   }
903   \stex_if_smsmode:TF {
904     \stex_get_document_url:
905     \str_gset_eq:cN {sref_url_\l__stex_refs_curr_label_str_str}\l_stex_current_docurl_str
906     \str_gset_eq:cN {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_url_str
907   }{
908     %\iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter\unexpanded\expandafter{
909     \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
910     \immediate\write\@auxout{\stexauxadddocref{\l_stex_current_docns_str}{\l_tmpa_str}}
911     \str_gset:cx {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_ref_str
912   }
913 }

```

(End definition for `\stex_ref_new_doc_target:n`. This function is documented on page 51.)

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

```

914 \cs_new_protected:Npn \stexauxadddocref #1 #2 {
915   \str_set:Nn \l_tmpa_str {#1?#2}
916   \str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
917   \seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
918     \seq_new:c {g__stex_refs_labels_#2_seq}
919   }
920   \seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
921     \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
922   }
923 }

```

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

```

924 \AtEndDocument{
925   \def\stexauxadddocref#1 #2 {}{}
926 }

```

`\stex_ref_new_sym_target:n`

```

927 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
928   \stex_if_smsmode:TF {
929     \str_if_exist:cF{sref_sym_#1_type}{
930       \stex_get_document_url:
931       \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str

```

```

932     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
933   }
934 }{
935   \str_if_empty:NF \l__stex_refs_curr_label_str {
936     \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
937     \immediate\write\@auxout{
938       \exp_not:N\expandafter\def\exp_not:N\csname \exp_not:N\detokenize{sref_sym_#1_label_
939         \l__stex_refs_curr_label_str
940       }
941     }
942   }
943 }
944 }

```

(End definition for `\stex_ref_new_sym_target:n`. This function is documented on page 51.)

## 26.3 Using References

```

945 \str_new:N \l__stex_refs_indocument_str

```

**\sref** Optional arguments:

```

946
947 \keys_define:nn { stex / sref } {
948   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
949   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
950   pre           .tl_set:N = \l__stex_refs_pre_tl ,
951   post          .tl_set:N = \l__stex_refs_post_tl ,
952 }
953 \cs_new_protected:Nn \__stex_refs_args:n {
954   \tl_clear:N \l__stex_refs_linktext_tl
955   \tl_clear:N \l__stex_refs_fallback_tl
956   \tl_clear:N \l__stex_refs_pre_tl
957   \tl_clear:N \l__stex_refs_post_tl
958   \str_clear:N \l__stex_refs_repo_str
959   \keys_set:nn { stex / sref } { #1 }
960 }

```

The actual macro:

```

961 \NewDocumentCommand \sref { 0{} m}{
962   \__stex_refs_args:n { #1 }
963   \str_if_empty:NTF \l__stex_refs_indocument_str {
964     \str_set:Nx \l_tmpa_str { #2 }
965     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
966     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
967       \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
968         \seq_get_left:cNF {g__stex_refs_labels_\l_tmpa_str _seq} \l_tmpa_str {
969           \str_clear:N \l_tmpa_str
970         }
971       }{
972         \str_clear:N \l_tmpa_str
973       }
974     }{
975       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
976       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str

```

```

977 \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
978 \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str_seq}{
979   \str_set_eq:NN \l_tmpc_str \l_tmpa_str
980   \str_clear:N \l_tmpa_str
981   \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str_seq} {
982     \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
983       \str_range:nnn { ##1 }{ -\l_tmpa_int}{ -1 }
984     }{
985       \seq_map_break:n {
986         \str_set:Nn \l_tmpa_str { ##1 }
987       }
988     }
989   }
990 }{
991   \str_clear:N \l_tmpa_str
992 }
993 }
994 \str_if_empty:NTF \l_tmpa_str {
995   \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_lin
996 }{
997   \str_if_eq:cNTF {sref_\l_tmpa_str_type} \c__stex_refs_ref_str {
998     \tl_if_empty:NTF \l__stex_refs_linktext_tl {
999       \cs_if_exist:cTF{autoref}{
1000         \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1001       }{
1002         \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1003       }
1004     }{
1005       \ltx@ifpackageloaded{hyperref}{
1006         \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
1007       }{
1008         \l__stex_refs_linktext_tl
1009       }
1010     }
1011   }{
1012     \ltx@ifpackageloaded{hyperref}{
1013       \href{\use:c{sref_url_\l_tmpa_str_str}}{\tl_if_empty:NTF \l__stex_refs_linktext_t
1014     }{
1015       \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs
1016     }
1017   }
1018 }
1019 }{
1020   % TODO
1021 }
1022 }

```

(End definition for `\sref`. This function is documented on page 52.)

## `\srefsym`

```

1023 \NewDocumentCommand \srefsym { 0{} m}{
1024   \stex_get_symbol:n { #2 }
1025   \__stex_refs_sym_aux:nn{##1}{\l_stex_get_symbol_uri_str}
1026 }

```

```

1027
1028 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1029   \str_if_exist:cTF {sref_sym_#2 _label_str }{
1030     \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1031   }{
1032     \__stex_refs_args:n { #1 }
1033     \str_if_empty:NTF \l__stex_refs_indocument_str {
1034       \tl_if_exist:cTF{sref_sym_#2 _type}{
1035         % doc uri in \l_tmpb_str
1036         \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1037         \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1038           % reference
1039           \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1040             \cs_if_exist:cTF{autoref}{
1041               \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1042             }{
1043               \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1044             }
1045           }{
1046             \ltx@ifpackageloaded{hyperref}{
1047               \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1048             }{
1049               \l__stex_refs_linktext_tl
1050             }
1051           }
1052         }{
1053           % URL
1054           \ltx@ifpackageloaded{hyperref}{
1055             \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl \
1056           }{
1057             \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_re
1058           }
1059         }
1060       }{
1061         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_l
1062       }
1063     }{
1064       % TODO
1065     }
1066   }
1067 }

```

(End definition for \srefsym. This function is documented on page 52.)

**\srefsymuri**

```

1068 \cs_new_protected:Npn \srefsymuri #1 #2 {
1069   \__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1070 }

```

(End definition for \srefsymuri. This function is documented on page 52.)

```

1071 </package>

```



## Chapter 27

# STEX -Modules Implementation

```
1072 <*package>
1073
1074 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1075
1076 <@@=stex_modules>
    Warnings and error messages
1077 \msg_new:nnn{stex}{error/unknownmodule}{
1078   No~module~#1~found
1079 }
1080 \msg_new:nnn{stex}{error/syntax}{
1081   Syntax~error:~#1
1082 }
1083 \msg_new:nnn{stex}{error/siglanguage}{
1084   Module~#1~declares~signature~#2,~but~does~not~
1085   declare~its~language
1086 }
1087 \msg_new:nnn{stex}{warning/deprecated}{
1088   #1~is~deprecated;~please~use~#2~instead!
1089 }
1090
1091 \msg_new:nnn{stex}{error/conflictingmodules}{
1092   Conflicting~imports~for~module~#1
1093 }
\l_stex_current_module_str The current module:
1094 \str_new:N \l_stex_current_module_str
    (End definition for \l_stex_current_module_str. This variable is documented on page 54.)
\l_stex_all_modules_seq Stores all available modules
1095 \seq_new:N \l_stex_all_modules_seq
    (End definition for \l_stex_all_modules_seq. This variable is documented on page 54.)
```

```

\stex_if_in_module_p:
\stex_if_in_module:TF
1096 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1097   \str_if_empty:NTF \l_stex_current_module_str
1098   \prg_return_false: \prg_return_true:
1099 }

(End definition for \stex_if_in_module:TF. This function is documented on page 54.)

```

```

\stex_if_module_exists_p:n
\stex_if_module_exists:nTF
1100 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1101   \prop_if_exist:cTF { c_stex_module_#1_prop }
1102   \prg_return_true: \prg_return_false:
1103 }

(End definition for \stex_if_module_exists:nTF. This function is documented on page 54.)

```

```

\stex_add_to_current_module:n
\STEXexport
1104 \cs_new_protected:Nn \stex_execute_in_module:n { \stex_if_in_module:T {
1105   \stex_add_to_current_module:n { #1 }
1106   \stex_do_up_to_module:n { #1 }
1107 }}
1108 \cs_generate_variant:Nn \stex_execute_in_module:n {x}
1109
1110 \cs_new_protected:Nn \stex_add_to_current_module:n {
1111   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1112 }
1113 \cs_generate_variant:Nn \stex_add_to_current_module:n {x}
1114 \cs_new_protected:Npn \STEXexport {
1115   \beginngroup
1116   \newlinechar=-1\relax
1117   \endlinechar=-1\relax
1118   %\catcode'\ = 9\relax
1119   \expandafter\endgroup\__stex_modules_export:n
1120 }
1121 \cs_new_protected:Nn \__stex_modules_export:n {
1122   \ignorespaces #1
1123   \stex_add_to_current_module:n { \ignorespaces #1 }
1124   \stex_smsmode_do:
1125 }
1126 \stex_deactivate_macro:Nn \STEXexport {module~environments}

(End definition for \stex_add_to_current_module:n and \STEXexport. These functions are documented
on page 54.)

```

```

\stex_add_constant_to_current_module:n
1127 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1128   \str_set:Nx \l_tmpa_str { #1 }
1129   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1130 }

(End definition for \stex_add_constant_to_current_module:n. This function is documented on page
54.)

```

`\stex_add_import_to_current_module:n`

```

1131 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1132   \str_set:Nx \l_tmpa_str { #1 }
1133   \exp_args:Nno
1134   \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1135     \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1136   }
1137 }

```

(End definition for `\stex_add_import_to_current_module:n`. This function is documented on page 54.)

`\stex_collect_imports:n`

```

1138 \cs_new_protected:Nn \stex_collect_imports:n {
1139   \seq_clear:N \l_stex_collect_imports_seq
1140   \__stex_modules_collect_imports:n {#1}
1141 }
1142 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1143   \seq_map_inline:cn {c_stex_module_#1_imports} {
1144     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1145       \__stex_modules_collect_imports:n { ##1 }
1146     }
1147   }
1148   \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1149     \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1150   }
1151 }

```

(End definition for `\stex_collect_imports:n`. This function is documented on page 54.)

`\stex_do_up_to_module:n`

```

1152 \int_new:N \l__stex_modules_group_depth_int
1153 \cs_new_protected:Nn \stex_do_up_to_module:n {
1154   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1155     #1
1156   }{
1157     #1
1158     \expandafter \tl_gset:Nn
1159     \csname l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1160     \expandafter\expandafter\expandafter\endcsname
1161     \expandafter\expandafter\expandafter { \csname
1162       l__stex_modules_aftergroup_\l_stex_current_module_str _tl\endcsname #1 }
1163     \aftergroup\__stex_modules_aftergroup_do:
1164   }
1165 }
1166 \cs_generate_variant:Nn \stex_do_up_to_module:n {x}
1167 \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1168   \stex_debug:nn{aftergroup}{\cs_meaning:c{
1169     l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1170   }}
1171   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1172     \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1173     \tl_gclear:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1174   }{
1175     \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}

```

```

1176     \aftergroup\__stex_modules_aftergroup_do:
1177   }
1178 }
1179 \cs_new_protected:Nn \stex_reset_up_to_module:n {
1180   \expandafter\let\csname l__stex_modules_aftergroup_#1_tl\endcsname\undefined
1181 }

```

(End definition for `\stex_do_up_to_module:n`. This function is documented on page 54.)

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

```

1182

```

(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.

```

1183 \str_new:N \l_stex_module_ns_str
1184 \str_new:N \l_stex_module_subpath_str
1185 \cs_new_protected:Nn \__stex_modules_compute_namespace:nN {
1186   \seq_set_eq:NN \l_tmpa_seq #2
1187   % split off file extension
1188   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str % <- filename
1189   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1190   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str % <- filename without suffixes
1191   \seq_put_right:No \l_tmpa_seq \l_tmpb_str % <- file path including name without suffixes
1192
1193   \bool_set_true:N \l_tmpa_bool
1194   \bool_while_do:Nn \l_tmpa_bool {
1195     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1196     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1197       {source} { \bool_set_false:N \l_tmpa_bool }
1198     }{}{
1199       \seq_if_empty:NT \l_tmpa_seq {
1200         \bool_set_false:N \l_tmpa_bool
1201       }
1202     }
1203   }
1204
1205   \stex_path_to_string:NN \l_tmpa_seq \l_stex_module_subpath_str
1206   % \l_tmpa_seq <- sub-path relative to archive
1207   \str_if_empty:NTF \l_stex_module_subpath_str {
1208     \str_set:Nx \l_stex_module_ns_str {#1}
1209   }{
1210     \str_set:Nx \l_stex_module_ns_str {
1211       #1/\l_stex_module_subpath_str
1212     }
1213   }
1214 }
1215
1216 \cs_new_protected:Nn \stex_modules_current_namespace: {
1217   \str_clear:N \l_stex_module_subpath_str
1218   \prop_if_exist:NTF \l_stex_current_repository_prop {
1219     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str

```

```

1220     \__stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1221   }{
1222     % split off file extension
1223     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1224     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1225     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1226     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1227     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1228     \str_set:Nx \l_stex_module_ns_str {
1229       file:/\stex_path_to_string:N \l_tmpa_seq
1230     }
1231   }
1232 }

```

(End definition for `\stex_modules_current_namespace:.` This function is documented on page 55.)

## 27.1 The smodule environment

smodule arguments:

```

1233 \keys_define:nn { stex / module } {
1234   title      .tl_set:N      = \smodulename ,
1235   type       .str_set_x:N   = \smodulename ,
1236   id         .str_set_x:N   = \smoduleid ,
1237   deprecate  .str_set_x:N   = \l_stex_module_deprecate_str ,
1238   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1239   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1240   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1241   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1242   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1243   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1244   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1245 }
1246
1247 \cs_new_protected:Nn \__stex_modules_args:n {
1248   \str_clear:N \smodulename
1249   \str_clear:N \smodulename
1250   \str_clear:N \smoduleid
1251   \str_clear:N \l_stex_module_ns_str
1252   \str_clear:N \l_stex_module_deprecate_str
1253   \str_clear:N \l_stex_module_lang_str
1254   \str_clear:N \l_stex_module_sig_str
1255   \str_clear:N \l_stex_module_creators_str
1256   \str_clear:N \l_stex_module_contributors_str
1257   \str_clear:N \l_stex_module_meta_str
1258   \str_clear:N \l_stex_module_srccite_str
1259   \keys_set:nn { stex / module } { #1 }
1260 }
1261
1262 % module parameters here? In the body?
1263

```

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

```

1264 \cs_new_protected:Nn \stex_module_setup:nn {

```

```

1265 \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1266 \str_set:Nx \l_stex_module_name_str { #2 }
1267 \__stex_modules_args:n { #1 }

```

First, we set up the name and namespace of the module.  
Are we in a nested module?

```

1268 \stex_if_in_module:TF {
1269   % Nested module
1270   \prop_get:cnN {c_stex_module\l_stex_current_module_str _prop}
1271   { ns } \l_stex_module_ns_str
1272   \str_set:Nx \l_stex_module_name_str {
1273     \prop_item:cn {c_stex_module\l_stex_current_module_str _prop}
1274     { name } / \l_stex_module_name_str
1275   }
1276   \str_if_empty:NT \l_stex_module_lang_str {
1277     \str_set:Nx \l_stex_module_lang_str {
1278       \prop_item:cn {c_stex_module\l_stex_current_module_str _prop}
1279       { lang }
1280     }
1281   }
1282 }{
1283   % not nested:
1284   \str_if_empty:NT \l_stex_module_ns_str {
1285     \stex_modules_current_namespace:
1286     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1287       / {\l_stex_module_ns_str}
1288     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1289     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1290       \str_set:Nx \l_stex_module_ns_str {
1291         \stex_path_to_string:N \l_tmpa_seq
1292       }
1293     }
1294   }
1295 }

```

Next, we determine the language of the module:

```

1296 \str_if_empty:NT \l_stex_module_lang_str {
1297   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1298   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1299   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1300   \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
1301     \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
1302       \exp_args:NNNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
1303     }
1304   }
1305   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1306   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
1307     \seq_pop_right:NN \l_tmpa_seq \l_stex_module_lang_str
1308     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1309       inferred~from~file~name}
1310   }
1311 }
1312
1313 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {

```

```

1314 \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1315 \l_tmpa_str {
1316   \ltx@ifpackageloaded{babel}{
1317     \exp_args:Nx \selectlanguage { \l_tmpa_str }
1318   }{}
1319 } {
1320   \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
1321 }
1322 }}

```

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

```

1323 \str_if_empty:NTF \l_stex_module_sig_str {
1324   \exp_args:Nnx \prop_gset_from_keyval:cn {
1325     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1326   } {
1327     name      = \l_stex_module_name_str ,
1328     ns        = \l_stex_module_ns_str ,
1329     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1330     lang      = \l_stex_module_lang_str ,
1331     sig       = \l_stex_module_sig_str ,
1332     deprecate = \l_stex_module_deprecate_str ,
1333     meta      = \l_stex_module_meta_str
1334   }
1335   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1336   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1337   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _copymodules}
1338   \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1339   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1340 \str_if_empty:NT \l_stex_module_meta_str {
1341   \str_set:Nx \l_stex_module_meta_str {
1342     \c_stex_metatheory_ns_str ? Metatheory
1343   }
1344 }
1345 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1346   \bool_set_true:N \l_stex_in_meta_bool
1347   \exp_args:Nx \stex_add_to_current_module:n {
1348     \bool_set_true:N \l_stex_in_meta_bool
1349     \stex_activate_module:n {\l_stex_module_meta_str}
1350     \bool_set_false:N \l_stex_in_meta_bool
1351   }
1352   \stex_activate_module:n {\l_stex_module_meta_str}
1353   \bool_set_false:N \l_stex_in_meta_bool
1354 }
1355 }{
1356   \str_if_empty:NT \l_stex_module_lang_str {
1357     \msg_error:nnxx{stex}{error/siglanguage}{
1358       \l_stex_module_ns_str?\l_stex_module_name_str
1359     }\l_stex_module_sig_str}
1360   }
1361   \stex_debug:nn{modules}{Signature~\l_stex_module_sig_str~for~\l_stex_module_ns_str?\l_st
1362   \stex_if_module_exists:nTF{\l_stex_module_ns_str?\l_stex_module_name_str}{

```

```

1363     \stex_debug:nn{modules}{(already exists)}
1364   }{
1365     \stex_debug:nn{modules}{(needs loading)}
1366     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1367     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1368     \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1369     \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1370     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1371     \str_set:Nx \l_tmpa_str {
1372       \stex_path_to_string:N \l_tmpa_seq /
1373       \l_tmpa_str . \l_stex_module_sig_str .tex
1374     }
1375     \IfFileExists \l_tmpa_str {
1376       \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1377         \str_clear:N \l_stex_current_module_str
1378         \seq_clear:N \l_stex_all_modules_seq
1379         \stex_debug:nn{modules}{Loading~signature}
1380       }
1381     }{
1382       \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1383     }
1384   }
1385   \stex_if_smsmode:F {
1386     \stex_activate_module:n {
1387       \l_stex_module_ns_str ? \l_stex_module_name_str
1388     }
1389   }
1390   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1391 }
1392 \str_if_empty:NF \l_stex_module_deprecate_str {
1393   \msg_warning:nnxx{stex}{warning/deprecated}{
1394     Module~\l_stex_current_module_str
1395   }{
1396     \l_stex_module_deprecate_str
1397   }
1398 }
1399 \seq_put_right:Nx \l_stex_all_modules_seq {
1400   \l_stex_module_ns_str ? \l_stex_module_name_str
1401 }
1402 \tl_clear:c{l__stex_modules_aftergroup\l_stex_module_ns_str ? \l_stex_module_name_str _tl}
1403 }

```

(End definition for `\stex_module_setup:nn`. This function is documented on page 55.)

**smodule** The module environment.

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

```

1404 \cs_new_protected:Nn \_stex_modules_begin_module: {
1405   \stex_reactivate_macro:N \STEXexport
1406   \stex_reactivate_macro:N \importmodule
1407   \stex_reactivate_macro:N \symdecl
1408   \stex_reactivate_macro:N \notation
1409   \stex_reactivate_macro:N \symdef
1410 }

```



```

1411 \stex_debug:nn{modules}{
1412   New~module:\
1413   Namespace:~\l_stex_module_ns_str\
1414   Name:~\l_stex_module_name_str\
1415   Language:~\l_stex_module_lang_str\
1416   Signature:~\l_stex_module_sig_str\
1417   Metatheory:~\l_stex_module_meta_str\
1418   File:~\stex_path_to_string:N \g_stex_currentfile_seq
1419 }
1420
1421 \stex_if_do_html:T{
1422   \begin{stex_annotate_env} {theory} {
1423     \l_stex_module_ns_str ? \l_stex_module_name_str
1424   }
1425
1426   \stex_annotate_invisible:nnn{header}{} {
1427     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1428     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1429     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1430       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1431     }
1432     \str_if_empty:NF \smodulotype {
1433       \stex_annotate:nnn{type}{\smodulotype}{}
1434     }
1435   }
1436 }
1437 % TODO: Inherit metatheory for nested modules?
1438 }
1439 \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}

```

1440 \cs_new_protected:Nn \_stex_modules_end_module: {
1441   \stex_debug:nn{modules}{Closing~module~\prop_item:cn {c_stex_module\_l_stex_current_module}
1442   \_stex_reset_up_to_module:n \l_stex_current_module_str
1443   \stex_if_smsmode:T {
1444     \stex_persist:x {
1445       \prop_set_from_keyval:cn{c_stex_module\_l_stex_current_module_str _prop}{
1446         \exp_after:wN \prop_to_keyval:N \csname c_stex_module\_l_stex_current_module_str _pr
1447       }
1448       \seq_set_from_clist:cn{c_stex_module\_l_stex_current_module_str _constants}{
1449         \seq_use:cn{c_stex_module\_l_stex_current_module_str _constants},
1450       }
1451       \seq_set_from_clist:cn{c_stex_module\_l_stex_current_module_str _imports}{
1452         \seq_use:cn{c_stex_module\_l_stex_current_module_str _imports},
1453       }
1454       \tl_set:cn {c_stex_module\_l_stex_current_module_str _code}
1455     }
1456     \exp_after:wN \let \exp_after:wN \l_tmpa_tl \csname c_stex_module\_l_stex_current_module
1457     \exp_after:wN \stex_persist:n \exp_after:wN { \exp_after:wN { \l_tmpa_tl } }
1458   }
1459 }

```

(End definition for \\_stex\_modules\_end\_module:.)

The core environment

```
1460 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1461 \NewDocumentEnvironment { smodule } { 0{} m } {
1462   \stex_module_setup:nn{#1}{#2}
1463   \par
1464   \stex_if_smsmode:F{
1465     \tl_clear:N \l_tmpa_tl
1466     \clist_map_inline:Nn \smoduletype {
1467       \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1468         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1469       }
1470     }
1471     \tl_if_empty:NTF \l_tmpa_tl {
1472       \__stex_modules_smodule_start:
1473     }{
1474       \l_tmpa_tl
1475     }
1476   }
1477   \__stex_modules_begin_module:
1478   \str_if_empty:NF \smoduleid {
1479     \stex_ref_new_doc_target:n \smoduleid
1480   }
1481   \stex_smsmode_do:
1482 } {
1483   \__stex_modules_end_module:
1484   \stex_if_smsmode:F {
1485     \end{stex_annotate_env}
1486     \clist_set:No \l_tmpa_clist \smoduletype
1487     \tl_clear:N \l_tmpa_tl
1488     \clist_map_inline:Nn \l_tmpa_clist {
1489       \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1490         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1491       }
1492     }
1493     \tl_if_empty:NTF \l_tmpa_tl {
1494       \__stex_modules_smodule_end:
1495     }{
1496       \l_tmpa_tl
1497     }
1498   }
1499 }
```

**\stexpatchmodule**

```
1500 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1501 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1502
1503 \newcommand\stexpatchmodule[3] [] {
1504   \str_set:Nx \l_tmpa_str{ #1 }
1505   \str_if_empty:NTF \l_tmpa_str {
1506     \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1507     \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1508   }{
```

```

1509     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1510     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1511   }
1512 }

```

(End definition for `\stexpatchmodule`. This function is documented on page 55.)

## 27.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n
1513 \NewDocumentCommand \STEXModule { m } {
1514   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1515   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1516   \tl_set:Nn \l_tmpa_tl {
1517     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1518   }
1519   \seq_map_inline:Nn \l_stex_all_modules_seq {
1520     \str_set:Nn \l_tmpb_str { ##1 }
1521     \str_if_eq:eeT { \l_tmpa_str } {
1522       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1523     } {
1524       \seq_map_break:n {
1525         \tl_set:Nn \l_tmpa_tl {
1526           \stex_invoke_module:n { ##1 }
1527         }
1528       }
1529     }
1530   }
1531   \l_tmpa_tl
1532 }
1533
1534 \cs_new_protected:Nn \stex_invoke_module:n {
1535   \stex_debug:nn{modules}{Invoking~module~#1}
1536   \peek_charcode_remove:NTF ! {
1537     \__stex_modules_invoke_uri:nN { #1 }
1538   } {
1539     \peek_charcode_remove:NTF ? {
1540       \__stex_modules_invoke_symbol:nn { #1 }
1541     } {
1542       \msg_error:nnx{stex}{error/syntax}{
1543         ?~or~!~expected~after~
1544         \c_backslash_str STEXModule{#1}
1545       }
1546     }
1547   }
1548 }
1549
1550 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1551   \str_set:Nn #2 { #1 }
1552 }
1553
1554 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1555   \stex_invoke_symbol:n{#1?#2}

```

```
1556 }
```

*(End definition for \STEXModule and \stex\_invoke\_module:n. These functions are documented on page 55.)*

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

```
1557 \bool_new:N \l_stex_in_meta_bool
1558 \bool_set_false:N \l_stex_in_meta_bool
1559 \cs_new_protected:Nn \stex_activate_module:n {
1560   \stex_debug:nn{modules}{Activating~module~#1}
1561   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1562     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1563     \use:c{ c_stex_module_#1_code }
1564   }
1565 }
```

*(End definition for \stex\_activate\_module:n. This function is documented on page 56.)*

```
1566 \</package>
```

## Chapter 28

# STEX -Module Inheritance Implementation

```
1567 <*package>
1568
1569 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1570
```

### 28.1 SMS Mode

```
1571 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1572 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1573 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1574 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1575
1576 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1577   \makeatletter
1578   \makeatother
1579   \ExplSyntaxOn
1580   \ExplSyntaxOff
1581   \rustexBREAK
1582 }
1583
1584 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1585   \symdef
1586   \importmodule
1587   \notation
1588   \symdecl
1589   \STEXexport
1590   \inlineass
1591   \inlinedef
1592   \inlineex
1593   \endinput
1594   \setnotation
```

```

1595 \copynotation
1596 \assign
1597 \renamedekl
1598 \donotcopy
1599 \instantiate
1600 }
1601
1602 \exp_args:N Nx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1603   \tl_to_str:n {
1604     smodule,
1605     copymodule,
1606     interpretmodule,
1607     sdefinition,
1608     sexample,
1609     sassertion,
1610     sparagraph,
1611     mathstructure
1612   }
1613 }

```

(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 57.)

`\stex_if_smsmode_p:`  
`\stex_if_smsmode:TF`

```

1614 \bool_new:N \g__stex_smsmode_bool
1615 \bool_set_false:N \g__stex_smsmode_bool
1616 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1617   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1618 }

```

(End definition for `\stex_if_smsmode:TF`. This function is documented on page 57.)

`\_stex_smsmode_in_smsmode:nn`

```

1619 \cs_new_protected:Nn \_stex_smsmode_in_smsmode:nn { \stex_suppress_html:n {
1620   \vbox_set:Nn \l_tmpa_box {
1621     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1622     \bool_gset_true:N \g__stex_smsmode_bool
1623     #2
1624     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1625   }
1626   \box_clear:N \l_tmpa_box
1627 } }

```

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

`\stex_file_in_smsmode:nn`

```

1628 \quark_new:N \q__stex_smsmode_break
1629
1630 \NewDocumentCommand \_stex_smsmode_importmodule: { 0{} m } {
1631   \seq_gput_right:Nn \l__stex_smsmode_importmodules_seq { {#1}{#2} }
1632   \stex_smsmode_do:
1633 }
1634
1635 \cs_new_protected:Nn \_stex_smsmode_module:nn {
1636   \_stex_modules_args:n{#1}

```

```

1637 \stex_if_in_module:F {
1638   \str_if_empty:NF \l_stex_module_sig_str {
1639     \stex_modules_current_namespace:
1640     \str_set:Nx \l_stex_module_name_str { #2 }
1641     \stex_if_module_exists:nF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1642       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1643       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1644       \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1645       \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1646       \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1647       \str_set:Nx \l_tmpa_str {
1648         \stex_path_to_string:N \l_tmpa_seq /
1649         \l_tmpa_str . \l_stex_module_sig_str .tex
1650       }
1651       \IfFileExists \l_tmpa_str {
1652         \exp_args:NNx \seq_gput_right:Nn \l__stex_smsmode_sigmodules_seq \l_tmpa_str
1653       }{
1654         \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1655       }
1656     }
1657   }
1658 }
1659 }
1660
1661 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1662   \stex_filestack_push:n{#1}
1663   \seq_gclear:N \l__stex_smsmode_importmodules_seq
1664   \seq_gclear:N \l__stex_smsmode_sigmodules_seq
1665   % ----- new -----
1666   \__stex_smsmode_in_smsmode:nn{#1}{
1667     \let\importmodule\__stex_smsmode_importmodule:
1668     \let\stex_module_setup:nn\__stex_smsmode_module:nn
1669     \let\__stex_modules_begin_module:\relax
1670     \let\__stex_modules_end_module:\relax
1671     \seq_clear:N \g_stex_smsmode_allowedenvs_seq
1672     \exp_args:NNx \seq_put_right:Nn \g_stex_smsmode_allowedenvs_seq {\tl_to_str:n{module}}
1673     \tl_clear:N \g_stex_smsmode_allowedmacros_tl
1674     \tl_clear:N \g_stex_smsmode_allowedmacros_escape_tl
1675     \tl_put_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\importmodule}
1676     \everyeof{\q__stex_smsmode_break\noexpand}
1677     \expandafter\expandafter\expandafter
1678     \stex_smsmode_do:
1679     \csname @ @ input\endcsname "#1"\relax
1680
1681     \seq_map_inline:Nn \l__stex_smsmode_sigmodules_seq {
1682       \stex_filestack_push:n{##1}
1683       \expandafter\expandafter\expandafter
1684       \stex_smsmode_do:
1685       \csname @ @ input\endcsname "##1"\relax
1686       \stex_filestack_pop:
1687     }
1688   }
1689   % ----- new -----
1690   \__stex_smsmode_in_smsmode:nn{#1} {

```

```

1691 #2
1692 % ----- new -----
1693 \begingroup
1694 %\stex_debug:nn{smsmode}{Here:~\seq_use:Nn\l__stex_smsmode_importmodules_seq, }
1695 \seq_map_inline:Nn \l__stex_smsmode_importmodules_seq {
1696   \stex_import_module_uri:nn ##1
1697   \stex_import_require_module:nnnn
1698   \l_stex_import_ns_str
1699   \l_stex_import_archive_str
1700   \l_stex_import_path_str
1701   \l_stex_import_name_str
1702 }
1703 \endgroup
1704 \stex_debug:nn{smsmode}{Actually~loading~file~#1}
1705 % ----- new -----
1706 \everyeof{\q__stex_smsmode_break\noexpand}
1707 \expandafter\expandafter\expandafter
1708 \stex_smsmode_do:
1709 \csname @ @ input\endcsname "#1"\relax
1710 }
1711 \stex_filestack_pop:
1712 }

```

(End definition for `\stex_file_in_smsmode:nn`. This function is documented on page 58.)

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

```

1713 \cs_new_protected:Npn \stex_smsmode_do: {
1714   \stex_if_smsmode:T {
1715     \__stex_smsmode_do:w
1716   }
1717 }
1718 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1719   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
1720     \expandafter\if\expandafter\relax\noexpand#1
1721     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1722     \else\expandafter\__stex_smsmode_do:w\fi
1723   }{
1724     \__stex_smsmode_do:w % #1
1725   }
1726 }
1727 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1728   \cs_if_eq:NNF #1 \q__stex_smsmode_break {
1729     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1730       #1\__stex_smsmode_do:w
1731     }{
1732       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1733         #1
1734       }{
1735         \cs_if_eq:NNTF \begin #1 {
1736           \__stex_smsmode_check_begin:n
1737         }{
1738           \cs_if_eq:NNTF \end #1 {
1739             \__stex_smsmode_check_end:n

```



```

1740         }{
1741         \__stex_smsmode_do:w
1742         }
1743     }
1744 }
1745 }
1746 }
1747 }
1748
1749 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1750 \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1751 \begin{#1}
1752 }{
1753 \__stex_smsmode_do:w
1754 }
1755 }
1756 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1757 \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1758 \end{#1}\__stex_smsmode_do:w
1759 }{
1760 \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1761 }
1762 }

```

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

## 28.2 Inheritance

```

1763 <@@=stex_importmodule>

\stex_import_module_uri:nn

1764 \cs_new_protected:Nn \stex_import_module_uri:nn {
1765 \str_set:Nx \l_stex_import_archive_str { #1 }
1766 \str_set:Nn \l_stex_import_path_str { #2 }
1767
1768 \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1769 \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1770 \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1771
1772 \stex_modules_current_namespace:
1773 \bool_lazy_all:nTF {
1774 {\str_if_empty_p:N \l_stex_import_archive_str}
1775 {\str_if_empty_p:N \l_stex_import_path_str}
1776 {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1777 }{
1778 \str_set_eq:NN \l_stex_import_path_str \l_stex_module_subpath_str
1779 \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1780 }{
1781 \str_if_empty:NT \l_stex_import_archive_str {
1782 \prop_if_exist:NT \l_stex_current_repository_prop {
1783 \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1784 }
1785 }
1786 \str_if_empty:NTF \l_stex_import_archive_str {

```

```

1787     \str_if_empty:NF \l_stex_import_path_str {
1788         \str_set:Nx \l_stex_import_ns_str {
1789             \l_stex_module_ns_str / \l_stex_import_path_str
1790         }
1791     }
1792 }{
1793     \stex_require_repository:n \l_stex_import_archive_str
1794     \prop_get:cnN { c_stex_mathhub_\l_stex_import_archive_str _manifest_prop } { ns }
1795     \l_stex_import_ns_str
1796     \str_if_empty:NF \l_stex_import_path_str {
1797         \str_set:Nx \l_stex_import_ns_str {
1798             \l_stex_import_ns_str / \l_stex_import_path_str
1799         }
1800     }
1801 }
1802 }
1803 }

```

(End definition for \stex\_import\_module\_uri:nn. This function is documented on page 59.)

```

\l_stex_import_name_str Store the return values of \stex_import_module_uri:nn.
\l_stex_import_archive_str
\l_stex_import_path_str
\l_stex_import_ns_str

```

(End definition for \l\_stex\_import\_name\_str and others. These variables are documented on page 59.)

```

\stex_import_require_module:nnnnn {<ns>} {<archive-ID>} {<path>} {<name>}
1808 \cs_new_protected:Nn \stex_import_require_module:nnnnn {
1809     \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1810
1811         %\stex_debug:nn{requiremodule}{Here:\\~1::~~2:\\~3::~~3\\~4::~~4}
1812
1813         \exp_args:NNxx \seq_set_split:Nnn \l_tmpa_seq {\tl_to_str:n{/{}} {#4}
1814         \seq_get_left:NN \l_tmpa_seq \l_tmpc_str
1815
1816         %\stex_debug:nn{requiremodule}{Top~module:\l_tmpc_str}
1817
1818         % archive
1819         \str_set:Nx \l_tmpa_str { #2 }
1820         \str_if_empty:NTF \l_tmpa_str {
1821             \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1822         } {
1823             \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1824             \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1825             \seq_put_right:Nn \l_tmpa_seq { source }
1826         }
1827
1828         % path
1829         \str_set:Nx \l_tmpb_str { #3 }
1830         \str_if_empty:NTF \l_tmpb_str {
1831             \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / \l_tmpc_str }
1832

```

```

1833 \ltx@ifpackageloaded{babel} {
1834   \exp_args:NnX \prop_get:NnNF \c_stex_language_abbrevs_prop
1835     { \language } \l_tmpb_str {
1836       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1837     }
1838   } {
1839     \str_clear:N \l_tmpb_str
1840   }
1841
1842   %\stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1843   \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1844     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1845   }{
1846     %\stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1847     \IfFileExists{ \l_tmpa_str.tex }{
1848       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1849     }{
1850       % try english as default
1851       %\stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1852       \IfFileExists{ \l_tmpa_str.en.tex }{
1853         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1854       }{
1855         \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1856       }
1857     }
1858   }
1859
1860 } {
1861   \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1862   \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1863
1864   \ltx@ifpackageloaded{babel} {
1865     \exp_args:NnX \prop_get:NnNF \c_stex_language_abbrevs_prop
1866       { \language } \l_tmpb_str {
1867         \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1868       }
1869   } {
1870     \str_clear:N \l_tmpb_str
1871   }
1872
1873   \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1874
1875   %\stex_debug:nn{modules}{Checking~\l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex}
1876   \IfFileExists{ \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }{
1877     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }
1878   }{
1879     %\stex_debug:nn{modules}{Checking~\l_tmpa_str/\l_tmpc_str.tex}
1880     \IfFileExists{ \l_tmpa_str/\l_tmpc_str.tex }{
1881       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.tex }
1882     }{
1883       % try english as default
1884       %\stex_debug:nn{modules}{Checking~\l_tmpa_str/\l_tmpc_str.en.tex}
1885       \IfFileExists{ \l_tmpa_str/\l_tmpc_str.en.tex }{
1886         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.en.tex }

```

```

1887     }{
1888         %\stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1889         \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1890             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1891         }{
1892             %\stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1893             \IfFileExists{ \l_tmpa_str.tex }{
1894                 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1895             }{
1896                 % try english as default
1897                 %\stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1898                 \IfFileExists{ \l_tmpa_str.en.tex }{
1899                     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1900                 }{
1901                     \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1902                 }
1903             }
1904         }
1905     }
1906 }
1907 }
1908 }
1909
1910 \str_if_eq:eeF{\g__stex_importmodule_file_str}{\seq_use:Nn \g_stex_currentfile_seq /}{
1911     \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
1912         \seq_clear:N \l_stex_all_modules_seq
1913         \str_clear:N \l_stex_current_module_str
1914         \str_set:Nx \l_tmpb_str { #2 }
1915         \str_if_empty:NF \l_tmpb_str {
1916             \stex_set_current_repository:n { #2 }
1917         }
1918         \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1919     }
1920
1921     \stex_if_module_exists:nF { #1 ? #4 } {
1922         \msg_error:nnx{stex}{error/unknownmodule}{
1923             #1?#4~(in~file~\g__stex_importmodule_file_str)
1924         }
1925     }
1926 }
1927
1928 }
1929 \stex_activate_module:n { #1 ? #4 }
1930 }

```

(End definition for `\stex_import_require_module:nnnn`. This function is documented on page 59.)

## `\importmodule`

```

1931 \NewDocumentCommand \importmodule { 0{} m } {
1932     \stex_import_module_uri:nn { #1 } { #2 }
1933     \stex_debug:nn{modules}{Importing~module:~
1934         \l_stex_import_ns_str ? \l_stex_import_name_str
1935     }
1936     \stex_import_require_module:nnnn

```

```

1937 { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1938 { \l_stex_import_path_str } { \l_stex_import_name_str }
1939 \stex_if_smsmode:F {
1940   \stex_annotate_invisible:nnn
1941   {import} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1942 }
1943 \exp_args:Nx \stex_add_to_current_module:n {
1944   \stex_import_require_module:nnnn
1945   { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1946   { \l_stex_import_path_str } { \l_stex_import_name_str }
1947 }
1948 \exp_args:Nx \stex_add_import_to_current_module:n {
1949   \l_stex_import_ns_str ? \l_stex_import_name_str
1950 }
1951 \stex_smsmode_do:
1952 \ignorespacesandpars
1953 }
1954 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

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

#### `\usemodule`

```

1955 \NewDocumentCommand \usemodule { 0{} m } {
1956   \stex_if_smsmode:F {
1957     \stex_import_module_uri:nn { #1 } { #2 }
1958     \stex_import_require_module:nnnn
1959     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1960     { \l_stex_import_path_str } { \l_stex_import_name_str }
1961     \stex_annotate_invisible:nnn
1962     {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1963   }
1964   \stex_smsmode_do:
1965   \ignorespacesandpars
1966 }

```

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

```

1967 \cs_new_protected:Nn \stex_csl_to_imports:Nn {
1968   \tl_if_empty:nF{#2}{
1969     \clist_set:Nn \l_tmpa_clist {#2}
1970     \clist_map_inline:Nn \l_tmpa_clist {
1971       \tl_if_head_eq_charcode:nNTF {##1}[{
1972         #1 ##1
1973       }{
1974         #1{##1}
1975       }
1976     }
1977   }
1978 }
1979 \cs_generate_variant:Nn \stex_csl_to_imports:Nn {No}
1980
1981
1982 </package>

```

## Chapter 29

# STEX -Symbols Implementation

```
1983 <*package>
1984
1985 %%%%%%%%%% symbols.dtx %%%%%%%%%%
1986
    Warnings and error messages
1987 \msg_new:nnn{stex}{error/wrongargs}{
1988   args~value~in~symbol~declaration~for~#1~
1989   needs~to~be~i,~a,~b~or~B,~but~#2~given
1990 }
1991 \msg_new:nnn{stex}{error/unknownsymbol}{
1992   No~symbol~#1~found!
1993 }
1994 \msg_new:nnn{stex}{error/seqlength}{
1995   Expected~#1~arguments;~got~#2!
1996 }
1997 \msg_new:nnn{stex}{error/unknownnotation}{
1998   Unknown~notation~#1~for~#2!
1999 }
```

### 29.1 Symbol Declarations

```
2000 <@@=stex_symdecl>
\stex_all_symbols:n Map over all available symbols
2001 \cs_new_protected:Nn \stex_all_symbols:n {
2002   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
2003   \seq_map_inline:Nn \l_stex_all_modules_seq {
2004     \seq_map_inline:cn{c_stex_module_##1_constants}{
2005       \__stex_symdecl_all_symbols_cs{##1?####1}
2006     }
2007   }
2008 }
```

(End definition for `\stex_all_symbols:n`. This function is documented on page [61](#).)

## **\STEXsymbol**

```
2009 \NewDocumentCommand \STEXsymbol { m } {  
2010   \stex_get_symbol:n { #1 }  
2011   \exp_args:No  
2012   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }  
2013 }
```

(End definition for \STEXsymbol. This function is documented on page 62.)

symdecl arguments:

```
2014 \keys_define:nn { stex / symdecl } {  
2015   name      .str_set_x:N = \l_stex_symdecl_name_str ,  
2016   local     .bool_set:N = \l_stex_symdecl_local_bool ,  
2017   args      .str_set_x:N = \l_stex_symdecl_args_str ,  
2018   type      .tl_set:N = \l_stex_symdecl_type_tl ,  
2019   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,  
2020   align     .str_set:N = \l_stex_symdecl_align_str , % TODO(?)  
2021   gfc       .str_set:N = \l_stex_symdecl_gfc_str , % TODO(?)  
2022   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)  
2023   def       .tl_set:N = \l_stex_symdecl_definiens_tl ,  
2024   reorder   .str_set_x:N = \l_stex_symdecl_reorder_str ,  
2025   assoc     .choices:nn =  
2026             {bin,binl,binr,pre,conj,pwconj}  
2027             {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}  
2028 }  
2029  
2030 \bool_new:N \l_stex_symdecl_make_macro_bool  
2031  
2032 \cs_new_protected:Nn \__stex_symdecl_args:n {  
2033   \str_clear:N \l_stex_symdecl_name_str  
2034   \str_clear:N \l_stex_symdecl_args_str  
2035   \str_clear:N \l_stex_symdecl_deprecate_str  
2036   \str_clear:N \l_stex_symdecl_reorder_str  
2037   \str_clear:N \l_stex_symdecl_assoctype_str  
2038   \bool_set_false:N \l_stex_symdecl_local_bool  
2039   \tl_clear:N \l_stex_symdecl_type_tl  
2040   \tl_clear:N \l_stex_symdecl_definiens_tl  
2041  
2042   \keys_set:nn { stex / symdecl } { #1 }  
2043 }
```

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

```
2044  
2045 \NewDocumentCommand \symdecl { s m O{} } {  
2046   \__stex_symdecl_args:n { #3 }  
2047   \IfBooleanTF #1 {  
2048     \bool_set_false:N \l_stex_symdecl_make_macro_bool  
2049   } {  
2050     \bool_set_true:N \l_stex_symdecl_make_macro_bool  
2051   }  
2052   \stex_symdecl_do:n { #2 }  
2053   \stex_smsmode_do:  
2054 }
```

```

2055
2056 \cs_new_protected:Nn \stex_symdecl_do:nn {
2057   \__stex_symdecl_args:n{#1}
2058   \bool_set_false:N \l_stex_symdecl_make_macro_bool
2059   \stex_symdecl_do:n{#2}
2060 }
2061
2062 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

(End definition for \symdecl. This function is documented on page 60.)

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

```

2063 \cs_new_protected:Nn \stex_symdecl_do:n {
2064   \stex_if_in_module:F {
2065     % TODO throw error? some default namespace?
2066   }
2067
2068   \str_if_empty:NT \l_stex_symdecl_name_str {
2069     \str_set:Nx \l_stex_symdecl_name_str { #1 }
2070   }
2071
2072   \prop_if_exist:cT { l_stex_symdecl_
2073     \l_stex_current_module_str ?
2074     \l_stex_symdecl_name_str
2075     _prop
2076   }{
2077     % TODO throw error (beware of circular dependencies)
2078   }
2079
2080   \prop_clear:N \l_tmpa_prop
2081   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
2082   \seq_clear:N \l_tmpa_seq
2083   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
2084   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
2085
2086   \str_if_empty:NT \l_stex_symdecl_deprecate_str {
2087     \str_if_empty:NF \l_stex_module_deprecate_str {
2088       \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
2089     }
2090   }
2091   \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
2092
2093   \exp_args:No \stex_add_constant_to_current_module:n {
2094     \l_stex_symdecl_name_str
2095   }
2096
2097   % arity/args
2098   \int_zero:N \l_tmpb_int
2099
2100   \bool_set_true:N \l_tmpa_bool
2101   \str_map_inline:Nn \l_stex_symdecl_args_str {
2102     \token_case_meaning:NnF ##1 {
2103       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2104       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }

```



```

2105     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2106     {\tl_to_str:n a} {
2107         \bool_set_false:N \l_tmpa_bool
2108         \int_incr:N \l_tmpb_int
2109     }
2110     {\tl_to_str:n B} {
2111         \bool_set_false:N \l_tmpa_bool
2112         \int_incr:N \l_tmpb_int
2113     }
2114 }{
2115     \msg_error:nnxx{stex}{error/wrongargs}{
2116         \l_stex_current_module_str ?
2117         \l_stex_symdecl_name_str
2118     }{##1}
2119 }
2120 }
2121 \bool_if:NTF \l_tmpa_bool {
2122     % possibly numeric
2123     \str_if_empty:NTF \l_stex_symdecl_args_str {
2124         \prop_put:Nnn \l_tmpa_prop { args } {}
2125         \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2126     }{
2127         \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
2128         \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2129         \str_clear:N \l_tmpa_str
2130         \int_step_inline:nn \l_tmpa_int {
2131             \str_put_right:Nn \l_tmpa_str i
2132         }
2133         \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
2134     }
2135 } {
2136     \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2137     \prop_put:Nnx \l_tmpa_prop { arity }
2138     { \str_count:N \l_stex_symdecl_args_str }
2139 }
2140 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2141
2142 \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
2143     \prop_put:Nnx \l_tmpa_prop { defined }{ false }
2144 }{
2145     \prop_put:Nnx \l_tmpa_prop { defined }{ true }
2146 }
2147
2148 % semantic macro
2149
2150 \bool_if:NT \l_stex_symdecl_make_macro_bool {
2151     \exp_args:Nx \stex_do_up_to_module:n {
2152         \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2153             \l_stex_current_module_str ? \l_stex_symdecl_name_str
2154         }}
2155     }
2156 }
2157
2158 \stex_debug:nn{symbols}{New~symbol:~

```

```

2159 \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2160 Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2161 Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
2162 Definiens:~\exp_not:o {\l_stex_symdecl_definiens_tl}
2163 }
2164
2165 % circular dependencies require this:
2166 \stex_if_do_html:T {
2167   \stex_annotate_invisible:nnn {symdecl} {
2168     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2169   } {
2170     \tl_if_empty:NF \l_stex_symdecl_type_tl {
2171       \stex_annotate_invisible:nnn{type}{}{\l_stex_symdecl_type_tl$}
2172     }
2173     \stex_annotate_invisible:nnn{args}{}{
2174       \prop_item:Nn \l_tmpa_prop { args }
2175     }
2176     \stex_annotate_invisible:nnn{macroname}{#1}{}
2177     \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2178       \stex_annotate_invisible:nnn{definiens}{}
2179       {\l_stex_symdecl_definiens_tl$}
2180     }
2181     \str_if_empty:NF \l_stex_symdecl_assoc_type_str {
2182       \stex_annotate_invisible:nnn{assoc_type}{\l_stex_symdecl_assoc_type_str}{}
2183     }
2184     \str_if_empty:NF \l_stex_symdecl_reorder_str {
2185       \stex_annotate_invisible:nnn{reorderargs}{\l_stex_symdecl_reorder_str}{}
2186     }
2187   }
2188 }
2189 \prop_if_exist:cF {
2190   l_stex_symdecl_
2191   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2192   _prop
2193 } {
2194   \bool_if:NTF \l_stex_symdecl_local_bool \stex_do_up_to_module:x \stex_execute_in_module:
2195   \__stex_symdecl_restore_symbol:nnnnnnn
2196     {\l_stex_symdecl_name_str}
2197     { \prop_item:Nn \l_tmpa_prop {args} }
2198     { \prop_item:Nn \l_tmpa_prop {arity} }
2199     { \prop_item:Nn \l_tmpa_prop {assoc} }
2200     { \prop_item:Nn \l_tmpa_prop {defined} }
2201     {\bool_if:NT \l_stex_symdecl_make_macro_bool {#1} }
2202     {\l_stex_current_module_str}
2203 }
2204 }
2205 }
2206 \cs_new_protected:Nn \__stex_symdecl_restore_symbol:nnnnnnn {
2207   \prop_clear:N \l_tmpa_prop
2208   \prop_put:Nnn \l_tmpa_prop { module } { #7 }
2209   \prop_put:Nnn \l_tmpa_prop { name } { #1 }
2210   \prop_put:Nnn \l_tmpa_prop { args } { #2 }
2211   \prop_put:Nnn \l_tmpa_prop { arity } { #3 }
2212   \prop_put:Nnn \l_tmpa_prop { assoc } { #4 }

```

```

2213 \prop_put:Nnn \l_tmpa_prop { defined } { #5 }
2214 \tl_if_empty:nF{#6}{
2215   \tl_set:cx{#6}{\stex_invoke_symbol:n{\detokenize{#7 ? #1}}}
2216 }
2217 \prop_set_eq:cN{l_stex_symdecl_ \detokenize{#7 ? #1} _prop}\l_tmpa_prop
2218 \seq_clear:c{l_stex_symdecl_ \detokenize{#7 ? #1} _notations}
2219 }

```

(End definition for `\stex_symdecl_do:n`. This function is documented on page 61.)

`\stex_get_symbol:n`

```

2220 \str_new:N \l_stex_get_symbol_uri_str
2221
2222 \cs_new_protected:Nn \stex_get_symbol:n {
2223   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2224     \tl_set:Nn \l_tmpa_tl { #1 }
2225     \__stex_symdecl_get_symbol_from_cs:
2226   }{
2227     % argument is a string
2228     % is it a command name?
2229     \cs_if_exist:cTF { #1 }{
2230       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2231       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2232       \str_if_empty:NNTF \l_tmpa_str {
2233         \exp_args:Nx \cs_if_eq:NNTF {
2234           \tl_head:N \l_tmpa_tl
2235         } \stex_invoke_symbol:n {
2236           \__stex_symdecl_get_symbol_from_cs:
2237         }{
2238           \__stex_symdecl_get_symbol_from_string:n { #1 }
2239         }
2240       } {
2241         \__stex_symdecl_get_symbol_from_string:n { #1 }
2242       }
2243     }{
2244       % argument is not a command name
2245       \__stex_symdecl_get_symbol_from_string:n { #1 }
2246       % \l_stex_all_symbols_seq
2247     }
2248   }
2249   \str_if_eq:eeF {
2250     \prop_item:cn {
2251       l_stex_symdecl_\l_stex_get_symbol_uri_str _prop
2252     }{ deprecate }
2253   }{
2254     \msg_warning:nxxx{stex}{warning/deprecated}{
2255       Symbol~\l_stex_get_symbol_uri_str
2256     }{
2257       \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{ deprecate }
2258     }
2259   }
2260 }
2261
2262 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {

```

```

2263 \tl_set:Nn \l_tmpa_tl {
2264   \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2265 }
2266 \str_set:Nn \l_tmpa_str { #1 }
2267
2268 %\int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2269
2270 \str_if_in:NnTF \l_tmpa_str ? {
2271   \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
2272   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
2273   \str_set:Nx \l_tmpb_str {\seq_use:Nn \l_tmpa_seq ?}
2274 }{
2275   \str_clear:N \l_tmpb_str
2276 }
2277 \str_if_empty:NNTF \l_tmpb_str {
2278   \seq_map_inline:Nn \l_stex_all_modules_seq {
2279     \seq_map_inline:cn{c_stex_module_###1_constants}{
2280       \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2281         \seq_map_break:n{\seq_map_break:n{
2282           \tl_set:Nn \l_tmpa_tl {
2283             \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2284           }
2285         }}
2286       }
2287     }
2288   }
2289 }{
2290   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpb_str }
2291   \seq_map_inline:Nn \l_stex_all_modules_seq {
2292     \str_if_eq:eeT{ \l_tmpb_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2293       \seq_map_inline:cn{c_stex_module_###1_constants}{
2294         \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2295           \seq_map_break:n{\seq_map_break:n{
2296             \tl_set:Nn \l_tmpa_tl {
2297               \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2298             }
2299           }}
2300         }
2301       }
2302     }
2303   }
2304 }
2305
2306 \l_tmpa_tl
2307 }
2308
2309 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
2310   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2311     { \tl_tail:N \l_tmpa_tl }
2312   \tl_if_single:NNTF \l_tmpa_tl {
2313     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2314       \exp_after:wN \str_set:Nn \exp_after:wN
2315         \l_stex_get_symbol_uri_str \l_tmpa_tl
2316     }{

```

```

2317     % TODO
2318     % tail is not a single group
2319   }
2320 }{
2321   % TODO
2322   % tail is not a single group
2323 }
2324 }

```

(End definition for `\stex_get_symbol:n`. This function is documented on page 61.)

## 29.2 Notations

```

2325 <@@=stex_notation>

notation arguments:
2326 \keys_define:nn { stex / notation } {
2327   % lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2328   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2329   prec     .str_set_x:N = \l__stex_notation_prec_str ,
2330   op       .tl_set:N = \l__stex_notation_op_tl ,
2331   primary .bool_set:N = \l__stex_notation_primary_bool ,
2332   primary .default:n = {true} ,
2333   unknown .code:n = \str_set:Nx
2334             \l__stex_notation_variant_str \l_keys_key_str
2335 }
2336
2337 \cs_new_protected:Nn \stex_notation_args:n {
2338   % \str_clear:N \l__stex_notation_lang_str
2339   \str_clear:N \l__stex_notation_variant_str
2340   \str_clear:N \l__stex_notation_prec_str
2341   \tl_clear:N \l__stex_notation_op_tl
2342   \bool_set_false:N \l__stex_notation_primary_bool
2343
2344   \keys_set:nn { stex / notation } { #1 }
2345 }

\notation

2346 \NewDocumentCommand \notation { s m O{}} {
2347   \stex_notation_args:n { #3 }
2348   \tl_clear:N \l_stex_symdecl_definiens_tl
2349   \stex_get_symbol:n { #2 }
2350   \tl_set:Nn \l_stex_notation_after_do_tl {
2351     \__stex_notation_final:
2352     \IfBooleanTF#1{
2353       \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2354     }{}
2355     \stex_smsmode_do:\ignorespacesandpars
2356   }
2357   \stex_notation_do:nnnnn
2358   { \prop_item:cn {\l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } }
2359   { \prop_item:cn { \l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } }
2360   { \l__stex_notation_variant_str }
2361   { \l__stex_notation_prec_str }

```

```

2362 }
2363 \stex_deactivate_macro:Nn \notation {module-environments}

```

(End definition for \notation. This function is documented on page 61.)

\stex\_notation\_do:nnnnn

```

2364 \seq_new:N \l__stex_notation_precedences_seq
2365 \tl_new:N \l__stex_notation_opprec_tl
2366 \int_new:N \l__stex_notation_currarg_int
2367 \tl_new:N \stex_symbol_after_invokation_tl
2368
2369 \cs_new_protected:Nn \stex_notation_do:nnnnn {
2370   \let\l_stex_current_symbol_str\relax
2371   \seq_clear:N \l__stex_notation_precedences_seq
2372   \tl_clear:N \l__stex_notation_opprec_tl
2373   \str_set:Nx \l__stex_notation_args_str { #1 }
2374   \str_set:Nx \l__stex_notation_arity_str { #2 }
2375   \str_set:Nx \l__stex_notation_suffix_str { #3 }
2376   \str_set:Nx \l__stex_notation_prec_str { #4 }
2377
2378   % precedences
2379   \str_if_empty:NTF \l__stex_notation_prec_str {
2380     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2381       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2382     }{
2383       \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2384     }
2385   } {
2386     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2387       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2388       \int_step_inline:nn { \l__stex_notation_arity_str } {
2389         \exp_args:NNo
2390         \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2391       }
2392     }{
2393       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2394       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2395         \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2396         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2397           \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2398             \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2399           \seq_map_inline:Nn \l_tmpa_seq {
2400             \seq_put_right:Nn \l_tmpb_seq { ##1 }
2401           }
2402         }
2403       }{
2404         \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2405           \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2406         }{
2407           \tl_set:No \l__stex_notation_opprec_tl { 0 }
2408         }
2409       }
2410     }
2411   }

```

```

2412
2413 \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2414 \int_step_inline:nn { \l__stex_notation_arity_str } {
2415   \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
2416     \exp_args:NNo
2417     \seq_put_right:No \l__stex_notation_precedences_seq {
2418       \l__stex_notation_opprec_tl
2419     }
2420   }
2421 }
2422 \tl_clear:N \l_stex_notation_dummyargs_tl
2423
2424 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2425   \exp_args:NNe
2426   \cs_set:Npn \l_stex_notation_macrocode_cs {
2427     \_stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2428     { \l__stex_notation_suffix_str }
2429     { \l__stex_notation_opprec_tl }
2430     { \exp_not:n { #5 } }
2431   }
2432   \l_stex_notation_after_do_tl
2433 }{
2434   \str_if_in:NnTF \l__stex_notation_args_str b {
2435     \exp_args:Nne \use:nn
2436     {
2437       \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2438       \cs_set:Npn \l__stex_notation_arity_str } { {
2439         \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2440         { \l__stex_notation_suffix_str }
2441         { \l__stex_notation_opprec_tl }
2442         { \exp_not:n { #5 } }
2443       } }
2444   }{
2445     \str_if_in:NnTF \l__stex_notation_args_str B {
2446       \exp_args:Nne \use:nn
2447       {
2448         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2449         \cs_set:Npn \l__stex_notation_arity_str } { {
2450           \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2451           { \l__stex_notation_suffix_str }
2452           { \l__stex_notation_opprec_tl }
2453           { \exp_not:n { #5 } }
2454         } }
2455     }{
2456       \exp_args:Nne \use:nn
2457       {
2458         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2459         \cs_set:Npn \l__stex_notation_arity_str } { {
2460           \_stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2461           { \l__stex_notation_suffix_str }
2462           { \l__stex_notation_opprec_tl }
2463           { \exp_not:n { #5 } }
2464         } }
2465     }

```

```

2466     }
2467
2468     \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2469     \int_zero:N \l__stex_notation_currarg_int
2470     \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2471     \__stex_notation_arguments:
2472   }
2473 }

```

(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

```

2474 \cs_new_protected:Nn \__stex_notation_arguments: {
2475   \int_incr:N \l__stex_notation_currarg_int
2476   \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2477     \l_stex_notation_after_do_tl
2478   }{
2479     \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2480     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2481     \str_if_eq:VnTF \l_tmpa_str a {
2482       \__stex_notation_argument_assoc:nn{a}
2483     }{
2484       \str_if_eq:VnTF \l_tmpa_str B {
2485         \__stex_notation_argument_assoc:nn{B}
2486       }{
2487         \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpb_str
2488         \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2489           { \stex_term_math_arg:nnn
2490             { \l_tmpa_str\int_use:N \l__stex_notation_currarg_int }
2491             { \l_tmpb_str }
2492             { ###\int_use:N \l__stex_notation_currarg_int }
2493           }
2494         }
2495         \__stex_notation_arguments:
2496       }
2497     }
2498   }
2499 }

```

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

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

```

2500 \cs_new_protected:Nn \__stex_notation_argument_assoc:nn {
2501
2502   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2503     {\l__stex_notation_arity_str}{
2504       #2
2505     }
2506   \int_zero:N \l_tmpa_int
2507   \tl_clear:N \l_tmpa_tl
2508   \str_map_inline:Nn \l__stex_notation_args_str {
2509     \int_incr:N \l_tmpa_int
2510     \tl_put_right:Nx \l_tmpa_tl {
2511       \str_if_eq:nnTF {##1}{a}{ { } }{ }

```



```

2512     \str_if_eq:nnTF {##1}{B}{ } }{
2513         {\_stex_term_arg:nn{##1\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa
2514         }
2515     }
2516 }
2517 }
2518 \exp_after:wN\exp_after:wN\exp_after:wN \def
2519 \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2520 \exp_after:wN\exp_after:wN\exp_after:wN ##
2521 \exp_after:wN\exp_after:wN\exp_after:wN 1
2522 \exp_after:wN\exp_after:wN\exp_after:wN ##
2523 \exp_after:wN\exp_after:wN\exp_after:wN 2
2524 \exp_after:wN\exp_after:wN\exp_after:wN {
2525     \exp_after:wN \exp_after:wN \exp_after:wN
2526     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2527         \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2528     }
2529 }
2530
2531 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2532 \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
2533     \_stex_term_math_assoc_arg:nnnn
2534     { #1\int_use:N \l__stex_notation_currarg_int }
2535     { \l_tmpa_str }
2536     { #####\int_use:N \l__stex_notation_currarg_int }
2537     { \l_tmpa_cs {####1} {####2} }
2538 } }
2539 \__stex_notation_arguments:
2540 }

```

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

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

```

2541 \cs_new_protected:Nn \_stex_notation_restore_notation:nnnnn {
2542     \cs_generate_from_arg_count:cNnn{stex_notation_\detokenize{#1} \c_hash_str \detokenize{#2}}
2543     \cs_set_nopar:Npn {#3}{#4}
2544     \tl_if_empty:nF {#5}{
2545         \tl_set:cn{stex_op_notation_\detokenize{#1} \c_hash_str \detokenize{#2}_cs}{\comp{ #5 }
2546     }
2547     \seq_if_exist:cT { l_stex_symdecl_\detokenize{#1} _notations }{
2548         \seq_put_right:cx { l_stex_symdecl_\detokenize{#1} _notations } { \detokenize{#2} }
2549     }
2550 }
2551
2552 \cs_new_protected:Nn \_stex_notation_final: {
2553
2554     \stex_execute_in_module:x {
2555         \_stex_notation_restore_notation:nnnnn
2556         {\l_stex_get_symbol_uri_str}
2557         {\l__stex_notation_suffix_str}
2558         {\l__stex_notation_arity_str}
2559     {
2560         \exp_after:wN \exp_after:wN \exp_after:wN
2561         \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN

```

```

2562     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2563   }
2564   {\exp_args:No \exp_not:n \l__stex_notation_op_tl }
2565 }
2566
2567 \stex_debug:nn{symbols}{
2568   Notation~\l__stex_notation_suffix_str
2569   ~for~\l_stex_get_symbol_uri_str^^J
2570   Operator~precedence:~\l__stex_notation_opprec_tl^^J
2571   Argument~precedences:~
2572     \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2573   Notation: \cs_meaning:c {
2574     stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2575     \l__stex_notation_suffix_str
2576     _cs
2577   }
2578 }
2579 % HTML annotations
2580 \stex_if_do_html:T {
2581   \stex_annotate_invisible:nnn { notation }
2582   { \l_stex_get_symbol_uri_str } {
2583     \stex_annotate_invisible:nnn { notationfragment }
2584     { \l__stex_notation_suffix_str }{}
2585     \stex_annotate_invisible:nnn { precedence }
2586     { \l__stex_notation_prec_str }{}
2587
2588     \int_zero:N \l_tmpa_int
2589     \str_set_eq:NN \l__stex_notation_remaining_args_str \l_stex_notation_args_str
2590     \tl_clear:N \l_tmpa_tl
2591     \int_step_inline:nn { \l__stex_notation_arity_str }{
2592       \int_incr:N \l_tmpa_int
2593       \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2594       \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_rema
2595       \str_if_eq:VnTF \l_tmpb_str a {
2596         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2597           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2598           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2599         } }
2600       }{
2601         \str_if_eq:VnTF \l_tmpb_str B {
2602           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2603             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2604             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2605           } }
2606         }{
2607           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2608             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
2609           } }
2610         }
2611       }
2612     }
2613     \stex_annotate_invisible:nnn { notationcomp }{}{
2614       \str_set:Nx \l_stex_current_symbol_str {\l_stex_get_symbol_uri_str }
2615       $ \exp_args:Nno \use:nn { \use:c {

```

```

2616         stex_notation_ \l_stex_current_symbol_str
2617         \c_hash_str \l__stex_notation_suffix_str _cs
2618     } } { \l_tmpa_tl } $
2619   }
2620 }
2621 }
2622 }

```

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

## \setnotation

```

2623 \keys_define:nn { stex / setnotation } {
2624   % lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2625   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2626   unknown .code:n      = \str_set:Nx
2627     \l__stex_notation_variant_str \l_keys_key_str
2628 }
2629
2630 \cs_new_protected:Nn \stex_setnotation_args:n {
2631   % \str_clear:N \l__stex_notation_lang_str
2632   \str_clear:N \l__stex_notation_variant_str
2633   \keys_set:nn { stex / setnotation } { #1 }
2634 }
2635
2636 \cs_new_protected:Nn \__stex_notation_setnotation:nn {
2637   \seq_if_exist:cT{l_stex_symdecl_#1_notations}{
2638     \seq_remove_all:cn { l_stex_symdecl_#1_notations }{ #2 }
2639     \seq_put_left:cn { l_stex_symdecl_#1_notations }{ #2 }
2640   }
2641 }
2642
2643 \cs_new_protected:Nn \stex_setnotation:n {
2644   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1_notations }
2645   { \l__stex_notation_variant_str }{
2646     \stex_execute_in_module:x{ \__stex_notation_setnotation:nn {#1}{\l__stex_notation_vari
2647     \stex_debug:nn {notations}{
2648       Setting~default~notation~
2649       {\l__stex_notation_variant_str }~for~
2650       #1 \\
2651       \expandafter\meaning\csname
2652       l_stex_symdecl_#1_notations\endcsname
2653     }
2654   }{
2655     \msg_error:nnxx{stex}{unknownnotation}{\l__stex_notation_variant_str}{#1}
2656   }
2657 }
2658
2659 \NewDocumentCommand \setnotation {m m} {
2660   \stex_get_symbol:n { #1 }
2661   \stex_setnotation_args:n { #2 }
2662   \stex_setnotation:n{\l_stex_get_symbol_uri_str}
2663   \stex_smsmode_do:\ignorespacesandpars
2664 }
2665

```

```

2666 \cs_new_protected:Nn \stex_copy_notations:nn {
2667   \stex_debug:nn {notations}{
2668     Copying~notations~from~#2~to~#1\
2669     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
2670   }
2671   \tl_clear:N \l_tmpa_tl
2672   \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2673     \tl_put_right:Nn \l_tmpa_tl { {##} ##1} }
2674   }
2675   \seq_map_inline:cn {l_stex_symdecl_#2_notations}{
2676     \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2677     \edef \l_tmpa_tl {
2678       \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2679       \exp_after:wN\exp_after:wN\exp_after:wN {
2680         \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2681       }
2682     }
2683   }
2684   \stex_execute_in_module:x {
2685     \__stex_notation_restore_notation:nnnnn
2686     {#1}{##1}
2687     { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }
2688     { \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl} }
2689     {
2690       \cs_if_exist:cT{stex_op_notation_ #2\c_hash_str ##1 _cs}{
2691         \exp_args:NNo\exp_args:No\exp_not:n{\c_name stex_op_notation_ #2\c_hash_str ##1
2692       }
2693     }
2694   }
2695 }
2696 }
2697
2698 \NewDocumentCommand \copynotation {m m} {
2699   \stex_get_symbol:n { #1 }
2700   \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2701   \stex_get_symbol:n { #2 }
2702   \exp_args:Noo
2703   \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2704   \stex_smsmode_do:\ignorespacesandpars
2705 }
2706

```

(End definition for \setnotation. This function is documented on page 18.)

## \symdef

```

2707 \keys_define:nn { stex / symdef } {
2708   name .str_set_x:N = \l_stex_symdecl_name_str ,
2709   local .bool_set:N = \l_stex_symdecl_local_bool ,
2710   args .str_set_x:N = \l_stex_symdecl_args_str ,
2711   type .tl_set:N = \l_stex_symdecl_type_tl ,
2712   def .tl_set:N = \l_stex_symdecl_definiens_tl ,
2713   reorder .str_set_x:N = \l_stex_symdecl_reorder_str ,
2714   op .tl_set:N = \l__stex_notation_op_tl ,
2715   % lang .str_set_x:N = \l__stex_notation_lang_str ,

```

```

2716   variant .str_set_x:N = \l__stex_notation_variant_str ,
2717   prec    .str_set_x:N = \l__stex_notation_prec_str ,
2718   assoc   .choices:nn =
2719       {bin,binl,binr,pre,conj,pwconj}
2720       {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
2721   unknown .code:n      = \str_set:Nx
2722       \l__stex_notation_variant_str \l_keys_key_str
2723 }
2724
2725 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2726   \str_clear:N \l_stex_symdecl_name_str
2727   \str_clear:N \l_stex_symdecl_args_str
2728   \str_clear:N \l_stex_symdecl_assoctype_str
2729   \str_clear:N \l_stex_symdecl_reorder_str
2730   \bool_set_false:N \l_stex_symdecl_local_bool
2731   \tl_clear:N \l_stex_symdecl_type_tl
2732   \tl_clear:N \l_stex_symdecl_definiens_tl
2733   % \str_clear:N \l__stex_notation_lang_str
2734   \str_clear:N \l__stex_notation_variant_str
2735   \str_clear:N \l__stex_notation_prec_str
2736   \tl_clear:N \l__stex_notation_op_tl
2737
2738   \keys_set:nn { stex / symdef } { #1 }
2739 }
2740
2741 \NewDocumentCommand \symdef { m O{} } {
2742   \__stex_notation_symdef_args:n { #2 }
2743   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2744   \stex_symdecl_do:n { #1 }
2745   \tl_set:Nn \l_stex_notation_after_do_tl {
2746     \__stex_notation_final:
2747     \stex_smsmode_do:\ignorespacesandpars
2748   }
2749   \str_set:Nx \l_stex_get_symbol_uri_str {
2750     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2751   }
2752   \exp_args:Nx \stex_notation_do:nnnnn
2753     { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } }
2754     { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } }
2755     { \l__stex_notation_variant_str }
2756     { \l__stex_notation_prec_str }
2757 }
2758 \stex_deactivate_macro:Nn \symdef {module~environments}

```

(End definition for `\symdef`. This function is documented on page 61.)

## 29.3 Variables

```

2759 <@@=stex_variables>
2760
2761 \keys_define:nn { stex / vardef } {
2762   name .str_set_x:N = \l__stex_variables_name_str ,
2763   args .str_set_x:N = \l__stex_variables_args_str ,
2764   type .tl_set:N    = \l__stex_variables_type_tl ,

```

```

2765 def      .tl_set:N      = \l__stex_variables_def_tl ,
2766 op       .tl_set:N      = \l__stex_variables_op_tl ,
2767 prec     .str_set_x:N    = \l__stex_variables_prec_str ,
2768 assoc    .choices:nn    =
2769     {bin,binl,binr,pre,conj,pwconj}
2770     {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
2771 bind     .choices:nn    =
2772     {forall,exists}
2773     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2774 }
2775
2776 \cs_new_protected:Nn \__stex_variables_args:n {
2777   \str_clear:N \l__stex_variables_name_str
2778   \str_clear:N \l__stex_variables_args_str
2779   \str_clear:N \l__stex_variables_prec_str
2780   \str_clear:N \l__stex_variables_assoctype_str
2781   \str_clear:N \l__stex_variables_bind_str
2782   \tl_clear:N \l__stex_variables_type_tl
2783   \tl_clear:N \l__stex_variables_def_tl
2784   \tl_clear:N \l__stex_variables_op_tl
2785
2786   \keys_set:nn { stex / vardef } { #1 }
2787 }
2788
2789 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{} } {
2790   \__stex_variables_args:n {#2}
2791   \str_if_empty:NT \l__stex_variables_name_str {
2792     \str_set:Nx \l__stex_variables_name_str { #1 }
2793   }
2794   \prop_clear:N \l_tmpa_prop
2795   \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
2796
2797   \int_zero:N \l_tmpb_int
2798   \bool_set_true:N \l_tmpa_bool
2799   \str_map_inline:Nn \l__stex_variables_args_str {
2800     \token_case_meaning:NnF ##1 {
2801       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2802       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2803       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2804       {\tl_to_str:n a} {
2805         \bool_set_false:N \l_tmpa_bool
2806         \int_incr:N \l_tmpb_int
2807       }
2808       {\tl_to_str:n B} {
2809         \bool_set_false:N \l_tmpa_bool
2810         \int_incr:N \l_tmpb_int
2811       }
2812     }{
2813       \msg_error:nnxx{stex}{error/wrongargs}{
2814         variable~\l__stex_variables_name_str
2815       }{##1}
2816     }
2817   }
2818   \bool_if:NTF \l_tmpa_bool {

```

```

2819 % possibly numeric
2820 \str_if_empty:NTF \l__stex_variables_args_str {
2821   \prop_put:Nnn \l_tmpa_prop { args } {}
2822   \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2823 }{
2824   \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
2825   \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2826   \str_clear:N \l_tmpa_str
2827   \int_step_inline:nn \l_tmpa_int {
2828     \str_put_right:Nn \l_tmpa_str i
2829   }
2830   \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
2831   \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2832 }
2833 } {
2834   \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2835   \prop_put:Nnx \l_tmpa_prop { arity }
2836     { \str_count:N \l__stex_variables_args_str }
2837 }
2838 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2839 \tl_set:cx { #1 }{ \stex_invoke_variable:n { \l__stex_variables_name_str } }
2840
2841 \prop_set_eq:cN { l_stex_variable_\l__stex_variables_name_str _prop } \l_tmpa_prop
2842
2843 \tl_if_empty:NF \l__stex_variables_op_tl {
2844   \cs_set:cpx {
2845     stex_var_op_notation_ \l__stex_variables_name_str _cs
2846   } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
2847 }
2848
2849 \tl_set:Nn \l_stex_notation_after_do_tl {
2850   \exp_args:Nne \use:nn {
2851     \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str _cs }
2852     \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
2853   } {{
2854     \exp_after:wN \exp_after:wN \exp_after:wN
2855     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2856     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_symbol
2857   }}
2858 \stex_if_do_html:T {
2859   \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
2860     \stex_annotate_invisible:nnn { precedence }
2861       { \l__stex_variables_prec_str }{}
2862     \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{}{\l__
2863     \stex_annotate_invisible:nnn{args}{}{\l__stex_variables_args_str }
2864     \stex_annotate_invisible:nnn{macroname}{#1}{}
2865     \tl_if_empty:NF \l__stex_variables_def_tl {
2866       \stex_annotate_invisible:nnn{definiens}{}
2867       {\l__stex_variables_def_tl$}
2868     }
2869     \str_if_empty:NF \l__stex_variables_assoctype_str {
2870       \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{}
2871     }
2872     \str_if_empty:NF \l__stex_variables_bind_str {

```

```

2873     \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{ }
2874 }
2875 \int_zero:N \l_tmpa_int
2876 \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
2877 \tl_clear:N \l_tmpa_tl
2878 \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
2879   \int_incr:N \l_tmpa_int
2880   \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
2881   \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables
2882   \str_if_eq:VnTF \l_tmpb_str a {
2883     \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2884       \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{ } ,
2885       \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{ }
2886     } }
2887   }{
2888     \str_if_eq:VnTF \l_tmpb_str B {
2889       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2890         \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{ } ,
2891         \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{ }
2892       } }
2893     }{
2894       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2895         \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{ }
2896       } }
2897     }
2898   }
2899 }
2900 \stex_annotate_invisible:nnn { notationcomp }{ }{
2901   \str_set:Nx \l_stex_current_symbol_str {var://\l__stex_variables_name_str }
2902   $ \exp_args:Nno \use:nn { \use:c {
2903     stex_var_notation_\l__stex_variables_name_str_cs
2904   } } { \l_tmpa_tl } $
2905 }
2906 }
2907 }\ignorespacesandpars
2908 }
2909
2910 \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
2911 }
2912
2913 \cs_new:Nn \_stex_reset:N {
2914   \tl_if_exist:NTF #1 {
2915     \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
2916   }{
2917     \let \exp_not:N #1 \exp_not:N \undefined
2918   }
2919 }
2920
2921 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
2922   \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
2923   \exp_args:Nnx \use:nn {
2924     % TODO
2925     \stex_annotate_invisible:nnn {vardecl}{\clist_use:Nn\l__stex_variables_names,}{
2926     #2

```



```

2927     }
2928   }{
2929     \_stex_reset:N \varnot
2930     \_stex_reset:N \vartype
2931     \_stex_reset:N \vardefi
2932   }
2933 }
2934
2935 \NewDocumentCommand \vardef { s } {
2936   \IfBooleanTF#1 {
2937     \__stex_variables_do_complex:nn
2938   }{
2939     \__stex_variables_do_simple:nnn
2940   }
2941 }
2942
2943 \NewDocumentCommand \svar { 0{} m }{
2944   \tl_if_empty:nTF {#1}{
2945     \str_set:Nn \l_tmpa_str { #2 }
2946   }{
2947     \str_set:Nn \l_tmpa_str { #1 }
2948   }
2949   \_stex_term_omv:nn {
2950     var://\l_tmpa_str
2951   }{
2952     \exp_args:Nnx \use:nn {
2953       \def\comp{\_varcomp}
2954       \str_set:Nx \l_stex_current_symbol_str { var://\l_tmpa_str }
2955       \comp{ #2 }
2956     }{
2957       \_stex_reset:N \comp
2958       \_stex_reset:N \l_stex_current_symbol_str
2959     }
2960   }
2961 }
2962
2963
2964
2965 \keys_define:nn { stex / varseq } {
2966   name .str_set_x:N = \l__stex_variables_name_str ,
2967   args .int_set:N   = \l__stex_variables_args_int ,
2968   type .tl_set:N    = \l__stex_variables_type_tl ,
2969   mid .tl_set:N     = \l__stex_variables_mid_tl ,
2970   bind .choices:nn =
2971     {forall,exists}
2972     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2973 }
2974
2975 \cs_new_protected:Nn \__stex_variables_seq_args:n {
2976   \str_clear:N \l__stex_variables_name_str
2977   \int_set:Nn \l__stex_variables_args_int 1
2978   \tl_clear:N \l__stex_variables_type_tl
2979   \str_clear:N \l__stex_variables_bind_str
2980

```

```

2981 \keys_set:nn { stex / varseq } { #1 }
2982 }
2983
2984 \NewDocumentCommand \varseq {m O{} m m m}{
2985   \__stex_variables_seq_args:n { #2 }
2986   \str_if_empty:NT \l__stex_variables_name_str {
2987     \str_set:Nx \l__stex_variables_name_str { #1 }
2988   }
2989   \prop_clear:N \l_tmpa_prop
2990   \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
2991
2992   \seq_set_from_clist:Nn \l_tmpa_seq {#3}
2993   \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
2994     \msg_error:nnxx{stex}{error/seqlength}
2995     {\int_use:N \l__stex_variables_args_int}
2996     {\seq_count:N \l_tmpa_seq}
2997   }
2998   \seq_set_from_clist:Nn \l_tmpb_seq {#4}
2999   \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
3000     \msg_error:nnxx{stex}{error/seqlength}
3001     {\int_use:N \l__stex_variables_args_int}
3002     {\seq_count:N \l_tmpb_seq}
3003   }
3004   \prop_put:Nnn \l_tmpa_prop {starts} {#3}
3005   \prop_put:Nnn \l_tmpa_prop {ends} {#4}
3006
3007   \cs_generate_from_arg_count:cNnn {stex_varseq\l__stex_variables_name_str _cs}
3008   \cs_set:Npn {\int_use:N \l__stex_variables_args_int} { #5 }
3009
3010   \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}
3011   \int_step_inline:nn \l__stex_variables_args_int {
3012     \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
3013   }
3014   \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
3015   \tl_put_right:Nn \l_tmpa_tl {\,\ellipses,}
3016   \tl_if_empty:NF \l__stex_variables_mid_tl {
3017     \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
3018     \tl_put_right:Nn \l_tmpa_tl {\,\ellipses,}
3019   }
3020   \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}
3021   \int_step_inline:nn \l__stex_variables_args_int {
3022     \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
3023   }
3024   \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
3025   \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
3026
3027
3028   \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl
3029
3030   \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}
3031
3032   \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}
3033
3034   \int_step_inline:nn \l__stex_variables_args_int {

```

```

3035 \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
3036   \_stex_term_math_arg:nnn{i##1}{0}{\exp_not:n{####}##1}
3037 }}
3038 }
3039
3040 \tl_set:Nx \l_tmpa_tl {
3041   \_stex_term_math_oma:nnnn { varseq://\l__stex_variables_name_str}{0}{
3042     \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
3043   }
3044 }
3045
3046 \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \stex_symbol_after_invokation_tl} }
3047
3048 \exp_args:Nno \use:nn {
3049   \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
3050   \cs_set:Npn {\int_use:N \l__stex_variables_args_int}{\l_tmpa_tl}
3051
3052   \stex_debug:nn{sequences}{New~Sequence:~
3053     \expandafter\meaning\csname stex_varseq_\l__stex_variables_name_str _cs\endcsname\\~\\
3054     \prop_to_keyval:N \l_tmpa_prop
3055   }
3056   \stex_if_do_html:T{\stex_annotate_invisible:nnn{varseq}{\l__stex_variables_name_str}{
3057     \tl_if_empty:NF \l__stex_variables_type_tl {
3058       \stex_annotate:nnn {type}{\}{\${\seqtype\l__stex_variables_type_tl$}
3059     }
3060     \stex_annotate:nnn {args}{\int_use:N \l__stex_variables_args_int}{\}
3061     \str_if_empty:NF \l__stex_variables_bind_str {
3062       \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{\}
3063     }
3064   }}
3065
3066   \prop_set_eq:cN {stex_varseq_\l__stex_variables_name_str _prop}\l_tmpa_prop
3067   \ignorespacesandpars
3068 }
3069
3070 </package>

```

## Chapter 30

# STEX -Terms Implementation

```
3071 <*package>
3072
3073 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
3074
3075 <@@=stex_terms>
3076
3077 Warnings and error messages
3078 \msg_new:nnn{stex}{error/nonotation}{
3079   Symbol~#1~invoked,~but~has~no~notation#2!
3080 }
3081 \msg_new:nnn{stex}{error/notationarg}{
3082   Error~in~parsing~notation~#1
3083 }
3084 \msg_new:nnn{stex}{error/noop}{
3085   Symbol~#1~has~no~operator~notation~for~notation~#2
3086 }
3087 \msg_new:nnn{stex}{error/notallowed}{
3088   Symbol~invocation~#1~not~allowed~in~notation~component~of~#2
3089 }
3090 \msg_new:nnn{stex}{error/doubleargument}{
3091   Argument~#1~of~symbol~#2~already~assigned
3092 }
3093 \msg_new:nnn{stex}{error/overarity}{
3094   Argument~#1~invalid~for~symbol~#2~with~arity~#3
3095 }
```

### 30.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
3095
3096
3097 \bool_new:N \l_stex_allow_semantic_bool
3098 \bool_set_true:N \l_stex_allow_semantic_bool
3099
```

```

3100 \cs_new_protected:Nn \stex_invoke_symbol:n {
3101   \bool_if:NTF \l_stex_allow_semantic_bool {
3102     \str_if_eq:eeF {
3103       \prop_item:cn {
3104         l_stex_symdecl_#1_prop
3105       }{ deprecate }
3106     }{}{
3107       \msg_warning:nxxx{stex}{warning/deprecated}{
3108         Symbol~#1
3109       }{
3110         \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
3111       }
3112     }
3113     \if_mode_math:
3114       \exp_after:wN \__stex_terms_invoke_math:n
3115     \else:
3116       \exp_after:wN \__stex_terms_invoke_text:n
3117     \fi: { #1 }
3118   }{
3119     \msg_error:nxxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
3120   }
3121 }
3122
3123 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
3124   \peek_charcode_remove:NTF ! {
3125     \__stex_terms_invoke_op_custom:nn {#1}
3126   }{
3127     \__stex_terms_invoke_custom:nn {#1}
3128   }
3129 }
3130
3131 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
3132   \peek_charcode_remove:NTF ! {
3133     % operator
3134     \peek_charcode_remove:NTF * {
3135       % custom op
3136       \__stex_terms_invoke_op_custom:nn {#1}
3137     }{
3138       % op notation
3139       \peek_charcode:NTF [ {
3140         \__stex_terms_invoke_op_notation:nw {#1}
3141       }{
3142         \__stex_terms_invoke_op_notation:nw {#1}[]
3143       }
3144     }
3145   }{
3146     \peek_charcode_remove:NTF * {
3147       \__stex_terms_invoke_custom:nn {#1}
3148       % custom
3149     }{
3150       % normal
3151       \peek_charcode:NTF [ {
3152         \__stex_terms_invoke_notation:nw {#1}
3153       }{

```

```

3154         \_stex_terms_invoke_notation:nw {#1}[]
3155     }
3156 }
3157 }
3158 }
3159
3160
3161 \cs_new_protected:Nn \_stex_terms_invoke_op_custom:nn {
3162     \exp_args:Nnx \use:nn {
3163         \def\comp{\_comp}
3164         \str_set:Nn \l_stex_current_symbol_str { #1 }
3165         \bool_set_false:N \l_stex_allow_semantic_bool
3166         \_stex_term_oms:nnn {#1}{#1 \c_hash_str CUSTOM-}{
3167             \comp{ #2 }
3168         }
3169     }{
3170         \_stex_reset:N \comp
3171         \_stex_reset:N \l_stex_current_symbol_str
3172         \bool_set_true:N \l_stex_allow_semantic_bool
3173     }
3174 }
3175
3176 \keys_define:nn { stex / terms } {
3177     % lang .tl_set_x:N = \l_stex_notation_lang_str ,
3178     variant .tl_set_x:N = \l_stex_notation_variant_str ,
3179     unknown .code:n = \str_set:Nx
3180         \l_stex_notation_variant_str \l_keys_key_str
3181 }
3182
3183 \cs_new_protected:Nn \_stex_terms_args:n {
3184     % \str_clear:N \l_stex_notation_lang_str
3185     \str_clear:N \l_stex_notation_variant_str
3186
3187     \keys_set:nn { stex / terms } { #1 }
3188 }
3189
3190 \cs_new_protected:Nn \stex_find_notation:nn {
3191     \_stex_terms_args:n { #2 }
3192     \seq_if_empty:cTF {
3193         l_stex_symdecl_ #1 _notations
3194     } {
3195         \msg_error:nxxx{stex}{error/nonotation}{#1}{s}
3196     } {
3197         \str_if_empty:NTF \l_stex_notation_variant_str {
3198             \seq_get_left:cN {l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
3199         }{
3200             \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
3201                 \l_stex_notation_variant_str
3202             }{
3203                 % \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3204             }{
3205                 \msg_error:nxxx{stex}{error/nonotation}{#1}{
3206                     ~\l_stex_notation_variant_str
3207                 }

```

```

3208     }
3209   }
3210 }
3211 }
3212
3213 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3214   \exp_args:Nnx \use:nn {
3215     \def\comp{\_comp}
3216     \str_set:Nn \l_stex_current_symbol_str { #1 }
3217     \stex_find_notation:nn { #1 }{ #2 }
3218     \bool_set_false:N \l_stex_allow_semantic_bool
3219     \cs_if_exist:cTF {
3220       stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3221     }{
3222       \_stex_term_oms:nnn { #1 }{
3223         #1 \c_hash_str \l_stex_notation_variant_str
3224       }{
3225         \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3226       }
3227     }{
3228       \int_compare:nNnTF {\prop_item:cn {\l_stex_symdecl_#1_prop}{arity}} = 0{
3229         \cs_if_exist:cTF {
3230           stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3231         }{
3232           \tl_set:Nx \stex_symbol_after_invokation_tl {
3233             \_stex_reset:N \comp
3234             \_stex_reset:N \stex_symbol_after_invokation_tl
3235             \_stex_reset:N \l_stex_current_symbol_str
3236             \bool_set_true:N \l_stex_allow_semantic_bool
3237           }
3238           \def\comp{\_comp}
3239           \str_set:Nn \l_stex_current_symbol_str { #1 }
3240           \bool_set_false:N \l_stex_allow_semantic_bool
3241           \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3242         }{
3243           \msg_error:nnxx{stex}{error/nonotation}{#1}{
3244             ~\l_stex_notation_variant_str
3245           }
3246         }
3247       }{
3248         \msg_error:nnxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3249       }
3250     }
3251   }{
3252     \_stex_reset:N \comp
3253     \_stex_reset:N \l_stex_current_symbol_str
3254     \bool_set_true:N \l_stex_allow_semantic_bool
3255   }
3256 }
3257
3258 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {
3259   \stex_find_notation:nn { #1 }{ #2 }
3260   \cs_if_exist:cTF {
3261     stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs

```

```

3262 }{
3263   \tl_set:Nx \stex_symbol_after_invokation_tl {
3264     \_stex_reset:N \comp
3265     \_stex_reset:N \stex_symbol_after_invokation_tl
3266     \_stex_reset:N \l_stex_current_symbol_str
3267     \bool_set_true:N \l_stex_allow_semantic_bool
3268   }
3269   \def\comp{\_comp}
3270   \str_set:Nn \l_stex_current_symbol_str { #1 }
3271   \bool_set_false:N \l_stex_allow_semantic_bool
3272   \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3273 }{
3274   \msg_error:nnxx{stex}{error/nonotation}{#1}{
3275     ~\l_stex_notation_variant_str
3276   }
3277 }
3278 }
3279
3280 \prop_new:N \l__stex_terms_custom_args_prop
3281
3282 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3283   \exp_args:Nnx \use:nn {
3284     \bool_set_false:N \l_stex_allow_semantic_bool
3285     \def\comp{\_comp}
3286     \str_set:Nn \l_stex_current_symbol_str { #1 }
3287     \prop_clear:N \l__stex_terms_custom_args_prop
3288     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3289     \prop_get:cnN {
3290       l_stex_symdecl_#1 _prop
3291     }{ args } \l_tmpa_str
3292     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3293     \tl_set:Nn \arg { \__stex_terms_arg: }
3294     \str_if_empty:NTF \l_tmpa_str {
3295       \_stex_term_oms:nnn {#1}{#1\c_hash_str CUSTOM-}{#2}
3296     }{
3297       \str_if_in:NnTF \l_tmpa_str b {
3298         \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3299       }{
3300         \str_if_in:NnTF \l_tmpa_str B {
3301           \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3302         }{
3303           \_stex_term_oma:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3304         }
3305       }
3306     }
3307     % TODO check that all arguments exist
3308   }{
3309     \_stex_reset:N \l_stex_current_symbol_str
3310     \_stex_reset:N \arg
3311     \_stex_reset:N \comp
3312     \_stex_reset:N \l__stex_terms_custom_args_prop
3313     \bool_set_true:N \l_stex_allow_semantic_bool
3314   }
3315 }

```



```

3316
3317 \NewDocumentCommand \__stex_terms_arg: { s 0{} m}{
3318   \tl_if_empty:nTF {#2}{
3319     \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
3320     \bool_set_true:N \l_tmpa_bool
3321     \bool_do_while:Nn \l_tmpa_bool {
3322       \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int
3323         \int_incr:N \l_tmpa_int
3324       }{
3325         \bool_set_false:N \l_tmpa_bool
3326       }
3327     }
3328   }{
3329     \int_set:Nn \l_tmpa_int { #2 }
3330   }
3331   \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
3332   \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3333     \msg_error:nnxxx{stex}{error/overarity}
3334     {\int_use:N \l_tmpa_int}
3335     {\l_stex_current_symbol_str}
3336     {\str_count:N \l_tmpa_str}
3337   }
3338   \str_set:Nx \l_tmpa_str {\str_item:Nn \l_tmpa_str \l_tmpa_int}
3339   \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
3340     \bool_lazy_any:nF {
3341       {\str_if_eq_p:Vn \l_tmpa_str {a}}
3342       {\str_if_eq_p:Vn \l_tmpa_str {B}}
3343     }{
3344       \msg_error:nnxx{stex}{error/doubleargument}
3345       {\int_use:N \l_tmpa_int}
3346       {\l_stex_current_symbol_str}
3347     }
3348   }
3349   \exp_args:NNx \prop_put:Nnn \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {#3}
3350   \bool_set_true:N \l_stex_allow_semantic_bool
3351   \IfBooleanTF#1{
3352     \stex_annotate_invisible:n { %TODO
3353       \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{#3}
3354     }
3355   }{ %TODO
3356     \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{#3}
3357   }
3358   \bool_set_false:N \l_stex_allow_semantic_bool
3359 }
3360
3361
3362 \cs_new_protected:Nn \_stex_term_arg:nn {
3363   \bool_set_true:N \l_stex_allow_semantic_bool
3364   \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3365   \bool_set_false:N \l_stex_allow_semantic_bool
3366 }
3367
3368 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
3369   \exp_args:Nnx \use:nn

```

```

3370 { \int_set:Nn \l__stex_terms_downprec { #2 }
3371   \stex_term_arg:nn { #1 }{ #3 }
3372 }
3373 { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3374 }

```

(End definition for `\stex_invoke_symbol:n`. This function is documented on page 62.)

`\stex_term_math_assoc_arg:nnnn`

```

3375 \cs_new_protected:Nn \stex_term_math_assoc_arg:nnnn {
3376   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3377   \tl_set:Nn \l_tmpb_tl {\stex_term_math_arg:nnn{#1}{#2}}
3378   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 } }{
3379     \expandafter\if\expandafter\relax\noexpand#3
3380     \expandafter\__stex_terms_math_assoc_arg_maybe_sequence:N\expandafter#3
3381     \else\expandafter\__stex_terms_math_assoc_arg_simple:nn
3382     \expandafter{\expandafter}\expandafter#3\fi
3383   }{
3384     \__stex_terms_math_assoc_arg_simple:nn{#1}{#3}
3385   }
3386 }
3387
3388 \cs_new_protected:Nn \__stex_terms_math_assoc_arg_maybe_sequence:N {
3389   \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
3390   \str_if_empty:NTF \l_tmpa_str {
3391     \exp_args:Nx \cs_if_eq:NNTF {
3392       \tl_head:N #1
3393     } \stex_invoke_sequence:n {
3394       \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
3395       \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
3396       \tl_set:Nx \l_tmpa_tl {\prop_item:cn {stex_varseq\l_tmpa_str _prop}{notation}}
3397       \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
3398       \tl_set:Nx \l_tmpa_tl {\exp_not:N \exp_not:n{
3399         \exp_not:n{\exp_args:Nnx \use:nn} {
3400           \exp_not:n {
3401             \def\comp{\_varcomp}
3402             \str_set:Nn \l_stex_current_symbol_str
3403             } {varseq://\l_tmpa_str}
3404             \exp_not:n{ ##1 }
3405           }{
3406             \exp_not:n {
3407               \stex_reset:N \comp
3408               \stex_reset:N \l_stex_current_symbol_str
3409             }
3410           }
3411         }}}
3412       \exp_args:Nno \use:nn {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
3413       \seq_reverse:N \l_tmpa_seq
3414       \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
3415       \seq_map_inline:Nn \l_tmpa_seq {
3416         \exp_args:NNo \exp_args:NNo \tl_set:N \l_tmpa_tl {
3417           \exp_args:Nno
3418           \l_tmpa_cs { ##1 } \l_tmpa_tl
3419         }

```

```

3420     }
3421     \tl_set:Nx \l_tmpa_tl {
3422         \stex_term_omv:nn {varseq://\l_tmpa_str}{
3423             \exp_args:No \exp_not:n \l_tmpa_tl
3424         }
3425     }
3426     \exp_args:No\l_tmpb_tl\l_tmpa_tl
3427 }{
3428     \__stex_terms_math_assoc_arg_simple:nn{} { #1 }
3429 }
3430 } {
3431     \__stex_terms_math_assoc_arg_simple:nn{} { #1 }
3432 }
3433 }
3434 }
3435
3436 \cs_new_protected:Nn \__stex_terms_math_assoc_arg_simple:nn {
3437     \clist_set:Nn \l_tmpa_clist{ #2 }
3438     \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3439         \tl_set:Nn \l_tmpa_tl { #2 }
3440     }{
3441         \clist_reverse:N \l_tmpa_clist
3442         \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3443         \tl_set:Nx \l_tmpa_tl { \stex_term_arg:nn{A#1}{
3444             \exp_args:No \exp_not:n \l_tmpa_tl
3445         }}
3446         \clist_map_inline:Nn \l_tmpa_clist {
3447             \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3448                 \exp_args:Nno
3449                 \l_tmpa_cs { \stex_term_arg:nn{A#1}{##1} } \l_tmpa_tl
3450             }
3451         }
3452     }
3453     \exp_args:No\l_tmpb_tl\l_tmpa_tl
3454 }

```

(End definition for `\stex_term_math_assoc_arg:nnnn`. This function is documented on page 62.)

## 30.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
3455 \tl_const:Nx \infprec {\int_use:N \c_max_int}
3456 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
3457 \int_new:N \l__stex_terms_downprec
3458 \int_set_eq:NN \l__stex_terms_downprec \infprec

```

(End definition for `\infprec`, `\neginfprec`, and `\l__stex_terms_downprec`. These variables are documented on page 63.)

Bracketing:

```

\l__stex_terms_left_bracket_str
\l__stex_terms_right_bracket_str
3459 \tl_set:Nn \l__stex_terms_left_bracket_str (
3460 \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

```

3461 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
3462   \bool_if:NTF \l__stex_terms_brackets_done_bool {
3463     \bool_set_false:N \l__stex_terms_brackets_done_bool
3464     #2
3465   } {
3466     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
3467       \bool_if:NTF \l__stex_inarray_bool { #2 } {
3468         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
3469         \dobrackets { #2 }
3470       }
3471     }{ #2 }
3472   }
3473 }

```

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

**`\dobrackets`**

```

3474 \bool_new:N \l__stex_terms_brackets_done_bool
3475 %\RequirePackage{scalareel}
3476 \cs_new_protected:Npn \dobrackets #1 {
3477   %\ThisStyle{\if D@m@switch
3478   %   \exp_args:Nnx \use:nn
3479   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
3480   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
3481   %   \else
3482   \exp_args:Nnx \use:nn
3483   {
3484     \bool_set_true:N \l__stex_terms_brackets_done_bool
3485     \int_set:Nn \l__stex_terms_downprec \infprec
3486     \l__stex_terms_left_bracket_str
3487     #1
3488   }
3489   {
3490     \bool_set_false:N \l__stex_terms_brackets_done_bool
3491     \l__stex_terms_right_bracket_str
3492     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3493   }
3494   %\fi}
3495 }

```

(End definition for `\dobrackets`. This function is documented on page 63.)

**`\withbrackets`**

```

3496 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
3497   \exp_args:Nnx \use:nn
3498   {
3499     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
3500     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
3501     #3
3502   }
3503 }

```

```

3504 \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
3505 {\l__stex_terms_left_bracket_str}
3506 \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
3507 {\l__stex_terms_right_bracket_str}
3508 }
3509 }

```

(End definition for `\withbrackets`. This function is documented on page 63.)

## `\STEXinvisible`

```

3510 \cs_new_protected:Npn \STEXinvisible #1 {
3511 \stex_annotate_invisible:n { #1 }
3512 }

```

(End definition for `\STEXinvisible`. This function is documented on page 63.)

OMDoc terms:

## `\_stex_term_math_oms:nnnn`

```

3513 \cs_new_protected:Nn \_stex_term_oms:nnn {
3514 \stex_annotate:nnn{ OMID }{ #2 }{
3515 #3
3516 }
3517 }
3518
3519 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
3520 \__stex_terms_maybe_brackets:nn { #3 }{
3521 \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3522 }
3523 }

```

(End definition for `\_stex_term_math_oms:nnnn`. This function is documented on page 62.)

## `\_stex_term_math_omv:nn`

```

3524 \cs_new_protected:Nn \_stex_term_omv:nn {
3525 \stex_annotate:nnn{ OMV }{ #1 }{
3526 #2
3527 }
3528 }

```

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

## `\_stex_term_math_oma:nnnn`

```

3529 \cs_new_protected:Nn \_stex_term_oma:nnn {
3530 \stex_annotate:nnn{ OMA }{ #2 }{
3531 #3
3532 }
3533 }
3534
3535 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
3536 \__stex_terms_maybe_brackets:nn { #3 }{
3537 \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3538 }
3539 }

```

(End definition for `\_stex_term_math_oma:nnnn`. This function is documented on page 62.)

`\_stex_term_math_omb:nnnn`

```
3540 \cs_new_protected:Nn \_stex_term_ombind:nnn {
3541   \stex_annotate:nnn{ OMBIND }{ #2 }{
3542     #3
3543   }
3544 }
3545
3546 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
3547   \__stex_terms_maybe_brackets:nn { #3 }{
3548     \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3549   }
3550 }
```

(End definition for `\_stex_term_math_omb:nnnn`. This function is documented on page 62.)

`\symref`

`\symname`

```
3551 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3552
3553 \keys_define:nn { stex / symname } {
3554   pre      .tl_set_x:N = \l__stex_terms_pre_tl ,
3555   post     .tl_set_x:N = \l__stex_terms_post_tl ,
3556   root     .tl_set_x:N = \l__stex_terms_root_tl
3557 }
3558
3559 \cs_new_protected:Nn \stex_symname_args:n {
3560   \tl_clear:N \l__stex_terms_post_tl
3561   \tl_clear:N \l__stex_terms_pre_tl
3562   \tl_clear:N \l__stex_terms_root_str
3563   \keys_set:nn { stex / symname } { #1 }
3564 }
3565
3566 \NewDocumentCommand \symref { m m }{
3567   \let\compemph_uri_prev:\compemph@uri
3568   \let\compemph@uri\symrefemph@uri
3569   \STEXsymbol{#1}!{ #2 }
3570   \let\compemph@uri\compemph_uri_prev:
3571 }
3572
3573 \NewDocumentCommand \synonym { 0{ } m m }{
3574   \stex_symname_args:n { #1 }
3575   \let\compemph_uri_prev:\compemph@uri
3576   \let\compemph@uri\symrefemph@uri
3577   % TODO
3578   \STEXsymbol{#2}!\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
3579   \let\compemph@uri\compemph_uri_prev:
3580 }
3581
3582 \NewDocumentCommand \symname { 0{ } m }{
3583   \stex_symname_args:n { #1 }
3584   \stex_get_symbol:n { #2 }
3585   \str_set:Nx \l_tmpa_str {
3586     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3587   }
3588   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
```

```

3589
3590 \let\compemph_uri_prev:\compemph@uri
3591 \let\compemph@uri\symrefemph@uri
3592 \exp_args:NNx \use:nn
3593 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!{
3594   \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
3595 } }
3596 \let\compemph@uri\compemph_uri_prev:
3597 }
3598
3599 \NewDocumentCommand \Symname { 0{ } m }{
3600   \stex_symname_args:n { #1 }
3601   \stex_get_symbol:n { #2 }
3602   \str_set:Nx \l_tmpa_str {
3603     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3604   }
3605   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3606   \let\compemph_uri_prev:\compemph@uri
3607   \let\compemph@uri\symrefemph@uri
3608   \exp_args:NNx \use:nn
3609   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!{
3610     \exp_after:wN \stex_capitalize:n \l_tmpa_str
3611     \l__stex_terms_post_tl
3612   } }
3613   \let\compemph@uri\compemph_uri_prev:
3614 }

```

(End definition for `\symref` and `\symname`. These functions are documented on page 62.)

### 30.3 Notation Components

```

3615 <@@=stex_notationcomps>

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

3616 \cs_new_protected:Npn \_comp #1 {
3617   \str_if_empty:NF \l_stex_current_symbol_str {
3618     \stex_html_backend:TF {
3619       \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
3620     }{
3621       \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3622     }
3623   }
3624 }
3625
3626 \cs_new_protected:Npn \_varcomp #1 {
3627   \str_if_empty:NF \l_stex_current_symbol_str {
3628     \stex_html_backend:TF {
3629       \stex_annotate:nnn { varcomp }{ \l_stex_current_symbol_str }{ #1 }
3630     }{
3631       \exp_args:Nnx \varemp@uri { #1 } { \l_stex_current_symbol_str }
3632     }
3633   }
3634 }
3635

```

```

3636 \def\comp{\_comp}
3637
3638 \cs_new_protected:Npn \compemph@uri #1 #2 {
3639   \compemph{ #1 }
3640 }
3641
3642
3643 \cs_new_protected:Npn \compemph #1 {
3644   #1
3645 }
3646
3647 \cs_new_protected:Npn \defemph@uri #1 #2 {
3648   \defemph{#1}
3649 }
3650
3651 \cs_new_protected:Npn \defemph #1 {
3652   \textbf{#1}
3653 }
3654
3655 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3656   \symrefemph{#1}
3657 }
3658
3659 \cs_new_protected:Npn \symrefemph #1 {
3660   \emph{#1}
3661 }
3662
3663 \cs_new_protected:Npn \varempemph@uri #1 #2 {
3664   \varempemph{#1}
3665 }
3666
3667 \cs_new_protected:Npn \varempemph #1 {
3668   #1
3669 }

```

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

## **\ellipses**

```

3670 \NewDocumentCommand \ellipses {} { \ldots }

```

(End definition for `\ellipses`. This function is documented on page 63.)

```

\parray
\prmatrix 3671 \bool_new:N \l_stex_inarray_bool
\parrayline 3672 \bool_set_false:N \l_stex_inarray_bool
\parraylineh 3673 \NewDocumentCommand \parray { m m } {
\parraycell 3674   \begingroup
3675   \bool_set_true:N \l_stex_inarray_bool
3676   \begin{array}{#1}
3677     #2
3678   \end{array}
3679   \endgroup
3680 }
3681
3682 \NewDocumentCommand \prmatrix { m } {

```



```

3683 \begingroup
3684 \bool_set_true:N \l_stex_inarray_bool
3685 \begin{matrix}
3686   #1
3687 \end{matrix}
3688 \endgroup
3689 }
3690
3691 \def \maybepline {
3692   \bool_if:NT \l_stex_inarray_bool {\hline}
3693 }
3694
3695 \def \parrayline #1 #2 {
3696   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3697 }
3698
3699 \def \pmrow #1 { \parrayline{}{ #1 } }
3700
3701 \def \parraylineh #1 #2 {
3702   #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3703 }
3704
3705 \def \parraycell #1 {
3706   #1 \bool_if:NT \l_stex_inarray_bool {&}
3707 }

```

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

## 30.4 Variables

```

3708 <@@=stex_variables>

```

`\stex_invoke_variable:n` Invokes a variable

```

3709 \cs_new_protected:Nn \stex_invoke_variable:n {
3710   \if_mode_math:
3711     \exp_after:wN \__stex_variables_invoke_math:n
3712   \else:
3713     \exp_after:wN \__stex_variables_invoke_text:n
3714   \fi: {#1}
3715 }
3716
3717 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
3718   %TODO
3719 }
3720
3721
3722 \cs_new_protected:Nn \__stex_variables_invoke_math:n {
3723   \peek_charcode_remove:NTF ! {
3724     \peek_charcode_remove:NTF ! {
3725       \peek_charcode:NTF [ {
3726         \__stex_variables_invoke_op_custom:nw
3727       }{
3728         % TODO throw error
3729       }

```

```

3730   }{
3731     \__stex_variables_invoke_op:n { #1 }
3732   }
3733 }{
3734   \peek_charcode_remove:NTF * {
3735     \__stex_variables_invoke_text:n { #1 }
3736   }{
3737     \__stex_variables_invoke_math_ii:n { #1 }
3738   }
3739 }
3740 }
3741
3742 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
3743   \cs_if_exist:cTF {
3744     stex_var_op_notation_ #1 _cs
3745   }{
3746     \exp_args:Nnx \use:nn {
3747       \def\comp{\_varcomp}
3748       \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3749       \_stex_term_omv:nn { var://#1 }{
3750         \use:c{stex_var_op_notation_ #1 _cs }
3751       }
3752     }{
3753       \_stex_reset:N \comp
3754       \_stex_reset:N \l_stex_current_symbol_str
3755     }
3756   }{
3757     \int_compare:nNnTF {\prop_item:cn {l_stex_variable_#1_prop}{arity}} = 0{
3758       \__stex_variables_invoke_math_ii:n {#1}
3759     }{
3760       \msg_error:nxxx{stex}{error/noop}{variable~#1}{}
3761     }
3762   }
3763 }
3764
3765 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n #1 {
3766   \cs_if_exist:cTF {
3767     stex_var_notation_#1_cs
3768   }{
3769     \tl_set:Nx \stex_symbol_after_invokation_tl {
3770       \_stex_reset:N \comp
3771       \_stex_reset:N \stex_symbol_after_invokation_tl
3772       \_stex_reset:N \l_stex_current_symbol_str
3773       \bool_set_true:N \l_stex_allow_semantic_bool
3774     }
3775     \def\comp{\_varcomp}
3776     \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3777     \bool_set_false:N \l_stex_allow_semantic_bool
3778     \use:c{stex_var_notation_#1_cs}
3779   }{
3780     \msg_error:nxxx{stex}{error/nonotation}{variable~#1}{s}
3781   }
3782 }

```

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

## 30.5 Sequences

```

3783 <@@=stex_sequences>
3784
3785 \cs_new_protected:Nn \stex_invoke_sequence:n {
3786   \peek_charcode_remove:NTF ! {
3787     \_stex_term_omv:nn {varseq://#1}{
3788       \exp_args:Nnx \use:nn {
3789         \def\comp{\_varcomp}
3790         \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3791         \prop_item:cn{stex_varseq_#1_prop}{notation}
3792       }{
3793         \_stex_reset:N \comp
3794         \_stex_reset:N \l_stex_current_symbol_str
3795       }
3796     }
3797   }{
3798     \bool_set_false:N \l_stex_allow_semantic_bool
3799     \def\comp{\_varcomp}
3800     \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3801     \tl_set:Nx \stex_symbol_after_invokation_tl {
3802       \_stex_reset:N \comp
3803       \_stex_reset:N \stex_symbol_after_invokation_tl
3804       \_stex_reset:N \l_stex_current_symbol_str
3805       \bool_set_true:N \l_stex_allow_semantic_bool
3806     }
3807     \use:c { stex_varseq_#1_cs }
3808   }
3809 }
3810 </package>

```

## Chapter 31

# STEX -Structural Features Implementation

```
3811 ⟨*package⟩
3812
3813 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3814
3815 Warnings and error messages
3816 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3817   Symbol~#1~can~not~be~assigned~in~copymodule~#2
3818 }
3819 \msg_new:nnn{stex}{error/interpretmodule/noddefinens}{
3820   Symbol~#1~not~assigned~in~interpretmodule~#2
3821 }
3822 \msg_new:nnn{stex}{error/unknownstructure}{
3823   No~structure~#1~found!
3824 }
3825
3826 \msg_new:nnn{stex}{error/unknownfield}{
3827   No~field~#1~in~instance~#2~found!~\#3
3828 }
3829
3830 \msg_new:nnn{stex}{error/keyval}{
3831   Invalid~key=value~pair~#1
3832 }
3833 \msg_new:nnn{stex}{error/instantiate/missing}{
3834   Assignments~missing~in~instantiate:~#1
3835 }
3836 \msg_new:nnn{stex}{error/incompatible}{
3837   Incompatible~signature:~#1~(#2)~and~#3~(#4)
3838 }
3839
```

## 31.1 Imports with modification

```

3840 <@@=stex_copymodule>
3841 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
3842   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3843     \tl_set:Nn \l_tmpa_tl { #1 }
3844     \__stex_copymodule_get_symbol_from_cs:
3845   }{
3846     % argument is a string
3847     % is it a command name?
3848     \cs_if_exist:cTF { #1 }{
3849       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3850       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3851       \str_if_empty:NNTF \l_tmpa_str {
3852         \exp_args:Nx \cs_if_eq:NNTF {
3853           \tl_head:N \l_tmpa_tl
3854         } \stex_invoke_symbol:n {
3855           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
3856         }{
3857           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3858         }
3859       } {
3860         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3861       }
3862     }{
3863       % argument is not a command name
3864       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3865       % \l_stex_all_symbols_seq
3866     }
3867   }
3868 }
3869
3870 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
3871   \str_set:Nn \l_tmpa_str { #1 }
3872   \bool_set_false:N \l_tmpa_bool
3873   \bool_if:NF \l_tmpa_bool {
3874     \tl_set:Nn \l_tmpa_tl {
3875       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
3876     }
3877     \str_set:Nn \l_tmpa_str { #1 }
3878     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3879     \seq_map_inline:Nn #2 {
3880       \str_set:Nn \l_tmpb_str { ##1 }
3881       \str_if_eq:eeT { \l_tmpa_str } {
3882         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3883       } {
3884         \seq_map_break:n {
3885           \tl_set:Nn \l_tmpa_tl {
3886             \str_set:Nn \l_stex_get_symbol_uri_str {
3887               ##1
3888             }
3889           }
3890         }
3891       }

```

```

3892     }
3893     \l_tmpa_tl
3894   }
3895 }
3896
3897 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
3898   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3899     { \tl_tail:N \l_tmpa_tl }
3900   \tl_if_single:NTF \l_tmpa_tl {
3901     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3902       \exp_after:wN \str_set:Nn \exp_after:wN
3903         \l_stex_get_symbol_uri_str \l_tmpa_tl
3904       \__stex_copymodule_get_symbol_check:n { #1 }
3905     }{
3906       % TODO
3907       % tail is not a single group
3908     }
3909   }{
3910     % TODO
3911     % tail is not a single group
3912   }
3913 }
3914
3915 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
3916   \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
3917     \msg_error:nxxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3918       :~\seq_use:Nn #1 {,~}
3919     }
3920   }
3921 }
3922
3923 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
3924   % import module
3925   \stex_import_module_uri:nn { #1 } { #2 }
3926   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3927   \stex_import_require_module:nnnn
3928     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3929     { \l_stex_import_path_str } { \l_stex_import_name_str }
3930
3931   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3932   \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
3933
3934   % fields
3935   \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
3936   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
3937     \seq_map_inline:cn {c_stex_module_##1_constants}{
3938       \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
3939         ##1 ? ####1
3940       }
3941     }
3942   }
3943
3944   % setup prop
3945   \seq_clear:N \l_tmpa_seq

```

```

3946 \exp_args:Nn \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3947   name      = \l_stex_current_copymodule_name_str ,
3948   module    = \l_stex_current_module_str ,
3949   from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3950   includes  = \l_tmpa_seq %,
3951 % fields    = \l_tmpa_seq
3952 }
3953 \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3954   as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3955 \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_seq {,
3956 \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
3957
3958 \stex_if_do_html:T {
3959   \begin{stex_annotate_env} {#4} {
3960     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3961   }
3962   \stex_annotate_invisible:nnn{domain}{\l_stex_import_ns_str ?\l_stex_import_name_str}{\}
3963 }
3964 }
3965
3966 \cs_new_protected:Nn \stex_copymodule_end:n {
3967   % apply to every field
3968   \def \l_tmpa_cs ##1 ##2 {#1}
3969
3970   \tl_clear:N \__stex_copymodule_module_tl
3971   \tl_clear:N \__stex_copymodule_exec_tl
3972
3973   %\prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3974   \seq_clear:N \__stex_copymodule_fields_seq
3975
3976   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
3977     \seq_map_inline:cn {c_stex_module_##1_constants}{
3978
3979       \tl_clear:N \__stex_copymodule_curr_symbol_tl % <- wrap in current symbol html
3980       \l_tmpa_cs{##1}{####1}
3981
3982       \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
3983         \str_set_eq:Nc \__stex_copymodule_curr_name_str {l__stex_copymodule_copymodule_##1?#
3984         \stex_if_do_html:T {
3985           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
3986             \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?###
3987           }
3988         }
3989       }{
3990         \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str /
3991       }
3992
3993       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
3994       \prop_put:Nnx \l_tmpa_prop { name } \__stex_copymodule_curr_name_str
3995       \prop_put:Nnx \l_tmpa_prop { module } \l_stex_current_module_str
3996
3997       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
3998         \stex_if_do_html:T {
3999           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {

```

```

4000         $\stex_annotate_invisible:nnn{definiens}{\exp_after:wN \exp_not:N\csname l__st
4001     }
4002 }
4003 \prop_put:Nnn \l_tmpa_prop { defined } { true }
4004 }
4005
4006 \stex_add_constant_to_current_module:n \__stex_copymodule_curr_name_str
4007 \tl_put_right:Nx \__stex_copymodule_module_tl {
4008     \seq_clear:c {l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_n
4009     \prop_set_from_keyval:cn {
4010         l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_name_str _prop
4011     }{
4012         \prop_to_keyval:N \l_tmpa_prop
4013     }
4014 }
4015
4016 \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
4017     \stex_if_do_html:T {
4018         \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4019             \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1
4020         }
4021     }
4022     \tl_put_right:Nx \__stex_copymodule_module_tl {
4023         \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
4024             \stex_invoke_symbol:n {
4025                 \l_stex_current_module_str ? \__stex_copymodule_curr_name_str
4026             }
4027         }
4028     }
4029 }
4030
4031 \seq_put_right:Nx \__stex_copymodule_fields_seq {\l_stex_current_module_str ? \__stex_
4032
4033 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4034     \stex_copy_notations:nn {\l_stex_current_module_str ? \__stex_copymodule_curr_name_s
4035 }
4036
4037 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4038     \stex_if_do_html:TF{
4039         \stex_annotate_invisible:nnn{assignment} {##1?####1} { \exp_after:wN \exp_not:n \e
4040     }{
4041         \exp_after:wN \exp_not:n \exp_after:wN {\__stex_copymodule_curr_symbol_tl}
4042     }
4043 }
4044 }
4045 }
4046
4047
4048 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \__stex_copymodule_fields_seq
4049 \tl_put_left:Nx \__stex_copymodule_module_tl {
4050     \prop_set_from_keyval:cn {
4051         l_stex_copymodule_ \l_stex_current_module_str? \l_stex_current_copymodule_name_str _pro
4052     }{
4053         \prop_to_keyval:N \l_stex_current_copymodule_prop

```



```

4054     }
4055 }
4056
4057 \seq_gput_right:cx{c_stex_module_\l_stex_current_module_str _copymodules}{
4058   \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4059 }
4060
4061 \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4062 \stex_debug:nn{copymodule}{result:\meaning \__stex_copymodule_module_tl}
4063 \stex_debug:nn{copymodule}{output:\meaning \__stex_copymodule_exec_tl}
4064
4065 \__stex_copymodule_exec_tl
4066 \stex_if_do_html:T {
4067   \end{stex_annotate_env}
4068 }
4069 }
4070
4071 \NewDocumentEnvironment {copymodule} { 0{} m m}{
4072   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ copymodule }
4073   \stex_deactivate_macro:Nn \symdecl {module~environments}
4074   \stex_deactivate_macro:Nn \symdef {module~environments}
4075   \stex_deactivate_macro:Nn \notation {module~environments}
4076   \stex_reactivate_macro:N \assign
4077   \stex_reactivate_macro:N \renamedekl
4078   \stex_reactivate_macro:N \donotcopy
4079   \stex_smsmode_do:
4080 }{
4081   \stex_copymodule_end:n {}
4082 }
4083
4084 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
4085   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ interpretmodule }
4086   \stex_deactivate_macro:Nn \symdecl {module~environments}
4087   \stex_deactivate_macro:Nn \symdef {module~environments}
4088   \stex_deactivate_macro:Nn \notation {module~environments}
4089   \stex_reactivate_macro:N \assign
4090   \stex_reactivate_macro:N \renamedekl
4091   \stex_reactivate_macro:N \donotcopy
4092   \stex_smsmode_do:
4093 }{
4094   \stex_copymodule_end:n {
4095     \tl_if_exist:cF {
4096       l__stex_copymodule_copymodule_##1?##2_def_tl
4097     }{
4098       \str_if_eq:eeF {
4099         \prop_item:cn{
4100           l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
4101         }{ true }{
4102           \msg_error:nxxx{stex}{error/interpretmodule/nodéfiniens}{
4103             ##1?##2
4104           }{\l_stex_current_copymodule_name_str}
4105         }
4106       }
4107     }

```

```

4108 }
4109
4110 \iffalse \begin{stex_annotate_env} \fi
4111 \NewDocumentEnvironment {realization} { 0 } { m } {
4112   \stex_copymodule_start:nnnn { #1 } { #2 } { #2 } { realize }
4113   \stex_deactivate_macro:Nn \symdecl {module~environments}
4114   \stex_deactivate_macro:Nn \symdef {module~environments}
4115   \stex_deactivate_macro:Nn \notation {module~environments}
4116   \stex_reactivate_macro:N \donotcopy
4117   \stex_reactivate_macro:N \assign
4118   \stex_smsmode_do:
4119 } {
4120   \stex_import_module_uri:nn { #1 } { #2 }
4121   \tl_clear:N \__stex_copymodule_exec_tl
4122   \tl_set:Nx \__stex_copymodule_module_tl {
4123     \stex_import_require_module:nnnn
4124     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4125     { \l_stex_import_path_str } { \l_stex_import_name_str }
4126   }
4127
4128   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4129     \seq_map_inline:cn {c_stex_module_##1_constants}{
4130       \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str / #1 }
4131       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?###1_def_tl}{
4132         \stex_if_do_html:T {
4133           \tl_put_right:Nx \__stex_copymodule_exec_tl {
4134             \stex_annotate_invisible:nnn{assignment} {##1?###1} {
4135               $\stex_annotate_invisible:nnn{definien}{}{\exp_after:wN \exp_not:N\csname l__
4136             }
4137           }
4138         }
4139         \tl_put_right:Nx \__stex_copymodule_module_tl {
4140           \prop_put:cnn {l_stex_symdecl_##1?###1_prop}{ defined }{ true }
4141         }
4142       }
4143     }
4144
4145     \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4146
4147     \__stex_copymodule_exec_tl
4148     \stex_if_do_html:T {\end{stex_annotate_env}}
4149   }
4150
4151   \NewDocumentCommand \donotcopy { m } {
4152     \str_clear:N \l_stex_import_name_str
4153     \str_set:Nn \l_tmpa_str { #1 }
4154     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4155     \seq_map_inline:Nn \l_stex_all_modules_seq {
4156       \str_set:Nn \l_tmpb_str { ##1 }
4157       \str_if_eq:eeT { \l_tmpa_str } {
4158         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4159       } {
4160         \seq_map_break:n {
4161           \stex_if_do_html:T {

```

```

4162         \stex_if_smsmode:F {
4163             \stex_annotate_invisible:nnn{donotcopy}{##1}{
4164                 \stex_annotate:nnn{domain}{##1}{}}
4165         }
4166     }
4167 }
4168 \str_set_eq:NN \l_stex_import_name_str \l_tmpb_str
4169 }
4170 }
4171 \seq_map_inline:cn {c_stex_module_##1_copymodules}{
4172     \str_set:Nn \l_tmpb_str { ####1 }
4173     \str_if_eq:eeT { \l_tmpa_str } {
4174         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4175     } {
4176         \seq_map_break:n {\seq_map_break:n {
4177             \stex_if_do_html:T {
4178                 \stex_if_smsmode:F {
4179                     \stex_annotate_invisible:nnn{donotcopy}{####1}{
4180                         \stex_annotate:nnn{domain}{
4181                             \prop_item:cn {l_stex_copymodule_ ####1 _prop}{module}
4182                         }{}
4183                     }
4184                 }
4185             }
4186             \str_set:Nx \l_stex_import_name_str {
4187                 \prop_item:cn {l_stex_copymodule_ ####1 _prop}{module}
4188             }
4189         }}
4190     }
4191 }
4192 }
4193 \str_if_empty:NTF \l_stex_import_name_str {
4194     % TODO throw error
4195 }{
4196     \stex_collect_imports:n {\l_stex_import_name_str }
4197     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4198         \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
4199         \seq_map_inline:cn {c_stex_module_##1_constants}{
4200             \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? ####1 }
4201             \bool_lazy_any:nT {
4202                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_name_str}}
4203                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_macroname_str}}
4204                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_def_tl}}
4205             }{
4206                 % TODO throw error
4207             }
4208         }
4209     }
4210     \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
4211     \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_name_str }
4212     \prop_put:Nno \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
4213 }
4214 \stex_smsmode_do:
4215 }

```

```

4216
4217 \NewDocumentCommand \assign { m m }{
4218   \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
4219   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
4220   \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_def_tl}{#2}
4221   \stex_smsmode_do:
4222 }
4223
4224 \keys_define:nn { stex / renamedekl } {
4225   name          .str_set_x:N = \l_stex_renamedekl_name_str
4226 }
4227 \cs_new_protected:Nn \__stex_copymodule_renamedekl_args:n {
4228   \str_clear:N \l_stex_renamedekl_name_str
4229   \keys_set:nn { stex / renamedekl } { #1 }
4230 }
4231
4232 \NewDocumentCommand \renamedekl { O{} m m }{
4233   \__stex_copymodule_renamedekl_args:n { #1 }
4234   \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
4235   \stex_debug:nn{renamedekl}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
4236   \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_macroname_str}{#3}
4237   \str_if_empty:NTF \l_stex_renamedekl_name_str {
4238     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4239       \l_stex_get_symbol_uri_str
4240     } }
4241   } {
4242     \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_name_str}{\l_stex
4243       \stex_debug:nn{renamedekl}{@~\l_stex_current_module_str ? \l_stex_renamedekl_name_str}
4244       \prop_set_eq:cc {l_stex_symdecl_
4245         \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4246         _prop
4247       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str_prop}
4248       \seq_set_eq:cc {l_stex_symdecl_
4249         \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4250         _notations
4251       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str_notations}
4252       \prop_put:cnx {l_stex_symdecl_
4253         \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4254         _prop
4255       }{ name }{ \l_stex_renamedekl_name_str }
4256       \prop_put:cnx {l_stex_symdecl_
4257         \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4258         _prop
4259       }{ module }{ \l_stex_current_module_str }
4260       \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
4261         \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4262       }
4263       \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4264         \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4265       } }
4266     }
4267   \stex_smsmode_do:
4268 }
4269

```

```

4270 \stex_deactivate_macro:Nn \assign {copymodules}
4271 \stex_deactivate_macro:Nn \renamedekl {copymodules}
4272 \stex_deactivate_macro:Nn \donotcopy {copymodules}
4273
4274

```

## 31.2 The feature environment

structural@feature

```

4275 <@@=stex_features>
4276
4277 \NewDocumentEnvironment{structural_feature_module}{ m m m }{
4278   \stex_if_in_module:F {
4279     \msg_set:nnn{stex}{error/nomodule}{
4280       Structural~Feature~has~to~occur~in~a~module:\\
4281       Feature~#2~of~type~#1\\
4282       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
4283     }
4284     \msg_error:nn{stex}{error/nomodule}
4285   }
4286
4287   \str_set_eq:NN \l_stex_feature_parent_str \l_stex_current_module_str
4288
4289   \stex_module_setup:nn{meta=NONE}{#2 - #1}
4290
4291   \stex_if_do_html:T {
4292     \begin{stex_annotate_env}{ feature:#1 }{\l_stex_feature_parent_str ? #2 - #1}
4293     \stex_annotate_invisible:nnn{header}{\{ #3 }
4294   }
4295   }{
4296     \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
4297     \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
4298     \stex_debug:nn{features}{
4299       Feature: \l_stex_last_feature_str
4300     }
4301     \stex_if_do_html:T {
4302       \end{stex_annotate_env}
4303     }
4304   }

```

## 31.3 Structure

structure

```

4305 <@@=stex_structures>
4306 \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
4307   \prop_if_exist:cF {c_stex_module_ \l_stex_current_module_str _structures}{
4308     \prop_new:c {c_stex_module_ \l_stex_current_module_str _structures}
4309   }
4310   \prop_gput:cxn{c_stex_module_ \l_stex_current_module_str _structures}
4311   {#1}{#2}
4312 }
4313

```

```

4314 \keys_define:nn { stex / features / structure } {
4315   name          .str_set_x:N = \l__stex_structures_name_str ,
4316 }
4317
4318 \cs_new_protected:Nn \__stex_structures_structure_args:n {
4319   \str_clear:N \l__stex_structures_name_str
4320   \keys_set:nn { stex / features / structure } { #1 }
4321 }
4322
4323 \NewDocumentEnvironment{mathstructure}{m O{}}{
4324   \__stex_structures_structure_args:n { #2 }
4325   \str_if_empty:NT \l__stex_structures_name_str {
4326     \str_set:Nx \l__stex_structures_name_str { #1 }
4327   }
4328   \stex_suppress_html:n {
4329     \exp_args:Nx \stex_symdecl_do:nn {
4330       name = \l__stex_structures_name_str ,
4331       def  = {\STEXsymbol{module-type}}{
4332         \stex_term_math_oms:nnnn {
4333           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4334             { ns } ?
4335           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4336             { name } / \l__stex_structures_name_str - structure
4337         }{}{0}{}
4338       }}
4339     }{ #1 }
4340   }
4341   \exp_args:Nnnx
4342   \begin{structural_feature_module}{ structure }
4343     { \l__stex_structures_name_str }{}
4344   \stex_smsmode_do:
4345 }{
4346   \end{structural_feature_module}
4347   \stex_reset_up_to_module:n \l_stex_last_feature_str
4348   \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
4349   \seq_clear:N \l_tmpa_seq
4350   \seq_map_inline:Nn \l_stex_collect_imports_seq {
4351     \seq_map_inline:cn{c_stex_module_##1_constants}{
4352       \seq_put_right:Nn \l_tmpa_seq { ##1 ? ###1 }
4353     }
4354   }
4355   \exp_args:Nnno
4356   \prop_gput:cnn {c_stex_module_\l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
4357   \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
4358   \stex_add_structure_to_current_module:nn
4359     \l__stex_structures_name_str
4360     \l_stex_last_feature_str
4361
4362   \stex_execute_in_module:x {
4363     \tl_set:cn { #1 }{
4364       \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures
4365     }
4366   }
4367 }

```

```

4368
4369 \cs_new:Nn \stex_invoke_structure:nn {
4370   \stex_invoke_symbol:n { #1?#2 }
4371 }
4372
4373 \cs_new_protected:Nn \stex_get_structure:n {
4374   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4375     \tl_set:Nn \l_tmpa_tl { #1 }
4376     \__stex_structures_get_from_cs:
4377   }{
4378     \cs_if_exist:cTF { #1 }{
4379       \cs_set_eq:Nc \l_tmpa_cs { #1 }
4380       \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
4381       \str_if_empty:NNTF \l_tmpa_str {
4382         \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs} \stex_invoke_structure:nn {
4383           \__stex_structures_get_from_cs:
4384         }{
4385           \__stex_structures_get_from_string:n { #1 }
4386         }
4387       }{
4388         \__stex_structures_get_from_string:n { #1 }
4389       }
4390     }{
4391       \__stex_structures_get_from_string:n { #1 }
4392     }
4393   }
4394 }
4395
4396 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
4397   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4398     { \tl_tail:N \l_tmpa_tl }
4399   \str_set:Nx \l_tmpa_str {
4400     \exp_after:wN \use_i:nn \l_tmpa_tl
4401   }
4402   \str_set:Nx \l_tmpb_str {
4403     \exp_after:wN \use_ii:nn \l_tmpa_tl
4404   }
4405   \str_set:Nx \l_stex_get_structure_str {
4406     \l_tmpa_str ? \l_tmpb_str
4407   }
4408   \str_set:Nx \l_stex_get_structure_module_str {
4409     \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
4410   }
4411 }
4412
4413 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
4414   \tl_set:Nn \l_tmpa_tl {
4415     \msg_error:nnn{stex}{error/unknownstructure}{#1}
4416   }
4417   \str_set:Nn \l_tmpa_str { #1 }
4418   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4419
4420   \seq_map_inline:Nn \l_stex_all_modules_seq {
4421     \prop_if_exist:cT {c_stex_module_##1_structures} {

```

```

4422 \prop_map_inline:cn {c_stex_module_##1_structures} {
4423   \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
4424     \prop_map_break:n{\seq_map_break:n{
4425       \tl_set:Nn \l_tmpa_tl {
4426         \str_set:Nn \l_stex_get_structure_str {##1?####1}
4427         \str_set:Nn \l_stex_get_structure_module_str {####2}
4428       }
4429     }}
4430   }
4431 }
4432 }
4433 }
4434 \l_tmpa_tl
4435 }

```

**\instantiate**

```

4436
4437 \keys_define:nn { stex / instantiate } {
4438   name          .str_set_x:N = \l__stex_structures_name_str
4439 }
4440 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
4441   \str_clear:N \l__stex_structures_name_str
4442   \keys_set:nn { stex / instantiate } { #1 }
4443 }
4444
4445 \NewDocumentCommand \instantiate {m O{} m m m}{
4446   \begingroup
4447     \stex_get_structure:n {#4}
4448     \__stex_structures_instantiate_args:n { #2 }
4449     \str_if_empty:NT \l__stex_structures_name_str {
4450       \str_set:Nn \l__stex_structures_name_str { #1 }
4451     }
4452     \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4453     \seq_clear:N \l__stex_structures_fields_seq
4454     \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4455     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4456       \seq_map_inline:cn {c_stex_module_##1_constants}{
4457         \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4458       }
4459     }
4460
4461     \tl_if_empty:nF{#3}{
4462       \seq_set_split:Nnn \l_tmpa_seq , {#3}
4463       \prop_clear:N \l_tmpa_prop
4464       \seq_map_inline:Nn \l_tmpa_seq {
4465         \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4466         \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4467           \msg_error:nnn{stex}{error/keyval}{##1}
4468         }
4469         \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
4470         \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4471         \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
4472         \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}
4473         \exp_args:Nxx \str_if_eq:nnF

```



```

4474         {\prop_item:cn{l_stex_symdecl\_l\_stex_structures_dom_str\_prop}{args}}
4475         {\prop_item:cn{l_stex_symdecl\_l\_stex_get_symbol_uri_str\_prop}{args}}{
4476         \msg_error:nnxxxx{stex}{error/incompatible}
4477         {l\_stex_structures_dom_str}
4478         {\prop_item:cn{l_stex_symdecl\_l\_stex_structures_dom_str\_prop}{args}}
4479         {\l_stex_get_symbol_uri_str}
4480         {\prop_item:cn{l_stex_symdecl\_l\_stex_get_symbol_uri_str\_prop}{args}}
4481     }
4482     \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
4483 }
4484 }
4485
4486 \seq_map_inline:Nn \l\_stex_structures_fields_seq {
4487     \str_set:Nx \l_tmpa_str {field:l\_stex_structures_name_str . \prop_item:cn {l_stex_sy
4488     \stex_debug:nn{instantiate}{Field~\l_tmpa_str :~##1}
4489
4490     \stex_add_constant_to_current_module:n {\l_tmpa_str}
4491     \stex_execute_in_module:x {
4492         \prop_set_from_keyval:cn { l_stex_symdecl\_l\_stex_current_module_str?\l_tmpa_str_p
4493         name = \l_tmpa_str ,
4494         args = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4495         arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4496         assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4497     }
4498     \seq_clear:c {l_stex_symdecl\_l\_stex_current_module_str?\l_tmpa_str\_notations}
4499 }
4500
4501 \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4502     \stex_find_notation:nn{##1}{}
4503     \stex_execute_in_module:x {
4504         \seq_put_right:cn {l_stex_symdecl\_l\_stex_current_module_str?\l_tmpa_str\_notation
4505     }
4506
4507     \stex_copy_control_sequence:ccN
4508     {stex_notation\_l\_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_notation_
4509     {stex_notation_##1\c_hash_str \l_stex_notation_variant_str_cs}
4510     \l_tmpa_tl
4511     \exp_args:No \stex_execute_in_module:n \l_tmpa_tl
4512
4513
4514     \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str_cs}{
4515         \tl_set_eq:Nc \l_tmpa_cs {stex_op_notation_##1\c_hash_str \l_stex_notation_variant
4516         \stex_execute_in_module:x {
4517             \tl_set:cn
4518             {stex_op_notation\_l\_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_nota
4519             { \exp_args:No \exp_not:n \l_tmpa_cs}
4520         }
4521     }
4522 }
4523 }
4524
4525 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\l_stex_cur
4526 }
4527

```

```

4528 \stex_execute_in_module:x {
4529   \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
4530   domain = \l_stex_get_structure_module_str ,
4531   \prop_to_keyval:N \l_tmpa_prop
4532 }
4533 \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
4534 }
4535 \stex_debug:nn{instantiate}{
4536   Instance~\l_stex_current_module_str?\l__stex_structures_name_str \
4537   \prop_to_keyval:N \l_tmpa_prop
4538 }
4539 \exp_args:Nxx \stex_symdecl_do:nn {
4540   type={\STEXsymbol{module-type}}{
4541     \stex_term_math_oms:nnnn {
4542       \l_stex_get_structure_module_str
4543     }{}{0}{}
4544   }}
4545 }{\l__stex_structures_name_str}
4546 % {
4547   \str_set:Nx \l_stex_get_symbol_uri_str {\l_stex_current_module_str?\l__stex_structures
4548   \tl_set:Nn \l_stex_notation_after_do_tl {\__stex_notation_final:}
4549   \stex_notation_do:nnnnn{}{}{0}{}{\comp{#5}}
4550 % }
4551 %\exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}
4552 \endgroup
4553 \stex_smsmode_do:\ignorespacesandpars
4554 }
4555
4556 \cs_new_protected:Nn \stex_symbol_or_var:n {
4557   \cs_if_exist:cTF{#1}{
4558     \cs_set_eq:Nc \l_tmpa_tl { #1 }
4559     \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4560     \str_if_empty:NTF \l_tmpa_str {
4561       \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
4562       \stex_invoke_variable:n {
4563         \bool_set_true:N \l_stex_symbol_or_var_bool
4564         \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4565         \str_set:Nx \l_stex_get_symbol_uri_str {
4566           \exp_after:wN \use:n \l_tmpa_tl
4567         }
4568       }{
4569         \bool_set_false:N \l_stex_symbol_or_var_bool
4570         \stex_get_symbol:n{#1}
4571       }
4572     }{
4573       \__stex_structures_symbolorvar_from_string:n{ #1 }
4574     }
4575   }{
4576     \__stex_structures_symbolorvar_from_string:n{ #1 }
4577   }
4578 }
4579
4580 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
4581   \prop_if_exist:cTF {l_stex_variable_#1 _prop}{

```

```

4582     \bool_set_true:N \l_stex_symbol_or_var_bool
4583     \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
4584   }{
4585     \bool_set_false:N \l_stex_symbol_or_var_bool
4586     \stex_get_symbol:n{#1}
4587   }
4588 }
4589
4590 \keys_define:nn { stex / varinstantiate } {
4591   name      .str_set_x:N = \l__stex_structures_name_str,
4592   bind      .choices:nn =
4593     {forall,exists}
4594     {\str_set:Nx \l__stex_structures_bind_str {\l_keys_choice_tl}}
4595 }
4596
4597 \cs_new_protected:Nn \__stex_structures_varinstantiate_args:n {
4598   \str_clear:N \l__stex_structures_name_str
4599   \str_clear:N \l__stex_structures_bind_str
4600   \keys_set:nn { stex / varinstantiate } { #1 }
4601 }
4602
4603 \NewDocumentCommand \varinstantiate {m O{} m m m}{
4604   \begin{group}
4605     \stex_get_structure:n {#4}
4606     \__stex_structures_varinstantiate_args:n { #2 }
4607     \str_if_empty:NT \l__stex_structures_name_str {
4608       \str_set:Nn \l__stex_structures_name_str { #1 }
4609     }
4610     \stex_if_do_html:TF{
4611       \stex_annotate:nnn{varinstance}{\l__stex_structures_name_str}
4612     }{\use:n}
4613     {
4614       \stex_if_do_html:T{
4615         \stex_annotate_invisible:nnn{domain}{\l_stex_get_structure_module_str}{}}
4616       }
4617       \seq_clear:N \l__stex_structures_fields_seq
4618       \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4619       \seq_map_inline:Nn \l_stex_collect_imports_seq {
4620         \seq_map_inline:cn {c_stex_module_##1_constants}{
4621           \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4622         }
4623       }
4624       \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4625       \prop_clear:N \l_tmpa_prop
4626       \tl_if_empty:nF {#3} {
4627         \seq_set_split:Nnn \l_tmpa_seq , {#3}
4628         \seq_map_inline:Nn \l_tmpa_seq {
4629           \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4630           \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4631             \msg_error:nnn{stex}{error/keyval}{##1}
4632           }
4633           \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_stru
4634           \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4635           \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol

```

```

4636 \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
4637 \stex_if_do_html:T{
4638   \stex_annotate:nnn{assign}{\l__stex_structures_dom_str,\l_stex_get_symbol_uri_str}
4639 }
4640 \bool_if:NTF \l_stex_symbol_or_var_bool {
4641   \exp_args:Nxx \str_if_eq:nnF
4642     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4643     {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}{
4644     \msg_error:nnxxxx{stex}{error/incompatible}
4645     {\l__stex_structures_dom_str}
4646     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4647     {\l_stex_get_symbol_uri_str}
4648     {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}
4649   }
4650   \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n}
4651 }{
4652   \exp_args:Nxx \str_if_eq:nnF
4653     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4654     {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
4655     \msg_error:nnxxxx{stex}{error/incompatible}
4656     {\l__stex_structures_dom_str}
4657     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4658     {\l_stex_get_symbol_uri_str}
4659     {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
4660   }
4661   \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n}
4662 }
4663 }
4664 }
4665 \tl_gclear:N \g__stex_structures_aftergroup_tl
4666 \seq_map_inline:Nn \l__stex_structures_fields_seq {
4667   \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdecl_\l__stex_structures_fields_seq}{args}}
4668   \stex_debug:nn{varinstantiate}{Field~\l_tmpa_str :~##1}
4669   \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4670     \stex_find_notation:nn{##1}{
4671       \cs_gset_eq:cc{g__stex_structures_tmpa_\l_tmpa_str _cs}
4672         {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4673       \stex_debug:nn{varinstantiate}{Notation:~\cs_meaning:c{g__stex_structures_tmpa_\l_tmpa_str _cs}}
4674       \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
4675         \cs_gset_eq:cc {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
4676           {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4677         \stex_debug:nn{varinstantiate}{Operator~Notation:~\cs_meaning:c{g__stex_structures_tmpa_op_\l_tmpa_str _cs}}
4678       }
4679     }
4680   }
4681   \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4682     \prop_set_from_keyval:cn { l_stex_variable_ \l_tmpa_str _prop}{
4683       name = \l_tmpa_str ,
4684       args = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4685       arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4686       assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4687     }
4688     \cs_set_eq:cc {stex_var_notation_\l_tmpa_str _cs}
4689     {g__stex_structures_tmpa_\l_tmpa_str _cs}

```

```

4690         \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str_cs}
4691         {g__stex_structures_tmpa_op_\l_tmpa_str_cs}
4692     }
4693     \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_inv
4694 }
4695     \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4696         \prop_set_from_keyval:cn {l_stex_varinstance_\l__stex_structures_name_str_prop }{
4697             domain = \l_stex_get_structure_module_str ,
4698             \prop_to_keyval:N \l_tmpa_prop
4699         }
4700         \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
4701         \tl_set:cn {l_stex_varinstance_\l__stex_structures_name_str_op_tl}{
4702             \exp_args:Nnx \exp_not:N \use:nn {
4703                 \str_set:Nn \exp_not:N \l_stex_current_symbol_str {var://\l__stex_structures_nam
4704                 \_stex_term_omv:nn {var://\l__stex_structures_name_str}{
4705                     \exp_not:n{
4706                         \_varcomp{#5}
4707                     }
4708                 }
4709             }{
4710                 \exp_not:n{\_stex_reset:N \l_stex_current_symbol_str}
4711             }
4712         }
4713     }
4714 }
4715 \stex_debug:nn{varinstantiate}{\expandafter\detokenize\expandafter{\g__stex_structures_a
4716 \aftergroup\g__stex_structures_aftergroup_tl
4717 \endgroup
4718 \stex_smsmode_do:\ignorespacesandpars
4719 }
4720
4721 \cs_new_protected:Nn \stex_invoke_instance:n {
4722     \peek_charcode_remove:NTF ! {
4723         \stex_invoke_symbol:n{#1}
4724     }{
4725         \_stex_invoke_instance:nn {#1}
4726     }
4727 }
4728
4729
4730 \cs_new_protected:Nn \stex_invoke_varinstance:n {
4731     \peek_charcode_remove:NTF ! {
4732         \exp_args:Nnx \use:nn {
4733             \def\comp{\_varcomp}
4734             \use:c{l_stex_varinstance_#1_op_tl}
4735         }{
4736             \_stex_reset:N \comp
4737         }
4738     }{
4739         \_stex_invoke_varinstance:nn {#1}
4740     }
4741 }
4742
4743 \cs_new_protected:Nn \_stex_invoke_instance:nn {

```

```

4744 \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
4745   \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
4746 }{
4747   \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
4748   \msg_error:nnxxx{stex}{error/unknownfield}{#2}{#1}{
4749     \prop_to_keyval:N \l_tmpa_prop
4750   }
4751 }
4752 }
4753
4754 \cs_new_protected:Nn \stex_invoke_varinstance:nn {
4755   \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
4756     \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
4757     \l_tmpa_tl
4758   }{
4759     \msg_error:nnnnn{stex}{error/unknownfield}{#2}{#1}{}
4760   }
4761 }

```

(End definition for `\instantiate`. This function is documented on page 31.)

`\stex_invoke_structure:nnn`

```

4762 % #1: URI of the instance
4763 % #2: URI of the instantiated module
4764 \cs_new_protected:Nn \stex_invoke_structure:nnn {
4765   \tl_if_empty:nTF{ #3 }{
4766     \prop_set_eq:Nc \l__stex_structures_structure_prop {
4767       c_stex_feature_ #2 _prop
4768     }
4769     \tl_clear:N \l_tmpa_tl
4770     \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
4771     \seq_map_inline:Nn \l_tmpa_seq {
4772       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
4773       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
4774       \cs_if_exist:cT {
4775         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
4776       }{
4777         \tl_if_empty:NF \l_tmpa_tl {
4778           \tl_put_right:Nn \l_tmpa_tl {,}
4779         }
4780         \tl_put_right:Nx \l_tmpa_tl {
4781           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
4782         }
4783       }
4784     }
4785     \exp_args:No \mathstruct \l_tmpa_tl
4786   }{
4787     \stex_invoke_symbol:n{#1/#3}
4788   }
4789 }

```

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

```

4790 </package>

```

## Chapter 32

# STEX -Statements Implementation

```
4791 <*package>
4792
4793 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4794
4795 <@@=stex_statements>
4796
4797 Warnings and error messages
4798
4799
4800 \titleemph
4801
4802 \def\titleemph#1{\textbf{#1}}
4803
4804 (End definition for \titleemph. This function is documented on page ??.)
```

### 32.1 Definitions

#### definiendum

```
4798 \keys_define:nn {stex / definiendum }{
4799   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
4800   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
4801   root     .str_set_x:N    = \l__stex_statements_definiendum_root_str,
4802   gfa      .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
4803 }
4804 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
4805   \str_clear:N \l__stex_statements_definiendum_root_str
4806   \tl_clear:N \l__stex_statements_definiendum_post_tl
4807   \str_clear:N \l__stex_statements_definiendum_gfa_str
4808   \keys_set:nn { stex / definiendum }{ #1 }
4809 }
4810 \NewDocumentCommand \definiendum { O{} m m } {
4811   \__stex_statements_definiendum_args:n { #1 }
4812   \stex_get_symbol:n { #2 }
4813   \stex_ref_new_sym_target:n \l__stex_get_symbol_uri_str
4814   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
4815     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```

4816     \tl_set:Nn \l_tmpa_tl { #3 }
4817   } {
4818     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
4819     \tl_set:Nn \l_tmpa_tl {
4820       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
4821     }
4822   }
4823 } {
4824   \tl_set:Nn \l_tmpa_tl { #3 }
4825 }
4826
4827 % TODO root
4828 \stex_html_backend:TF {
4829   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
4830 } {
4831   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
4832 }
4833 }
4834 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

(End definition for definiendum. This function is documented on page 40.)

#### definame

```

4835
4836 \NewDocumentCommand \definame { 0{ } m } {
4837   \__stex_statements_definiendum_args:n { #1 }
4838   % TODO: root
4839   \stex_get_symbol:n { #2 }
4840   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4841   \str_set:Nx \l_tmpa_str {
4842     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4843   }
4844   \str_replace_all:Nnn \l_tmpa_str {-} {~}
4845   \stex_html_backend:TF {
4846     \stex_if_do_html:T {
4847       \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4848         \l_tmpa_str\l__stex_statements_definiendum_post_tl
4849       }
4850     }
4851   } {
4852     \exp_args:Nnx \defemph@uri {
4853       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4854     } { \l_stex_get_symbol_uri_str }
4855   }
4856 }
4857 \stex_deactivate_macro:Nn \definame {definition~environments}
4858
4859 \NewDocumentCommand \Definame { 0{ } m } {
4860   \__stex_statements_definiendum_args:n { #1 }
4861   \stex_get_symbol:n { #2 }
4862   \str_set:Nx \l_tmpa_str {
4863     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4864   }
4865   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}

```



```

4866 \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4867 \stex_html_backend:TF {
4868   \stex_if_do_html:T {
4869     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4870       \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
4871     }
4872   }
4873 } {
4874   \exp_args:Nnx \defemph@uri {
4875     \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
4876   } { \l_stex_get_symbol_uri_str }
4877 }
4878 }
4879 \stex_deactivate_macro:Nn \Definame {definition-environments}
4880
4881 \NewDocumentCommand \premise { m }{
4882   \stex_annotate:nnn{ premise }{}{ #1 }
4883 }
4884 \NewDocumentCommand \conclusion { m }{
4885   \stex_annotate:nnn{ conclusion }{}{ #1 }
4886 }
4887 \NewDocumentCommand \definiens { 0{} m }{
4888   \str_clear:N \l_stex_get_symbol_uri_str
4889   \tl_if_empty:nF {#1} {
4890     \stex_get_symbol:n { #1 }
4891   }
4892   \str_if_empty:NT \l_stex_get_symbol_uri_str {
4893     \int_compare:nNnTF {\clist_count:N \l__stex_statements_sdefinition_for_clist} = 1 {
4894       \str_set:Nx \l_stex_get_symbol_uri_str {\clist_item:Nn \l__stex_statements_sdefinition
4895     }{
4896       % TODO throw error
4897     }
4898   }
4899   \str_if_eq:eeT {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{module}}
4900   {\l_stex_current_module_str}{
4901     \str_if_eq:eeF {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defin
4902   }{true}{
4903     \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
4904     \exp_args:Nx \stex_add_to_current_module:n {
4905       \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
4906     }
4907   }
4908 }
4909 \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
4910 }
4911
4912 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
4913 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
4914 \stex_deactivate_macro:Nn \definiens {definition~environments}
4915

```

(End definition for `definame`. This function is documented on page 40.)

`sdefinition`

```

4916
4917 \keys_define:nn {stex / sdefinition }{
4918   type      .str_set_x:N = \sdefinitiontype,
4919   id        .str_set_x:N = \sdefinitionid,
4920   name      .str_set_x:N = \sdefinitionname,
4921   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4922   title     .tl_set:N     = \sdefinitiontitle
4923 }
4924 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
4925   \str_clear:N \sdefinitiontype
4926   \str_clear:N \sdefinitionid
4927   \str_clear:N \sdefinitionname
4928   \clist_clear:N \l__stex_statements_sdefinition_for_clist
4929   \tl_clear:N \sdefinitiontitle
4930   \keys_set:nn { stex / sdefinition }{ #1 }
4931 }
4932
4933 \NewDocumentEnvironment{sdefinition}{O{}}{
4934   \__stex_statements_sdefinition_args:n{ #1 }
4935   \stex_reactivate_macro:N \definiendum
4936   \stex_reactivate_macro:N \definame
4937   \stex_reactivate_macro:N \Definame
4938   \stex_reactivate_macro:N \premise
4939   \stex_reactivate_macro:N \definiens
4940   \stex_if_smsmode:F{
4941     \seq_clear:N \l_tmpa_seq
4942     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4943       \tl_if_empty:nF{ ##1 }{
4944         \stex_get_symbol:n { ##1 }
4945         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4946           \l_stex_get_symbol_uri_str
4947         }
4948       }
4949     }
4950     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpa_seq
4951     \exp_args:Nnnx
4952     \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {},}}
4953     \str_if_empty:NF \sdefinitiontype {
4954       \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
4955     }
4956     \str_if_empty:NF \sdefinitionname {
4957       \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
4958     }
4959     \clist_set:Nn \l_tmpa_clist \sdefinitiontype
4960     \tl_clear:N \l_tmpa_tl
4961     \clist_map_inline:Nn \l_tmpa_clist {
4962       \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
4963         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
4964       }
4965     }
4966     \tl_if_empty:NTF \l_tmpa_tl {
4967       \__stex_statements_sdefinition_start:
4968     }{
4969       \l_tmpa_tl

```

```

4970   }
4971   }
4972   \stex_ref_new_doc_target:n \sdefinitionid
4973   \stex_smsmode_do:
4974   ){
4975   \stex_suppress_html:n {
4976     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
4977   }
4978   \stex_if_smsmode:F {
4979     \clist_set:No \l_tmpa_clist \sdefinitiontype
4980     \tl_clear:N \l_tmpa_tl
4981     \clist_map_inline:Nn \l_tmpa_clist {
4982       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
4983         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
4984       }
4985     }
4986     \tl_if_empty:NTF \l_tmpa_tl {
4987       \__stex_statements_sdefinition_end:
4988     }{
4989       \l_tmpa_tl
4990     }
4991     \end{stex_annotate_env}
4992   }
4993   }

```

### **\stexpatchdefinition**

```

4994 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
4995   \par\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
4996     ~(\sdefinitiontitle)
4997   }~}
4998 }
4999 \cs_new_protected:Nn \__stex_statements_sdefinition_end: { \par\medskip}
5000
5001 \newcommand\stexpatchdefinition[3] [] {
5002   \str_set:Nx \l_tmpa_str{ #1 }
5003   \str_if_empty:NTF \l_tmpa_str {
5004     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
5005     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
5006   }{
5007     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
5008     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
5009   }
5010 }

```

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

\inlinedef inline:

```

5011 \keys_define:nn {stex / inlinedef }{
5012   type      .str_set_x:N = \sdefinitiontype,
5013   id        .str_set_x:N = \sdefinitionid,
5014   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5015   name      .str_set_x:N = \sdefinitionname
5016 }
5017 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {

```

```

5018 \str_clear:N \sdefinitiontype
5019 \str_clear:N \sdefinitionid
5020 \str_clear:N \sdefinitionname
5021 \clist_clear:N \l__stex_statements_sdefinition_for_clist
5022 \keys_set:nn { stex / inlinedef }{ #1 }
5023 }
5024 \NewDocumentCommand \inlinedef { 0{} m } {
5025   \begingroup
5026   \__stex_statements_inlinedef_args:n{ #1 }
5027   \stex_reactivate_macro:N \definiendum
5028   \stex_reactivate_macro:N \definame
5029   \stex_reactivate_macro:N \Definame
5030   \stex_reactivate_macro:N \premise
5031   \stex_reactivate_macro:N \definiens
5032   \stex_ref_new_doc_target:n \sdefinitionid
5033   \stex_if_smsmode:TF{\stex_suppress_html:n {
5034     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5035   }}{
5036     \seq_clear:N \l_tmpa_seq
5037     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5038       \tl_if_empty:nF{ ##1 }{
5039         \stex_get_symbol:n { ##1 }
5040         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5041           \l_stex_get_symbol_uri_str
5042         }
5043       }
5044     }
5045     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpa_seq
5046     \exp_args:Nnx
5047     \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
5048       \str_if_empty:NF \sdefinitiontype {
5049         \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5050       }
5051       #2
5052       \str_if_empty:NF \sdefinitionname {
5053         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sdefinitionname}}
5054         \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5055       }
5056     }
5057   }
5058   \endgroup
5059   \stex_smsmode_do:
5060 }

```

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

## 32.2 Assertions

**sassertion**

```

5061
5062 \keys_define:nn {stex / sassertion }{
5063   type      .str_set_x:N = \sassertiontype,
5064   id        .str_set_x:N = \sassertionid,

```

```

5065 title .tl_set:N = \sassertiontitle ,
5066 for .clist_set:N = \l__stex_statements_sassertion_for_clist ,
5067 name .str_set_x:N = \sassertionname
5068 }
5069 \cs_new_protected:Nn \l__stex_statements_sassertion_args:n {
5070 \str_clear:N \sassertiontype
5071 \str_clear:N \sassertionid
5072 \str_clear:N \sassertionname
5073 \clist_clear:N \l__stex_statements_sassertion_for_clist
5074 \tl_clear:N \sassertiontitle
5075 \keys_set:nn { stex / sassertion }{ #1 }
5076 }
5077
5078 %\tl_new:N \g__stex_statements_aftergroup_tl
5079
5080 \NewDocumentEnvironment{sassertion}{0{}}{
5081 \l__stex_statements_sassertion_args:n{ #1 }
5082 \stex_reactivate_macro:N \premise
5083 \stex_reactivate_macro:N \conclusion
5084 \stex_if_smsmode:F {
5085 \seq_clear:N \l_tmpa_seq
5086 \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5087 \tl_if_empty:nF{ ##1 }{
5088 \stex_get_symbol:n { ##1 }
5089 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5090 \l_stex_get_symbol_uri_str
5091 }
5092 }
5093 }
5094 \exp_args:Nnnx
5095 \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}
5096 \str_if_empty:NF \sassertiontype {
5097 \stex_annotate_invisible:nnn{type}{\sassertiontype}{ }
5098 }
5099 \str_if_empty:NF \sassertionname {
5100 \stex_annotate_invisible:nnn{statementname}{\sassertionname}{ }
5101 }
5102 \clist_set:Nn \l_tmpa_clist \sassertiontype
5103 \tl_clear:N \l_tmpa_tl
5104 \clist_map_inline:Nn \l_tmpa_clist {
5105 \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
5106 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
5107 }
5108 }
5109 \tl_if_empty:NTF \l_tmpa_tl {
5110 \l__stex_statements_sassertion_start:
5111 }{
5112 \l_tmpa_tl
5113 }
5114 }
5115 \str_if_empty:NTF \sassertionid {
5116 \str_if_empty:NF \sassertionname {
5117 \stex_ref_new_doc_target:n { }
5118 }

```

```

5119 } {
5120   \stex_ref_new_doc_target:n \sassertionid
5121 }
5122 \stex_smsmode_do:
5123 }{
5124   \str_if_empty:NF \sassertionname {
5125     \stex_suppress_html:n{\stex_symdecl_do:nn{ }\sassertionname}}
5126     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5127   }
5128   \stex_if_smsmode:F {
5129     \clist_set:Nn \l_tmpa_clist \sassertiontype
5130     \tl_clear:N \l_tmpa_tl
5131     \clist_map_inline:Nn \l_tmpa_clist {
5132       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
5133         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
5134       }
5135     }
5136     \tl_if_empty:NTF \l_tmpa_tl {
5137       \__stex_statements_sassertion_end:
5138     }{
5139       \l_tmpa_tl
5140     }
5141     \end{stex_annotate_env}
5142   }
5143 }

```

**\stexpatchassertion**

```

5144
5145 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
5146   \par\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
5147     (\sassertiontitle)
5148   }~}
5149 }
5150 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
5151
5152 \newcommand\stexpatchassertion[3] [] {
5153   \str_set:Nx \l_tmpa_str{ #1 }
5154   \str_if_empty:NTF \l_tmpa_str {
5155     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
5156     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
5157   }{
5158     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2 }
5159     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
5160   }
5161 }

```

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

**\inlineass** inline:

```

5162 \keys_define:nn {stex / inlineass }{
5163   type      .str_set_x:N = \sassertiontype,
5164   id        .str_set_x:N = \sassertionid,
5165   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
5166   name      .str_set_x:N = \sassertionname

```

```

5167 }
5168 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
5169   \str_clear:N \sassertiontype
5170   \str_clear:N \sassertionid
5171   \str_clear:N \sassertionname
5172   \clist_clear:N \l__stex_statements_sassertion_for_clist
5173   \keys_set:nn { stex / inlineass }{ #1 }
5174 }
5175 \NewDocumentCommand \inlineass { 0{} m } {
5176   \begingroup
5177   \stex_reactivate_macro:N \premise
5178   \stex_reactivate_macro:N \conclusion
5179   \__stex_statements_inlineass_args:n{ #1 }
5180   \str_if_empty:NTF \sassertionid {
5181     \str_if_empty:NF \sassertionname {
5182       \stex_ref_new_doc_target:n {}
5183     }
5184   } {
5185     \stex_ref_new_doc_target:n \sassertionid
5186   }
5187
5188   \stex_if_smsmode:TF{
5189     \str_if_empty:NF \sassertionname {
5190       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5191       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5192     }
5193   }{
5194     \seq_clear:N \l_tmpa_seq
5195     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5196       \tl_if_empty:nF{ ##1 }{
5197         \stex_get_symbol:n { ##1 }
5198         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5199           \l_stex_get_symbol_uri_str
5200         }
5201       }
5202     }
5203     \exp_args:Nnx
5204     \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{
5205       \str_if_empty:NF \sassertiontype {
5206         \stex_annotate_invisible:nnn{typestrings}{\sassertiontype}{}
5207       }
5208       #2
5209       \str_if_empty:NF \sassertionname {
5210         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5211         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5212         \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
5213       }
5214     }
5215   }
5216   \endgroup
5217   \stex_smsmode_do:
5218 }

```

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

## 32.3 Examples

sexample

```

5219 \keys_define:nn {stex / sexample }{
5220   type      .str_set_x:N = \exampletype,
5221   id        .str_set_x:N = \sexampleid,
5222   title     .tl_set:N     = \sexampletitle,
5223   name      .str_set_x:N = \sexamplename ,
5224   for       .clist_set:N = \l__stex_statements_sexample_for_clist,
5225 }
5226 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
5227   \str_clear:N \sexampletype
5228   \str_clear:N \sexampleid
5229   \str_clear:N \sexamplename
5230   \tl_clear:N \sexampletitle
5231   \clist_clear:N \l__stex_statements_sexample_for_clist
5232   \keys_set:nn { stex / sexample }{ #1 }
5233 }
5234
5235 \NewDocumentEnvironment{sexample}{0{}}{
5236   \__stex_statements_sexample_args:n{ #1 }
5237   \stex_reactivate_macro:N \premise
5238   \stex_reactivate_macro:N \conclusion
5239   \stex_if_smsmode:F {
5240     \seq_clear:N \l_tmpa_seq
5241     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5242       \tl_if_empty:NF{ ##1 }{
5243         \stex_get_symbol:n { ##1 }
5244         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5245           \l_stex_get_symbol_uri_str
5246         }
5247       }
5248     }
5249   }
5250   \exp_args:Nnnx
5251   \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
5252   \str_if_empty:NF \sexampletype {
5253     \stex_annotate_invisible:nnn{typestrings}{\sexampletype}{}
5254   }
5255   \str_if_empty:NF \sexamplename {
5256     \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
5257   }
5258   \clist_set:N \l_tmpa_clist \sexampletype
5259   \tl_clear:N \l_tmpa_tl
5260   \clist_map_inline:Nn \l_tmpa_clist {
5261     \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
5262       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
5263     }
5264   }
5265   \tl_if_empty:NTF \l_tmpa_tl {
5266     \__stex_statements_sexample_start:
5267   }{
5268     \l_tmpa_tl
5269   }

```



```

5270 }
5271 \str_if_empty:NF \sexampleid {
5272   \stex_ref_new_doc_target:n \sexampleid
5273 }
5274 \stex_smsmode_do:
5275 ){
5276   \str_if_empty:NF \sexamplename {
5277     \stex_suppress_html:n{\stex_symdecl_do:nn}{\sexamplename}}
5278   }
5279   \stex_if_smsmode:F {
5280     \clist_set:Nn \l_tmpa_clist \sexamplotype
5281     \tl_clear:N \l_tmpa_tl
5282     \clist_map_inline:Nn \l_tmpa_clist {
5283       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
5284         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
5285       }
5286     }
5287     \tl_if_empty:NTF \l_tmpa_tl {
5288       \__stex_statements_sexample_end:
5289     }{
5290       \l_tmpa_tl
5291     }
5292     \end{stex_annotate_env}
5293   }
5294 }

```

**\stexpatchexample**

```

5295
5296 \cs_new_protected:Nn \__stex_statements_sexample_start: {
5297   \par\noindent\titllemph{Example~\tl_if_empty:NF \sexamplotype {
5298     (\sexamplotype)
5299   }~}
5300 }
5301 \cs_new_protected:Nn \__stex_statements_sexample_end: { \par\medskip}
5302
5303 \newcommand\stexpatchexample[3] [] {
5304   \str_set:Nx \l_tmpa_str{ #1 }
5305   \str_if_empty:NTF \l_tmpa_str {
5306     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
5307     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
5308   }{
5309     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
5310     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
5311   }
5312 }

```

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

**\inlineex** inline:

```

5313 \keys_define:nn {stex / inlineex }{
5314   type      .str_set_x:N = \sexamplotype,
5315   id        .str_set_x:N = \sexampleid,
5316   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
5317   name      .str_set_x:N = \sexamplename

```

```

5318 }
5319 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
5320   \str_clear:N \sexamplotype
5321   \str_clear:N \sexampleid
5322   \str_clear:N \sexamplename
5323   \clist_clear:N \l__stex_statements_sexample_for_clist
5324   \keys_set:nn { stex / inlineex }{ #1 }
5325 }
5326 \NewDocumentCommand \inlineex { 0{ } m } {
5327   \beginngroup
5328   \stex_reactivate_macro:N \premise
5329   \stex_reactivate_macro:N \conclusion
5330   \__stex_statements_inlineex_args:n{ #1 }
5331   \str_if_empty:NF \sexampleid {
5332     \stex_ref_new_doc_target:n \sexampleid
5333   }
5334   \stex_if_smsmode:TF{
5335     \str_if_empty:NF \sexamplename {
5336       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5337     }
5338   }{
5339     \seq_clear:N \l_tmpa_seq
5340     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5341       \tl_if_empty:nF{ ##1 }{
5342         \stex_get_symbol:n { ##1 }
5343         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5344           \l_stex_get_symbol_uri_str
5345         }
5346       }
5347     }
5348     \exp_args:Nnx
5349     \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpa_seq {,}}{
5350       \str_if_empty:NF \sexamplotype {
5351         \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{ }
5352       }
5353       #2
5354       \str_if_empty:NF \sexamplename {
5355         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5356         \stex_annotate_invisible:nnn{statementname}{\sexamplename}{ }
5357       }
5358     }
5359   }
5360   \endgroup
5361   \stex_smsmode_do:
5362 }

```

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

## 32.4 Logical Paragraphs

sparagraph

```

5363 \keys_define:nn { stex / sparagraph } {
5364   id          .str_set_x:N    = \sparagraphid ,

```

```

5365 title .tl_set:N = \l_stex_sparagraph_title_tl ,
5366 type .str_set_x:N = \sparagraphtype ,
5367 for .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5368 from .tl_set:N = \sparagraphfrom ,
5369 to .tl_set:N = \sparagraphto ,
5370 start .tl_set:N = \l_stex_sparagraph_start_tl ,
5371 name .str_set:N = \sparagraphname ,
5372 imports .tl_set:N = \l__stex_statements_sparagraph_imports_tl
5373 }
5374
5375 \cs_new_protected:Nn \stex_sparagraph_args:n {
5376 \tl_clear:N \l_stex_sparagraph_title_tl
5377 \tl_clear:N \sparagraphfrom
5378 \tl_clear:N \sparagraphto
5379 \tl_clear:N \l_stex_sparagraph_start_tl
5380 \tl_clear:N \l__stex_statements_sparagraph_imports_tl
5381 \str_clear:N \sparagraphid
5382 \str_clear:N \sparagraphtype
5383 \clist_clear:N \l__stex_statements_sparagraph_for_clist
5384 \str_clear:N \sparagraphname
5385 \keys_set:nn { stex / sparagraph }{ #1 }
5386 }
5387 \newif\if@in@omtext\@in@omtextfalse
5388
5389 \NewDocumentEnvironment {sparagraph} { 0{} } {
5390 \stex_sparagraph_args:n { #1 }
5391 \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5392 \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
5393 }{
5394 \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
5395 }
5396 \@in@omtexttrue
5397 \stex_if_smsmode:F {
5398 \seq_clear:N \l_tmpa_seq
5399 \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5400 \tl_if_empty:NF{ ##1 }{
5401 \stex_get_symbol:n { ##1 }
5402 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5403 \l_stex_get_symbol_uri_str
5404 }
5405 }
5406 }
5407 \exp_args:Nnnx
5408 \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
5409 \str_if_empty:NF \sparagraphtype {
5410 \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{ }
5411 }
5412 \str_if_empty:NF \sparagraphfrom {
5413 \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{ }
5414 }
5415 \str_if_empty:NF \sparagraphto {
5416 \stex_annotate_invisible:nnn{to}{\sparagraphto}{ }
5417 }
5418 \str_if_empty:NF \sparagraphname {

```

```

5419     \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{\{
5420   }
5421   \clist_set:No \l_tmpa_clist \sparagraphtype
5422   \tl_clear:N \l_tmpa_tl
5423   \clist_map_inline:Nn \sparagraphtype {
5424     \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
5425       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
5426     }
5427   }
5428   \stex_csl_to_imports:No \usemodule \l__stex_statements_sparagraph_imports_tl
5429   \tl_if_empty:NTF \l_tmpa_tl {
5430     \__stex_statements_sparagraph_start:
5431   }{
5432     \l_tmpa_tl
5433   }
5434 }
5435 \clist_set:No \l_tmpa_clist \sparagraphtype
5436 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5437 {
5438   \stex_reactivate_macro:N \definiendum
5439   \stex_reactivate_macro:N \definame
5440   \stex_reactivate_macro:N \Definame
5441   \stex_reactivate_macro:N \premise
5442   \stex_reactivate_macro:N \definiens
5443 }
5444 \str_if_empty:NTF \sparagraphid {
5445   \str_if_empty:NTF \sparagraphname {
5446     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5447       \stex_ref_new_doc_target:n {}
5448     }
5449   } {
5450     \stex_ref_new_doc_target:n {}
5451   }
5452 } {
5453   \stex_ref_new_doc_target:n \sparagraphid
5454 }
5455 \exp_args:NNx
5456 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5457   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5458     \tl_if_empty:nF{ ##1 }{
5459       \stex_get_symbol:n { ##1 }
5460       \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5461     }
5462   }
5463 }
5464 \stex_smsmode_do:
5465 \ignorespacesandpars
5466 }{
5467   \str_if_empty:NF \sparagraphname {
5468     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5469     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5470   }
5471   \stex_if_smsmode:F {
5472     \clist_set:No \l_tmpa_clist \sparagraphtype

```

```

5473 \tl_clear:N \l_tmpa_tl
5474 \clist_map_inline:Nn \l_tmpa_clist {
5475   \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
5476     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
5477   }
5478 }
5479 \tl_if_empty:NTF \l_tmpa_tl {
5480   \__stex_statements_sparagraph_end:
5481 }{
5482   \l_tmpa_tl
5483 }
5484 \end{stex_annotate_env}
5485 }
5486 }

```

### **\stexpatchparagraph**

```

5487
5488 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
5489   \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5490     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
5491       \titleemph{\l_stex_sparagraph_title_tl}:~
5492     }
5493   }{
5494     \titleemph{\l_stex_sparagraph_start_tl}~
5495   }
5496 }
5497 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
5498
5499 \newcommand\stexpatchparagraph[3] [] {
5500   \str_set:Nx \l_tmpa_str{ #1 }
5501   \str_if_empty:NTF \l_tmpa_str {
5502     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
5503     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
5504   }{
5505     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
5506     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
5507   }
5508 }
5509
5510 \keys_define:nn { stex / inlinepara } {
5511   id      .str_set_x:N = \sparagraphid ,
5512   type    .str_set_x:N = \sparagraphtype ,
5513   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5514   from    .tl_set:N    = \sparagraphfrom ,
5515   to      .tl_set:N    = \sparagraphto ,
5516   name    .str_set:N    = \sparagraphname
5517 }
5518 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
5519   \tl_clear:N \sparagraphfrom
5520   \tl_clear:N \sparagraphto
5521   \str_clear:N \sparagraphid
5522   \str_clear:N \sparagraphtype
5523   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5524   \str_clear:N \sparagraphname

```

```

5525 \keys_set:nn { stex / inlinepara }{ #1 }
5526 }
5527 \NewDocumentCommand \inlinepara { 0{} m } {
5528   \beginingroup
5529   \__stex_statements_inlinepara_args:n{ #1 }
5530   \clist_set:No \l_tmpa_clist \sparagraphtype
5531   \str_if_empty:NTF \sparagraphid {
5532     \str_if_empty:NTF \sparagraphname {
5533       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5534         \stex_ref_new_doc_target:n {}
5535       }
5536     } {
5537       \stex_ref_new_doc_target:n {}
5538     }
5539   } {
5540     \stex_ref_new_doc_target:n \sparagraphid
5541   }
5542   \stex_if_smsmode:TF{
5543     \str_if_empty:NF \sparagraphname {
5544       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
5545     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5546   }
5547 }{
5548   \seq_clear:N \l_tmpa_seq
5549   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5550     \tl_if_empty:nF{ ##1 }{
5551       \stex_get_symbol:n { ##1 }
5552       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5553         \l_stex_get_symbol_uri_str
5554       }
5555     }
5556   }
5557   \exp_args:NNx
5558   \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
5559     \str_if_empty:NF \sparagraphtype {
5560       \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
5561     }
5562     \str_if_empty:NF \sparagraphfrom {
5563       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5564     }
5565     \str_if_empty:NF \sparagraphto {
5566       \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5567     }
5568     \str_if_empty:NF \sparagraphname {
5569       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
5570     \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
5571     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5572   }
5573   \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5574     \clist_map_inline:Nn \l_tmpa_seq {
5575       \stex_ref_new_sym_target:n {##1}
5576     }
5577   }
5578   #2

```

```

5579     }
5580   }
5581   \endgroup
5582   \stex_smsmode_do:
5583 }
5584

```

*(End definition for \stexpatchparagraph. This function is documented on page [42](#).)*

```

5585 \endpackage

```

# Chapter 33

## The Implementation

### 33.1 Package Options

We declare some switches which will modify the behavior according to the package options. Generally, an option `xxx` will just set the appropriate switches to true (otherwise they stay false).<sup>8</sup>

```
5586 <*package>
5587 <@@=stex_sproof>
5588
5589 %%%%%%%%%% sproof.dtx %%%%%%%%%%
5590
```

### 33.2 Proofs

We first define some keys for the proof environment.

```
5591 \keys_define:nn { stex / spf } {
5592   id          .str_set_x:N = \spfid,
5593   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,
5594   from        .tl_set:N    = \l__stex_sproof_spf_from_tl ,
5595   proofend    .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
5596   type        .str_set_x:N = \spftype,
5597   title       .tl_set:N    = \spftitle,
5598   continues   .tl_set:N    = \l__stex_sproof_spf_continues_tl,
5599   functions   .tl_set:N    = \l__stex_sproof_spf_functions_tl,
5600   method      .tl_set:N    = \l__stex_sproof_spf_method_tl
5601 }
5602 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
5603   \str_clear:N \spfid
5604   \tl_clear:N \l__stex_sproof_spf_for_tl
5605   \tl_clear:N \l__stex_sproof_spf_from_tl
5606   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
5607   \str_clear:N \spftype
5608   \tl_clear:N \spftitle
5609   \tl_clear:N \l__stex_sproof_spf_continues_tl
5610   \tl_clear:N \l__stex_sproof_spf_functions_tl

```

---

<sup>8</sup>EdNOTE: need an implementation for L<sup>A</sup>T<sub>E</sub>X<sub>M</sub>L



```

5611 \tl_clear:N \l__stex_sproof_spf_method_tl
5612 \bool_set_false:N \l__stex_sproof_inc_counter_bool
5613 \keys_set:nn { stex / spf }{ #1 }
5614 }

```

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

```

5615 \str_set:Nn\c__stex_sproof_flow_str{inline}

```

(End definition for `\c__stex_sproof_flow_str`.)

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

`pst@with@label` This environment manages<sup>7</sup> the path labeling of the proof steps in the description environment of the outermost `proof` environment. The argument is the label prefix up to now; which we cache in `\pst@label` (we need evaluate it first, since are in the right place now!). Then we increment the proof depth which is stored in `\count10` (lower counters are used by T<sub>E</sub>X for page numbering) and initialize the next level counter `\count\count10` with 1. In the end call for this environment, we just decrease the proof depth counter by 1 again.

```

5616 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
5617 \cs_new_protected:Npn \sproofnumber {
5618   \int_set:Nn \l_tmpa_int {1}
5619   \bool_while_do:nn {
5620     \int_compare_p:nNn {
5621       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5622     } > 0
5623   }{
5624     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
5625     \int_incr:N \l_tmpa_int
5626   }
5627 }
5628 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
5629   \int_set:Nn \l_tmpa_int {1}
5630   \bool_while_do:nn {
5631     \int_compare_p:nNn {
5632       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5633     } > 0
5634   }{
5635     \int_incr:N \l_tmpa_int
5636   }
5637   \int_compare:nNnF \l_tmpa_int = 1 {
5638     \int_decr:N \l_tmpa_int
5639   }
5640   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
5641     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1

```

---

<sup>7</sup>This gets the labeling right but only works 8 levels deep



```

5686     }
5687     \clist_if_in:NnT \l_tmpa_clist {finnish}{
5688       \input{sproof-finnish.ldf}
5689     }
5690     \clist_if_in:NnT \l_tmpa_clist {french}{
5691       \input{sproof-french.ldf}
5692     }
5693     \clist_if_in:NnT \l_tmpa_clist {russian}{
5694       \input{sproof-russian.ldf}
5695     }
5696     \makeatother
5697   }{}
5698 }

```

spfsketch

```

5699 \newcommand\spfsketch[2] [] {
5700   \beginingroup
5701   \let \premise \stex_proof_premise:
5702   \__stex_sproof_spf_args:n{#1}
5703   \stex_if_smsmode:TF {
5704     \str_if_empty:NF \spfid {
5705       \stex_ref_new_doc_target:n \spfid
5706     }
5707   }{
5708     \seq_clear:N \l_tmpa_seq
5709     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5710       \tl_if_empty:nF{ ##1 }{
5711         \stex_get_symbol:n { ##1 }
5712         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5713           \l_stex_get_symbol_uri_str
5714         }
5715       }
5716     }
5717     \exp_args:Nnx
5718     \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
5719       \str_if_empty:NF \spftype {
5720         \stex_annotate_invisible:nnn{type}{\spftype}{-}
5721       }
5722       \clist_set:No \l_tmpa_clist \spftype
5723       \tl_set:Nn \l_tmpa_tl {
5724         \titleemph{
5725           \tl_if_empty:NTF \spftitle {
5726             \spf@proofsketch@kw
5727           }{
5728             \spftitle
5729           }
5730         }::~
5731       }
5732       \clist_map_inline:Nn \l_tmpa_clist {
5733         \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5734           \tl_clear:N \l_tmpa_tl
5735         }
5736       }
5737       \str_if_empty:NF \spfid {

```

```

5738         \stex_ref_new_doc_target:n \spfid
5739     }
5740     \l_tmpa_tl #2 \sproofend
5741 }
5742 }
5743 \endgroup
5744 \stex_smsmode_do:
5745 }
5746

```

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

**spfeq** This is very similar to \spfsketch, but uses a computation array<sup>910</sup>

```

5747 \newenvironment{spfeq}[2][]{
5748   \__stex_sproof_spf_args:n{#1}
5749   \let \premise \stex_proof_premise:
5750   \stex_if_smsmode:TF {
5751     \str_if_empty:NF \spfid {
5752       \stex_ref_new_doc_target:n \spfid
5753     }
5754   }{
5755     \seq_clear:N \l_tmpa_seq
5756     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5757       \tl_if_empty:NF{ ##1 }{
5758         \stex_get_symbol:n { ##1 }
5759         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5760           \l_stex_get_symbol_uri_str
5761         }
5762       }
5763     }
5764     \exp_args:Nnnx
5765     \begin{stex_annotate_env}{spfeq}{\seq_use:Nn \l_tmpa_seq {,}}
5766     \str_if_empty:NF \spftype {
5767       \stex_annotate_invisible:nnn{type}{\spftype}{ }
5768     }
5769
5770     \clist_set:No \l_tmpa_clist \spftype
5771     \tl_clear:N \l_tmpa_tl
5772     \clist_map_inline:Nn \l_tmpa_clist {
5773       \tl_if_exist:cT {__stex_sproof_spfeq_##1_start:}{
5774         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_start:}}
5775       }
5776       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5777         \tl_set:Nn \l_tmpa_tl {\use:n{ }}
5778       }
5779     }
5780     \tl_if_empty:NTF \l_tmpa_tl {
5781       \__stex_sproof_spfeq_start:
5782     }{
5783       \l_tmpa_tl
5784     }{-#2}

```

<sup>9</sup>EDNOTE: This should really be more like a tabular with an ensuremath in it. or invoke text on the last column

<sup>10</sup>EDNOTE: document above

```

5785 \str_if_empty:NF \spfid {
5786 \stex_ref_new_doc_target:n \spfid
5787 }
5788 \begin{displaymath}\begin{array}{rc1l}
5789 }
5790 \stex_smsmode_do:
5791 }{
5792 \stex_if_smsmode:F {
5793 \end{array}\end{displaymath}
5794 \clist_set:No \l_tmpa_clist \spftype
5795 \tl_clear:N \l_tmpa_tl
5796 \clist_map_inline:Nn \l_tmpa_clist {
5797 \tl_if_exist:cT {__stex_sproof_spfeq_##1_end:}{
5798 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_end:}}
5799 }
5800 }
5801 \tl_if_empty:NTF \l_tmpa_tl {
5802 \__stex_sproof_spfeq_end:
5803 }{
5804 \l_tmpa_tl
5805 }
5806 \end{stex_annotate_env}
5807 }
5808 }
5809
5810 \cs_new_protected:Nn \__stex_sproof_spfeq_start: {
5811 \titleemph{
5812 \tl_if_empty:NTF \spftitle {
5813 \spf@proof@kw
5814 }{
5815 \spftitle
5816 }
5817 }:
5818 }
5819 \cs_new_protected:Nn \__stex_sproof_spfeq_end: {\sproofend}
5820
5821 \newcommand\stexpatchspfeq[3] [] {
5822 \str_set:Nx \l_tmpa_str{ #1 }
5823 \str_if_empty:NTF \l_tmpa_str {
5824 \tl_set:Nn \__stex_sproof_spfeq_start: { #2 }
5825 \tl_set:Nn \__stex_sproof_spfeq_end: { #3 }
5826 }{
5827 \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_start:\endcsname{ #2 }
5828 \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_end:\endcsname{ #3 }
5829 }
5830 }
5831

```

(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.

```

5832 \newenvironment{sproof}[2] []{

```

```

5833 \let \premise \stex_proof_premise:
5834 \intarray_gzero:N \l__stex_sproof_counter_intarray
5835 \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
5836 \__stex_sproof_spf_args:n{#1}
5837 \stex_if_smsmode:TF {
5838   \str_if_empty:NF \spfid {
5839     \stex_ref_new_doc_target:n \spfid
5840   }
5841 }{
5842   \seq_clear:N \l_tmpa_seq
5843   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5844     \tl_if_empty:NF{ ##1 }{
5845       \stex_get_symbol:n { ##1 }
5846       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5847         \l_stex_get_symbol_uri_str
5848       }
5849     }
5850   }
5851   \exp_args:Nnnx
5852   \begin{stex_annotate_env}{sproof}{\seq_use:Nn \l_tmpa_seq {},}}
5853   \str_if_empty:NF \spftype {
5854     \stex_annotate_invisible:nnn{type}{\spftype}{}
5855   }
5856
5857   \clist_set:No \l_tmpa_clist \spftype
5858   \tl_clear:N \l_tmpa_tl
5859   \clist_map_inline:Nn \l_tmpa_clist {
5860     \tl_if_exist:cT {__stex_sproof_sproof_##1_start:}{
5861       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_start:}}
5862     }
5863     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5864       \tl_set:Nn \l_tmpa_tl {\use:n{}}
5865     }
5866   }
5867   \tl_if_empty:NTF \l_tmpa_tl {
5868     \__stex_sproof_sproof_start:
5869   }{
5870     \l_tmpa_tl
5871   }{~#2}
5872   \str_if_empty:NF \spfid {
5873     \stex_ref_new_doc_target:n \spfid
5874   }
5875   \begin{description}
5876 }
5877 \stex_smsmode_do:
5878 }{
5879   \stex_if_smsmode:F{
5880     \end{description}
5881     \clist_set:No \l_tmpa_clist \spftype
5882     \tl_clear:N \l_tmpa_tl
5883     \clist_map_inline:Nn \l_tmpa_clist {
5884       \tl_if_exist:cT {__stex_sproof_sproof_##1_end:}{
5885         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_end:}}
5886       }

```

```

5887     }
5888     \tl_if_empty:NTF \l_tmpa_tl {
5889       \__stex_sproof_sproof_end:
5890     }{
5891       \l_tmpa_tl
5892     }
5893     \end{stex_annotate_env}
5894   }
5895 }
5896
5897 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
5898   \par\noindent\titleemph{
5899     \tl_if_empty:NTF \spftype {
5900       \spf@proof@kw
5901     }{
5902       \spftype
5903     }
5904   }:
5905 }
5906 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
5907
5908 \newcommand\stexpatchproof[3] [] {
5909   \str_set:Nx \l_tmpa_str{ #1 }
5910   \str_if_empty:NTF \l_tmpa_str {
5911     \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
5912     \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
5913   }{
5914     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
5915     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
5916   }
5917 }

```

\spfidea

```

5918 \newcommand\spfidea[2] []{
5919   \__stex_sproof_spf_args:n{#1}
5920   \titleemph{
5921     \tl_if_empty:NTF \spftype {Proof~Idea}{
5922       \spftype
5923     }:
5924   }~#2
5925   \sproofend
5926 }

```

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

The next two environments (proof steps) and comments, are mostly semantical, they take `KeyVal` arguments that specify their semantic role. In draft mode, they read these values and show them. If the surrounding proof had `display=flow`, then no new `\item` is generated, otherwise it is. In any case, the proof step number (at the current level) is incremented.

spfstep

```

5927 \newenvironment{spfstep}[1] []{
5928   \__stex_sproof_spf_args:n{#1}
5929   \stex_if_smsmode:TF {

```

```

5930 \str_if_empty:NF \spfid {
5931 \stex_ref_new_doc_target:n \spfid
5932 }
5933 }{
5934 \@in@omtexttrue
5935 \seq_clear:N \l_tmpa_seq
5936 \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5937 \tl_if_empty:nF{ ##1 }{
5938 \stex_get_symbol:n { ##1 }
5939 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5940 \l_stex_get_symbol_uri_str
5941 }
5942 }
5943 }
5944 \exp_args:Nnnx
5945 \begin{stex_annotate_env}{spfstep}{\seq_use:Nn \l_tmpa_seq {,}}
5946 \str_if_empty:NF \spftype {
5947 \stex_annotate_invisible:nnn{type}{\spftype}{}
5948 }
5949 \clist_set:No \l_tmpa_clist \spftype
5950 \tl_set:Nn \l_tmpa_tl {
5951 \item[\sproofnumber]
5952 \bool_set_true:N \l__stex_sproof_inc_counter_bool
5953 }
5954 \clist_map_inline:Nn \l_tmpa_clist {
5955 \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5956 \tl_clear:N \l_tmpa_tl
5957 }
5958 }
5959 \l_tmpa_tl
5960 \tl_if_empty:NF \spftitle {
5961 {(\titleemph{\spftitle})\enspace}
5962 }
5963 \str_if_empty:NF \spfid {
5964 \stex_ref_new_doc_target:n \spfid
5965 }
5966 }
5967 \stex_smsmode_do:
5968 \ignorespacesandpars
5969 }{
5970 \bool_if:NT \l__stex_sproof_inc_counter_bool {
5971 \__stex_sproof_inc_counter:
5972 }
5973 \stex_if_smsmode:F {
5974 \end{stex_annotate_env}
5975 }
5976 }

```

sproofcomment

```

5977 \newenvironment{sproofcomment}[1][]{
5978 \__stex_sproof_spf_args:n{#1}
5979 \clist_set:No \l_tmpa_clist \spftype
5980 \tl_set:Nn \l_tmpa_tl {
5981 \item[\sproofnumber]

```



```

5982 \bool_set_true:N \l__stex_sproof_inc_counter_bool
5983 }
5984 \clist_map_inline:Nn \l_tmpa_clist {
5985   \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5986     \tl_clear:N \l_tmpa_tl
5987   }
5988 }
5989 \l_tmpa_tl
5990 }{
5991   \bool_if:NT \l__stex_sproof_inc_counter_bool {
5992     \__stex_sproof_inc_counter:
5993   }
5994 }

```

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.

```

5995 \newenvironment{subproof}[2][]{
5996   \__stex_sproof_spf_args:n{#1}
5997   \stex_if_smsmode:TF{
5998     \str_if_empty:NF \spfid {
5999       \stex_ref_new_doc_target:n \spfid
6000     }
6001   }{
6002     \seq_clear:N \l_tmpa_seq
6003     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6004       \tl_if_empty:nF{ ##1 }{
6005         \stex_get_symbol:n { ##1 }
6006         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6007           \l_stex_get_symbol_uri_str
6008         }
6009       }
6010     }
6011     \exp_args:Nnnx
6012     \begin{stex_annotate_env}{subproof}{\seq_use:Nn \l_tmpa_seq {,}}
6013     \str_if_empty:NF \spftype {
6014       \stex_annotate_invisible:nnn{type}{\spftype}{\}
6015     }
6016
6017     \clist_set:No \l_tmpa_clist \spftype
6018     \tl_set:Nn \l_tmpa_tl {
6019       \item[\sproofnumber]
6020       \bool_set_true:N \l__stex_sproof_inc_counter_bool
6021     }
6022     \clist_map_inline:Nn \l_tmpa_clist {
6023       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6024         \tl_clear:N \l_tmpa_tl
6025       }
6026     }
6027     \l_tmpa_tl
6028     \tl_if_empty:NF \spftitle {
6029       {(\titleemph{\spftitle})\enspace}
6030     }

```

```

6031     {~#2}
6032     \str_if_empty:NF \spfid {
6033       \stex_ref_new_doc_target:n \spfid
6034     }
6035   }
6036   \__stex_sproof_add_counter:
6037   \stex_smsmode_do:
6038 }{
6039   \__stex_sproof_remove_counter:
6040   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6041     \__stex_sproof_inc_counter:
6042   }
6043   \stex_if_smsmode:F{
6044     \end{stex_annotate_env}
6045   }
6046 }

```

**spfcases** In the **pfcases** environment, the start text is displayed as the first comment of the proof.

```

6047 \newenvironment{spfcases}[2][]{
6048   \tl_if_empty:nTF{#1}{
6049     \begin{subproof}[method=by-cases]{#2}
6050   }{
6051     \begin{subproof}[#1,method=by-cases]{#2}
6052   }
6053 }{
6054   \end{subproof}
6055 }

```

**spfcase** In the **pfcase** environment, the start text is displayed specification of the case after the **\item**

```

6056 \newenvironment{spfcase}[2][]{
6057   \__stex_sproof_spf_args:n{#1}
6058   \stex_if_smsmode:TF {
6059     \str_if_empty:NF \spfid {
6060       \stex_ref_new_doc_target:n \spfid
6061     }
6062   }{
6063     \seq_clear:N \l_tmpa_seq
6064     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6065       \tl_if_empty:nF{ ##1 }{
6066         \stex_get_symbol:n { ##1 }
6067         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6068           \l_stex_get_symbol_uri_str
6069         }
6070       }
6071     }
6072     \exp_args:Nnnx
6073     \begin{stex_annotate_env}{spfcase}{\seq_use:Nn \l_tmpa_seq {,}}
6074     \str_if_empty:NF \spftype {
6075       \stex_annotate_invisible:nnn{type}{\spftype}{}}
6076   }
6077   \clist_set:Nn \l_tmpa_clist \spftype
6078   \tl_set:Nn \l_tmpa_tl {
6079     \item[\sproofnumber]

```

```

6080     \bool_set_true:N \l__stex_sproof_inc_counter_bool
6081   }
6082   \clist_map_inline:Nn \l_tmpa_clist {
6083     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6084       \tl_clear:N \l_tmpa_tl
6085     }
6086   }
6087   \l_tmpa_tl
6088   \tl_if_empty:nF{#2}{
6089     \titleemph{#2}:~
6090   }
6091 }
6092 \__stex_sproof_add_counter:
6093 \stex_smsmode_do:
6094 ){
6095   \__stex_sproof_remove_counter:
6096   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6097     \__stex_sproof_inc_counter:
6098   }
6099   \stex_if_smsmode:F{
6100     \clist_set:No \l_tmpa_clist \spftype
6101     \tl_set:Nn \l_tmpa_tl{\sproofend}
6102     \clist_map_inline:Nn \l_tmpa_clist {
6103       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6104         \tl_clear:N \l_tmpa_tl
6105       }
6106     }
6107     \l_tmpa_tl
6108     \end{stex_annotate_env}
6109   }
6110 }

```

**spfcase** similar to **spfcase**, takes a third argument.

```

6111 \newcommand\spfcasesketch[3][]{
6112   \begin{spfcase}[#1]{#2}#3\end{spfcase}
6113 }

```

### 33.3 Justifications

We define the actions that are undertaken, when the keys for justifications are encountered. Here this is very simple, we just define an internal macro with the value, so that we can use it later.

```

6114 \keys_define:nn { stex / just }{
6115   id      .str_set:x:N = \l__stex_sproof_just_id_str,
6116   method  .tl_set:N    = \l__stex_sproof_just_method_tl,
6117   premises .tl_set:N    = \l__stex_sproof_just_premises_tl,
6118   args     .tl_set:N    = \l__stex_sproof_just_args_tl
6119 }

```

The next three environments and macros are purely semantic, so we ignore the keyval arguments for now and only display the content.<sup>11</sup>

<sup>11</sup>EDNOTE: need to do something about the premise in draft mode.

**justification**

```
6120 \newenvironment{justification}[1] [] {}{}
```

**\premise**

```
6121 \newcommand\stex_proof_promise:[2] [] {#2}
```

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

**\justarg** the **\justarg** macro is purely semantic, so we ignore the keyval arguments for now and only display the content.

```
6122 \newcommand\justarg[2] [] {#2}
```

```
6123 \end{package}
```

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

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

## Chapter 34

# STEX -Others Implementation

```
6124 <*package>
6125
6126 %%%%%%%%%% others.dtx %%%%%%%%%%
6127
6128 <@@=stex_others>
        Warnings and error messages
6129 % None

\MSC Math subject classifier

6130 \NewDocumentCommand \MSC {m} {
6131 % TODO
6132 }

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

6133 \@ifpackageloaded{tikzinput}{
6134 \RequirePackage{stex-tikzinput}
6135 }{}
6136
6137 \bool_if:NT \c_stex_persist_mode_bool {
6138 \input{\jobname.sms}
6139 \prop_if_exist:NT\c_stex_mathhub_main_manifest_prop{
6140 \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
6141 \l_tmpa_str
6142 \prop_set_eq:cN { c_stex_mathhub\_l_tmpa_str_manifest_prop }
6143 \c_stex_mathhub_main_manifest_prop
6144 \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
6145 }
6146 }

6147 </package>
```

## Chapter 35

# STEX -Metatheory Implementation

```
6148 <*package>
6149 <@@=stex_modules>
6150
6151 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
6152
6153 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX/meta}
6154 \begingroup
6155 \stex_module_setup:nn{
6156   ns=\c_stex_metatheory_ns_str,
6157   meta=NONE
6158 }{Metatheory}
6159 \stex_reactivate_macro:N \symdecl
6160 \stex_reactivate_macro:N \notation
6161 \stex_reactivate_macro:N \symdef
6162 \ExplSyntaxOff
6163 \csname stex_suppress_html:n\endcsname{
6164   % is-a (a:A, a \in A, a is an A, etc.)
6165   \symdecl{isa}[args=ai]
6166   \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
6167   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
6168   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
6169
6170   % bind (\forall, \Pi, \lambda etc.)
6171   \symdecl{bind}[args=Bi]
6172   \notation{bind}[forall]{\comp\forall #1.;#2}{##1 \comp, ##2}
6173   \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6174   \notation{bind}[depfun]{\comp( #1 \comp{ }\;\to\; ) #2}{##1 \comp, ##2}
6175
6176   % implicit bind
6177   \symdef{implicitbind}[args=Bi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6178
6179   % dummy variable
6180   \symdecl{dummyvar}
6181   \notation{dummyvar}[underscore]{\comp\_}
6182   \notation{dummyvar}[dot]{\comp\cdot}
```

```

6183 \notation{dummyvar}[dash]{\comp{\rm --}}
6184
6185 %fromto (function space, Hom-set, implication etc.)
6186 \symdecl{fromto}[args=ai]
6187 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
6188 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
6189
6190 % mapto (lambda etc.)
6191 \symdecl{mapto}[args=Bi]
6192 \notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
6193 \notation{mapto}[lambda]{\comp\lambda #1 \comp.; #2}{#1 \comp, #2}
6194 \notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.; #2}{#1 \comp, #2}
6195
6196 % function/operator application
6197 \symdecl{apply}[args=ia]
6198 \notation{apply}[prec=0;0x\infprec,parens]{#1 \comp( #2 \comp)}{##1 \comp, ##2}
6199 \notation{apply}[prec=0;0x\infprec,lambda]{#1 \; #2 }{##1 \; ; ##2}
6200
6201 % collection of propositions/booleans/truth values
6202 \symdecl{prop}[name=proposition]
6203 \notation{prop}[prop]{\comp{\rm prop}}
6204 \notation{prop}[BOOL]{\comp{\rm BOOL}}
6205
6206 \symdecl{judgmentholds}[args=1]
6207 \notation{judgmentholds}[vdash,op=\vdash]{\comp\vdash\; ; #1}
6208
6209 % sequences
6210 \symdecl{seqtype}[args=1]
6211 \notation{seqtype}[kleene]{#1^{\comp\ast}}
6212
6213 \symdecl{seqexpr}[args=a]
6214 \notation{seqexpr}[angle,prec=nobrackets]{\comp\angle #1\comp\rangle}{##1\comp,##2}
6215
6216 \symdef{sequence-index}[args=2,li,prec=nobrackets]{#{#1}_{#2}}
6217 \notation{sequence-index}[ui,prec=nobrackets]{#{#1}^{#2}}
6218
6219 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{\,\ellipses}}{##1\comp,##2}
6220 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{\,\ellipses},#2}{##1\comp,##2}
6221 \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{\,\ellipses},#2\comp{\,\ellipses},#3}
6222
6223 % letin (''let'', local definitions, variable substitution)
6224 \symdecl{letin}[args=bii]
6225 \notation{letin}[let]{\comp{\rm let}}\;#1\comp{=}\;#2\; \comp{\rm in}}\;#3}
6226 \notation{letin}[subst]{#3 \comp[ #1 \comp/ #2 \comp]}
6227 \notation{letin}[frac]{#3 \comp[ \frac{#2}{#1} \comp]}
6228
6229 % structures
6230 \symdecl*{module-type}[args=1]
6231 \notation{module-type}{\comp{\mathtt{MOD}}} #1}
6232 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
6233 \notation{mathstruct}[angle,prec=nobrackets]{\comp\angle #1 \comp\rangle}{##1 \comp, ##2}
6234
6235 % objects
6236 \symdecl{object}

```

```

6237 \notation{object}{\comp{\mathtt{OBJECT}}}
6238
6239 }
6240 \ExplSyntaxOn
6241 \stex_add_to_current_module:n{
6242   \let\nappa\apply
6243   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
6244   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
6245   \def\livar{\csname sequence-index\endcsname[li]}
6246   \def\uivar{\csname sequence-index\endcsname[ui]}
6247   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
6248   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
6249   \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
6250 }
6251 \__stex_modules_end_module:
6252 \endgroup
6253 \</package>

```



## Chapter 36

# Tikzinput Implementation

```
6254 <@@=tikzinput>
6255 <*package>
6256
6257 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
6258
6259 \ProvidesExplPackage{tikzinput}{2022/02/26}{3.0.1}{tikzinput package}
6260 \RequirePackage{l3keys2e}
6261
6262 \keys_define:nn { tikzinput } {
6263   image .bool_set:N = \c_tikzinput_image_bool,
6264   image .default:n = false ,
6265   unknown .code:n = {}
6266 }
6267
6268 \ProcessKeysOptions { tikzinput }
6269
6270 \bool_if:NTF \c_tikzinput_image_bool {
6271   \RequirePackage{graphicx}
6272
6273   \providecommand\usetikzlibrary[]{}
6274   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
6275 }{
6276   \RequirePackage{tikz}
6277   \RequirePackage{standalone}
6278
6279   \newcommand \tikzinput [2] [] {
6280     \setkeys{Gin}{#1}
6281     \ifx \Gin@ewidth \Gin@exclamation
6282       \ifx \Gin@eheight \Gin@exclamation
6283         \input { #2 }
6284       \else
6285         \resizebox{!}{ \Gin@eheight }{
6286           \input { #2 }
6287         }
6288       \fi
6289     \else
6290       \ifx \Gin@eheight \Gin@exclamation
6291         \resizebox{ \Gin@ewidth }{!}{
```

```

6292         \input { #2 }
6293     }
6294     \else
6295         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
6296             \input { #2 }
6297         }
6298     \fi
6299 \fi
6300 }
6301 }
6302
6303 \newcommand \ctikzinput [2] [] {
6304     \begin{center}
6305         \tikzinput [#1] {#2}
6306     \end{center}
6307 }
6308
6309 \@ifpackageloaded{stex}{
6310     \RequirePackage{stex-tikzinput}
6311 }{}
6312
6313 </package>
6314 <*stex>
6315 \ProvidesExplPackage{stex-tikzinput}{2022/02/26}{3.0.1}{stex-tikzinput}
6316 \RequirePackage{stex}
6317 \RequirePackage{tikzinput}
6318
6319 \newcommand\mhtikzinput[2] []{%
6320     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
6321     \stex_in_repository:nn\Gin@mhrepos{
6322         \tikzinput[#1]{\mhp@hpath{##1}{#2}}
6323     }
6324 }
6325 \newcommand\cmhtikzinput[2] []{\begin{center}\mhtikzinput[#1]{#2}\end{center}}
6326
6327 \cs_new_protected:Nn \__tikzinput_usetikzlibrary:nn {
6328     \pgfkeys@spdef\pgf@temp{#1}
6329     \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
6330     \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgf@temp
6331     \expandafter\edef\csname tikz@library@#1@atcode\endcsname{\the\catcode'\@}
6332     \expandafter\edef\csname tikz@library@#1@barcode\endcsname{\the\catcode'\|}
6333     \expandafter\edef\csname tikz@library@#1@dollarcode\endcsname{\the\catcode'\$}
6334     \catcode'\@=11
6335     \catcode'\|=12
6336     \catcode'\$=3
6337     \pgfutil@InputIfFileExists{#2}{-}{-}
6338     \catcode'\@=\csname tikz@library@#1@atcode\endcsname
6339     \catcode'\|=\csname tikz@library@#1@barcode\endcsname
6340     \catcode'\$=\csname tikz@library@#1@dollarcode\endcsname
6341 }
6342
6343
6344 \newcommand\libusetikzlibrary[1]{

```

```

6345 \prop_if_exist:NF \l_stex_current_repository_prop {
6346   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6347 }
6348 \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
6349   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6350 }
6351 \seq_clear:N \l__tikzinput_libinput_files_seq
6352 \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
6353 \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
6354
6355 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
6356   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / tikzlibrary
6357   \IfFileExists{ \l_tmpa_str }{
6358     \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6359   }{}
6360   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
6361   \seq_put_right:No \l_tmpa_seq \l_tmpa_str
6362 }
6363
6364 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / tikzlibrary #1 .code.t
6365 \IfFileExists{ \l_tmpa_str }{
6366   \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6367 }{}
6368
6369 \seq_if_empty:NTF \l__tikzinput_libinput_files_seq {
6370   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .code.t
6371 }{
6372   \int_compare:nNnTF {\seq_count:N \l__tikzinput_libinput_files_seq} = 1 {
6373     \seq_map_inline:Nn \l__tikzinput_libinput_files_seq {
6374       \__tikzinput_usetikzlibrary:nn{#1}{ ##1 }
6375     }
6376   }{
6377     \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .co
6378   }
6379 }
6380 }
6381 </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

```
6382 <*package>
6383 <@@=document_structure>
6384 \ProvidesExplPackage{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure}
6385 \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).

```
6386
6387 \keys_define:nn{ document-structure }{
6388   class      .str_set_x:N = \c_document_structure_class_str,
6389   topsect    .str_set_x:N = \c_document_structure_topsect_str,,
6390   unknown    .code:n      = {
6391     \PassOptionsToClass{\CurrentOption}{stex}
6392     \PassOptionsToClass{\CurrentOption}{tikzinput}
6393   }
6394   % showignores .bool_set:N = \c_document_structure_showignores_bool,
6395 }
6396 \ProcessKeysOptions{ document-structure }
6397 \str_if_empty:NT \c_document_structure_class_str {
6398   \str_set:Nn \c_document_structure_class_str {article}
6399 }
6400 \str_if_empty:NT \c_document_structure_topsect_str {
6401   \str_set:Nn \c_document_structure_topsect_str {section}
6402 }
```

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

```
6403 \RequirePackage{xspace}
6404 \RequirePackage{comment}
6405 \RequirePackage{stex}
6406 \AddToHook{begindocument}{
```

```

6407 \ltx@ifpackageloaded{babel}{
6408   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6409   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6410     \makeatletter\input{document-structure-ngerman.ldf}\makeatother
6411   }
6412 }{}
6413 }

```

`\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.

```

6414 \int_new:N \l_document_structure_section_level_int
6415 \str_case:NnF \c_document_structure_topsect_str {
6416   {part}}{
6417     \int_set:Nn \l_document_structure_section_level_int {0}
6418   }
6419   {chapter}{
6420     \int_set:Nn \l_document_structure_section_level_int {1}
6421   }
6422 }{
6423   \str_case:NnF \c_document_structure_class_str {
6424     {book}{
6425       \int_set:Nn \l_document_structure_section_level_int {0}
6426     }
6427     {report}{
6428       \int_set:Nn \l_document_structure_section_level_int {0}
6429     }
6430   }{
6431     \int_set:Nn \l_document_structure_section_level_int {2}
6432   }
6433 }

```

## 37.2 Document Structure

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

`\currentsectionlevel` For the `\currentsectionlevel` and `\Currentsectionlevel` macros we use an internal macro `\current@section@level` that only contains the keyword (no markup). We initialize it with “document” as a default. In the generated OMDoc, we only generate a text element of class `omdoc_currentsectionlevel`, which will be instantiated by CSS later.<sup>12</sup>

EdN:12

```

6434 \def\current@section@level{document}%
6435 \newcommand\currentsectionlevel{\lowercase\expandafter\current@section@level\xspace}%
6436 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%

```

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

`\skipomgroup`

```

6437 \cs_new_protected:Npn \skipomgroup {

```

---

<sup>12</sup>EdNOTE: MK: we may have to experiment with the more powerful uppercasing macro from `mfirstuc.sty` once we internationalize.

```

6438 \ifcase\l_document_structure_section_level_int
6439 \or\stepcounter{part}
6440 \or\stepcounter{chapter}
6441 \or\stepcounter{section}
6442 \or\stepcounter{subsection}
6443 \or\stepcounter{subsubsection}
6444 \or\stepcounter{paragraph}
6445 \or\stepcounter{subparagraph}
6446 \fi
6447 }

```

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

**blindfragment**

```

6448 \newcommand\at@begin@blindomgroup[1]{
6449 \newenvironment{blindfragment}
6450 {
6451 \int_incr:N\l_document_structure_section_level_int
6452 \at@begin@blindomgroup\l_document_structure_section_level_int
6453 }{}

```

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

```

6454 \newcommand\omgroup@nonum[2]{
6455 \ifx\hyper@anchor\undefined\else\phantomsection\fi
6456 \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
6457 }

```

(End definition for \omgroup@nonum. This function is documented on page ??.)

**\omgroup@num** convenience macro: `\omgroup@num{<level>}{<title>}` makes numbered sectioning with title `<title>` at level `<level>`. We have to check the `short` key was given in the `omgroup` environment and – if it is use it. But how to do that depends on whether the `rdmeta` package has been loaded. In the end we call `\sref@label@id` to enable crossreferencing.

```

6458 \newcommand\omgroup@num[2]{
6459 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
6460 \@nameuse{#1}{#2}
6461 }{
6462 \cs_if_exist:NTF\rdmeta@sectioning{
6463 \@nameuse{rdmeta@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
6464 }{
6465 \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
6466 }
6467 }
6468 %\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\omgroup@id
6469 }

```

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

**sfragment**

```

6470 \keys_define:nn { document-structure / omgroupp }{
6471 id .str_set_x:N = \l__document_structure_omgroup_id_str,
6472 date .str_set_x:N = \l__document_structure_omgroup_date_str,
6473 creators .clist_set:N = \l__document_structure_omgroup_creators_clist,

```

```

6474 contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
6475 srccite .tl_set:N = \l__document_structure_omgroup_srccite_tl,
6476 type .tl_set:N = \l__document_structure_omgroup_type_tl,
6477 short .tl_set:N = \l__document_structure_omgroup_short_tl,
6478 display .tl_set:N = \l__document_structure_omgroup_display_tl,
6479 intro .tl_set:N = \l__document_structure_omgroup_intro_tl,
6480 imports .tl_set:N = \l__document_structure_omgroup_imports_tl,
6481 loadmodules .bool_set:N = \l__document_structure_omgroup_loadmodules_bool
6482 }
6483 \cs_new_protected:Nn \l__document_structure_omgroup_args:n {
6484 \str_clear:N \l__document_structure_omgroup_id_str
6485 \str_clear:N \l__document_structure_omgroup_date_str
6486 \clist_clear:N \l__document_structure_omgroup_creators_clist
6487 \clist_clear:N \l__document_structure_omgroup_contributors_clist
6488 \tl_clear:N \l__document_structure_omgroup_srccite_tl
6489 \tl_clear:N \l__document_structure_omgroup_type_tl
6490 \tl_clear:N \l__document_structure_omgroup_short_tl
6491 \tl_clear:N \l__document_structure_omgroup_display_tl
6492 \tl_clear:N \l__document_structure_omgroup_imports_tl
6493 \tl_clear:N \l__document_structure_omgroup_intro_tl
6494 \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
6495 \keys_set:nn { document-structure / omgrou } { #1 }
6496 }

```

\at@begin@omgroup we define a switch for numbering lines and a hook for the beginning of groups: The \at@begin@omgroup macro allows customization. It is run at the beginning of the omgrou, i.e. after the section heading.

```

6497 \newif\if@mainmatter\@mainmattertrue
6498 \newcommand\at@begin@omgroup[3][]{ }

```

Then we define a helper macro that takes care of the sectioning magic. It comes with its own key/value interface for customization.

```

6499 \keys_define:nn { document-structure / sectioning }{
6500 name .str_set_x:N = \l__document_structure_sect_name_str ,
6501 ref .str_set_x:N = \l__document_structure_sect_ref_str ,
6502 clear .bool_set:N = \l__document_structure_sect_clear_bool ,
6503 clear .default:n = {true} ,
6504 num .bool_set:N = \l__document_structure_sect_num_bool ,
6505 num .default:n = {true}
6506 }
6507 \cs_new_protected:Nn \l__document_structure_sect_args:n {
6508 \str_clear:N \l__document_structure_sect_name_str
6509 \str_clear:N \l__document_structure_sect_ref_str
6510 \bool_set_false:N \l__document_structure_sect_clear_bool
6511 \bool_set_false:N \l__document_structure_sect_num_bool
6512 \keys_set:nn { document-structure / sectioning } { #1 }
6513 }
6514 \newcommand\omdoc@sectioning[3][]{ }
6515 \l__document_structure_sect_args:n {#1 }
6516 \let\omdoc@sect@name\l__document_structure_sect_name_str
6517 \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
6518 \if@mainmatter% numbering not overridden by frontmatter, etc.
6519 \bool_if:NTF \l__document_structure_sect_num_bool {
6520 \omgroup@num{#2}{#3}

```

```

6521     }{
6522       \omgroup@nonum{#2}{#3}
6523     }
6524     \def\current@section@level{\omdoc@sect@name}
6525   \else
6526     \omgroup@nonum{#2}{#3}
6527   \fi
6528 }% 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.

```

6529 \newcommand\omgroup@redefine@addtocontents[1]{%
6530   %\edef\__document_structureimport{#1}%
6531   %\@for\@I:=\__document_structureimport\do{%
6532     %\edef\@path{\csname module@\@I @path\endcsname}%
6533     %\@ifundefined{tf@toc}\relax%
6534     %    {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
6535   %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
6536   %\def\addcontentsline##1##2##3{%
6537     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6538   %\else% hyperref.sty not loaded
6539   %\def\addcontentsline##1##2##3{%
6540     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6541   %\fi
6542 }% hypreref.sty loaded?

```

now the `omgroup` environment itself. This takes care of the table of contents via the helper macro above and then selects the appropriate sectioning command from `article.cls`. It also registers the current level of `omgroups` in the `\omgroup@level` counter.

```

6543 \newenvironment{sfragment}[2] []% keys, title
6544 {
6545   \__document_structure_omgroup_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.

```

6546 \stex_csl_to_imports:No \usemodule \l__document_structure_omgroup_imports_tl
6547
6548 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
6549   \omgroup@redefine@addtocontents{
6550     %\@ifundefined{module@id}\used@modules%
6551     %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}
6552   }
6553 }

```

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

```

6554 \int_incr:N\l__document_structure_section_level_int
6555 \ifcase\l__document_structure_section_level_int
6556   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
6557   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
6558   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
6559   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
6560   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
6561   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
6562   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{paragr

```



```

6563 \fi
6564 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
6565 \str_if_empty:NF \l__document_structure_omgroup_id_str {
6566   \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
6567 }
6568 }% for customization
6569 {}

```

and finally, we localize the sections

```

6570 \newcommand\omdoc@part@kw{Part}
6571 \newcommand\omdoc@chapter@kw{Chapter}
6572 \newcommand\omdoc@section@kw{Section}
6573 \newcommand\omdoc@subsection@kw{Subsection}
6574 \newcommand\omdoc@subsubsection@kw{Subsubsection}
6575 \newcommand\omdoc@paragraph@kw{paragraph}
6576 \newcommand\omdoc@subparagraph@kw{subparagraph}

```

## 37.3 Front and Backmatter

Index markup is provided by the `omtext` package [Koh20c], so in the `document-structure` package we only need to supply the corresponding `\printindex` command, if it is not already defined

`\printindex`

```

6577 \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).

```

6578 \cs_if_exist:NTF\frontmatter{
6579   \let\__document_structure_orig_frontmatter\frontmatter
6580   \let\frontmatter\relax
6581 }{
6582   \tl_set:Nn\__document_structure_orig_frontmatter{
6583     \clearpage
6584     \@mainmatterfalse
6585     \pagenumbering{roman}
6586   }
6587 }
6588 \cs_if_exist:NTF\backmatter{
6589   \let\__document_structure_orig_backmatter\backmatter
6590   \let\backmatter\relax
6591 }{
6592   \tl_set:Nn\__document_structure_orig_backmatter{
6593     \clearpage
6594     \@mainmatterfalse
6595     \pagenumbering{roman}
6596   }
6597 }

```

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.

```

6598 \newenvironment{frontmatter}{
6599   \_document_structure_orig_frontmatter
6600 }{
6601   \cs_if_exist:NTF\mainmatter{
6602     \mainmatter
6603   }{
6604     \clearpage
6605     \@mainmattertrue
6606     \pagenumbering{arabic}
6607   }
6608 }

```

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

```

6609 \newenvironment{backmatter}{
6610   \_document_structure_orig_backmatter
6611 }{
6612   \cs_if_exist:NTF\mainmatter{
6613     \mainmatter
6614   }{
6615     \clearpage
6616     \@mainmattertrue
6617     \pagenumbering{arabic}
6618   }
6619 }

```

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

```

6620 \@mainmattertrue\pagenumbering{arabic}

```

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

```

6621 \def \c__document_structure_document_str{document}
6622 \newcommand\afterprematurestop{}
6623 \def\prematurestop@endomgroup{
6624   \unless\ifx\@currenvir\c__document_structure_document_str
6625     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
6626       \expandafter\prematurestop@endomgroup
6627     \fi
6628   }
6629 \providecommand\prematurestop{
6630   \message{Stopping~sTeX~processing~prematurely}
6631   \prematurestop@endomgroup
6632   \afterprematurestop
6633   \end{document}
6634 }

```

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

## 37.4 Global Variables

`\setSGvar` set a global variable

```
6635 \RequirePackage{etoolbox}
6636 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}
```

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

`\useSGvar` use a global variable

```
6637 \newrobustcmd\useSGvar[1]{%
6638   \@ifundefined{sTeX@Gvar@#1}
6639   {\PackageError{document-structure}
6640    {The sTeX Global variable #1 is undefined}
6641    {set it with \protect\setSGvar}}
6642   \@nameuse{sTeX@Gvar@#1}}
```

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

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

```
6643 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
6644   \@ifundefined{sTeX@Gvar@#1}
6645   {\PackageError{document-structure}
6646    {The sTeX Global variable #1 is undefined}
6647    {set it with \protect\setSGvar}}
6648   {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}
```

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

## 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.

```
6649 \*cls)
6650 \@@=notesslides)
6651 \ProvidesExplClass{notesslides}{2022/02/28}{3.1.0}{notesslides Class}
6652 \RequirePackage{13keys2e}
6653
6654 \keys_define:nn{notesslides / cls}{
6655   class .str_set_x:N = \c__notesslides_class_str,
6656   notes .bool_set:N = \c__notesslides_notes_bool ,
6657   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
6658   docopt .str_set_x:N = \c__notesslides_docopt_str,
6659   unknown .code:n = {
6660     \PassOptionsToPackage{\CurrentOption}{document-structure}
6661     \PassOptionsToClass{\CurrentOption}{beamer}
6662     \PassOptionsToPackage{\CurrentOption}{notesslides}
6663     \PassOptionsToPackage{\CurrentOption}{stex}
6664   }
6665 }
6666 \ProcessKeysOptions{ notesslides / cls }
6667
6668 \str_if_empty:NF \c__notesslides_class_str {
6669   \PassOptionsToPackage{class=\c__notesslides_class_str}{document-structure}
6670 }
6671
6672 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{book}{
6673   \PassOptionsToPackage{defaulttopsect=part}{notesslides}
6674 }
6675 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{report}{
6676   \PassOptionsToPackage{defaulttopsect=part}{notesslides}
6677 }
6678
6679 \RequirePackage{stex}
```

```

6680 \stex_html_backend:T {
6681   \bool_set_true:N\c__notesslides_notes_bool
6682 }
6683
6684 \bool_if:NTF \c__notesslides_notes_bool {
6685   \PassOptionsToPackage{notes=true}{notesslides}
6686 }{
6687   \PassOptionsToPackage{notes=false}{notesslides}
6688 }
6689 \</cls>

```

now we do the same for the notesslides package.

```

6690 \*package>
6691 \ProvidesExplPackage{notesslides}{2022/02/28}{3.1.0}{notesslides Package}
6692 \RequirePackage{13keys2e}
6693
6694 \keys_define:nn{notesslides / pkg}{
6695   topsect          .str_set_x:N = \c__notesslides_topsect_str,
6696   defaulttopsect   .str_set_x:N = \c__notesslides_defaulttopsec_str,
6697   notes            .bool_set:N  = \c__notesslides_notes_bool ,
6698   slides           .code:n       = { \bool_set_false:N \c__notesslides_notes_bool },
6699   sectocframes     .bool_set:N  = \c__notesslides_sectocframes_bool ,
6700   frameimages      .bool_set:N  = \c__notesslides_frameimages_bool ,
6701   fiboxed          .bool_set:N  = \c__notesslides_fiboxed_bool ,
6702   nopproblems      .bool_set:N  = \c__notesslides_nopproblems_bool,
6703   unknown          .code:n       = {
6704     \PassOptionsToClass{\CurrentOption}{stex}
6705     \PassOptionsToClass{\CurrentOption}{tikzinput}
6706   }
6707 }
6708 \ProcessKeysOptions{ notesslides / pkg }
6709
6710 \RequirePackage{stex}
6711 \stex_html_backend:T {
6712   \bool_set_true:N\c__notesslides_notes_bool
6713 }
6714
6715 \newif\ifnotes
6716 \bool_if:NTF \c__notesslides_notes_bool {
6717   \notesttrue
6718 }{
6719   \notesfalse
6720 }
6721

```

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

```

6722 \str_if_empty:NTF \c__notesslides_topsect_str {
6723   \str_set_eq:NN \__notesslides_topsect \c__notesslides_defaulttopsec_str
6724 }{
6725   \str_set_eq:NN \__notesslides_topsect \c__notesslides_topsect_str
6726 }
6727 \PassOptionsToPackage{topsect=\__notesslides_topsect}{document-structure}
6728 \</package>

```

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

```

6729 <*cls>
6730 \bool_if:NTF \c__notesslides_notes_bool {
6731   \str_if_empty:NT \c__notesslides_class_str {
6732     \str_set:Nn \c__notesslides_class_str {article}
6733   }
6734   \exp_after:wN\LoadClass\exp_after:wN[\c__notesslides_docostr]
6735     {\c__notesslides_class_str}
6736 }{
6737   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
6738   \newcounter{Item}
6739   \newcounter{paragraph}
6740   \newcounter{subparagraph}
6741   \newcounter{Hfootnote}
6742 }
6743 \RequirePackage{document-structure}

```

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

```

6744 \RequirePackage{notesslides}
6745 </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.

```

6746 <*package>
6747 \bool_if:NT \c__notesslides_notes_bool {
6748   \RequirePackage{a4wide}
6749   \RequirePackage{marginnote}
6750   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
6751   \RequirePackage{mdframed}
6752   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
6753   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
6754 }
6755 \RequirePackage{stex-tikzinput}
6756 \RequirePackage{etoolbox}
6757 \RequirePackage{amssymb}
6758 \RequirePackage{amsmath}
6759 \RequirePackage{comment}
6760 \RequirePackage{textcomp}
6761 \RequirePackage{url}
6762 \RequirePackage{graphicx}
6763 \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>13</sup>

```

6764 \bool_if:NT \c__notesslides_notes_bool {
6765   \renewcommand\usetheme[2] [] {\usepackage[#1]{beamernotestheme#2}}
6766 }
6767
6768
6769 \NewDocumentCommand \libusetheme {0{} m} {
6770   \bool_if:NTF \c__notesslides_notes_bool {
6771     \libusepackage[#1]{beamernotestheme#2}
6772   }{
6773     \libusepackage[#1]{beamertheme#2}
6774   }
6775 }

```

We define the sizes of slides in the notes. Somehow, we cannot get by with the same here.

```

6776 \newcounter{slide}
6777 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
6778 \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.

```

6779 \bool_if:NTF \c__notesslides_notes_bool {
6780   \renewenvironment{note}{\ignorespaces}{}
6781 }{
6782   \excludacomment{note}
6783 }

```

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.

```

6784 \bool_if:NT \c__notesslides_notes_bool {
6785   \newlength{\slideframewidth}
6786   \setlength{\slideframewidth}{1.5pt}

```

**frame** We first define the keys.

```

6787 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
6788   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
6789     \bool_set_true:N #1
6790   }{
6791     \bool_set_false:N #1
6792   }
6793 }
6794 \keys_define:nn{notesslides / frame}{
6795   label .str_set_x:N = \l__notesslides_frame_label_str,
6796   allowframebreaks .code:n = {
6797     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
6798   },
6799   allowdisplaybreaks .code:n = {

```

---

<sup>13</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.

```

6800     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_allowdisplaybreaks\_bool { #1 }
6801 },
6802 fragile .code:n = {
6803     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_fragile\_bool { #1 }
6804 },
6805 shrink .code:n = {
6806     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_shrink\_bool { #1 }
6807 },
6808 squeeze .code:n = {
6809     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_squeeze\_bool { #1 }
6810 },
6811 t .code:n = {
6812     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_t\_bool { #1 }
6813 },
6814 }
6815 \cs\_new\_protected:Nn \_notesslides\_frame\_args:n {
6816     \str\_clear:N \l\_notesslides\_frame\_label\_str
6817     \bool\_set\_true:N \l\_notesslides\_frame\_allowframebreaks\_bool
6818     \bool\_set\_true:N \l\_notesslides\_frame\_allowdisplaybreaks\_bool
6819     \bool\_set\_true:N \l\_notesslides\_frame\_fragile\_bool
6820     \bool\_set\_true:N \l\_notesslides\_frame\_shrink\_bool
6821     \bool\_set\_true:N \l\_notesslides\_frame\_squeeze\_bool
6822     \bool\_set\_true:N \l\_notesslides\_frame\_t\_bool
6823     \keys\_set:nn { notesslides / frame }{ #1 }
6824 }

```

We define the environment, read them, and construct the slide number and label.

```

6825 \renewenvironment{frame}[1][]{
6826     \_notesslides\_frame\_args:n{#1}
6827     \sffamily
6828     \stepcounter{slide}
6829     \def\@currentlabel{\theslide}
6830     \str\_if\_empty:NF \l\_notesslides\_frame\_label\_str {
6831         \label{\l\_notesslides\_frame\_label\_str}
6832     }

```

We redefine the `itemize` environment so that it looks more like the one in `beamer`.

```

6833 \def\itemize@level{outer}
6834 \def\itemize@outer{outer}
6835 \def\itemize@inner{inner}
6836 \renewcommand\newpage{\addtocounter{framenum}{1}}
6837 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
6838 \renewenvironment{itemize}{
6839     \ifx\itemize@level\itemize@outer
6840         \def\itemize@label{\$ \rhd \$}
6841     \fi
6842     \ifx\itemize@level\itemize@inner
6843         \def\itemize@label{\$ \scriptstyle \rhd \$}
6844     \fi
6845     \begin{list}
6846     {\itemize@label}
6847     {\setlength{\labelsep}{.3em}
6848     \setlength{\labelwidth}{.5em}
6849     \setlength{\leftmargin}{1.5em}
6850     }

```



```

6851     \edef\itemize@level{\itemize@inner}
6852   }{
6853     \end{list}
6854   }

```

We create the box with the `mdframed` environment from the `equinymous` package.

```

6855     \stex_html_backend:TF {
6856       \begin{stex_annotate_env}{frame}{}\vbox\bgroup
6857     }{
6858       \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwid
6859     ]
6860   }{
6861     \stex_html_backend:TF {
6862       \miko@slidelabel\egroup\end{stex_annotate_env}
6863     }\medskip\miko@slidelabel\end{mdframed}}
6864   }

```

Now, we need to redefine the `frametitle` (we are still in course notes mode).

`\frametitle`

```

6865     \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip}
6866   }

```

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

EdN:14

`\pause` 14

```

6867 \bool_if:NT \c__notesslides_notes_bool {
6868   \newcommand\pause{}
6869 }

```

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

`nparagraph`

```

6870 \bool_if:NTF \c__notesslides_notes_bool {
6871   \newenvironment{nparagraph}[1] []{\begin{sparagraph}[#1]}\end{sparagraph}}
6872 }{
6873   \excludecomment{nparagraph}
6874 }

```

`nfragment`

```

6875 \bool_if:NTF \c__notesslides_notes_bool {
6876   \newenvironment{nfragment}[2] []{\begin{sfragment}[#1]{#2}}\end{sfragment}}
6877 }{
6878   \excludecomment{nfragment}
6879 }

```

`ndefinition`

```

6880 \bool_if:NTF \c__notesslides_notes_bool {
6881   \newenvironment{ndefinition}[1] []{\begin{sdefinition}[#1]}\end{sdefinition}}
6882 }{
6883   \excludecomment{ndefinition}
6884 }

```

---

<sup>14</sup>EDNOTE: MK: fake it in notes mode for now

nassertion

```

6885 \bool_if:NTF \c_notesslides_notes_bool {
6886   \newenvironment{nassertion}[1][\begin{sassertion}[#1]}\end{sassertion}}
6887 }{
6888   \excludecomment{nassertion}
6889 }

```

nsproof

```

6890 \bool_if:NTF \c_notesslides_notes_bool {
6891   \newenvironment{nproof}[2][\begin{sproof}[#1]{#2}]\end{sproof}}
6892 }{
6893   \excludecomment{nproof}
6894 }

```

nexample

```

6895 \bool_if:NTF \c_notesslides_notes_bool {
6896   \newenvironment{nexample}[1][\begin{sexample}[#1]}\end{sexample}}
6897 }{
6898   \excludecomment{nexample}
6899 }

```

\inputref@\*skip We customize the hooks for in \inputref.

```

6900 \def\inputref@preskip{\smallskip}
6901 \def\inputref@postskip{\medskip}

```

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

\inputref\*

```

6902 \let\orig@inputref\inputref
6903 \def\inputref{\@ifstar\ninputref\orig@inputref}
6904 \newcommand\ninputref[2][\{
6905   \bool_if:NT \c_notesslides_notes_bool {
6906     \orig@inputref[#1]{#2}
6907   }
6908 }

```

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

## 38.3 Header and Footer Lines

Now, we set up the infrastructure for the footer line of the slides, we use boxes for the logos, so that they are only loaded once, that considerably speeds up processing.

\setslidelogo The default logo is the  $\text{\TeX}$  logo. Customization can be done by \setslidelogo{<logo name>}.

```

6909 \newlength{\slidelogoheight}
6910
6911 \bool_if:NTF \c_notesslides_notes_bool {
6912   \setlength{\slidelogoheight}{.4cm}
6913 }{
6914   \setlength{\slidelogoheight}{1cm}
6915 }
6916 \newsavebox{\slidelogo}

```

```

6917 \sbox{\slidelogo}{\TeX}
6918 \newrobustcmd{\setslidelogo}[1]{
6919   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
6920 }

```

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

**\setsource** `\source` stores the writer's name. By default it is *Michael Kohlhase* since he is the main user and designer of this package. `\setsource{<name>}` can change the writer's name.

```

6921 \def\source{Michael Kohlhase}% customize locally
6922 \newrobustcmd{\setsource}[1]{\def\source{#1}}

```

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

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

```

6923 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
6924 \newsavebox{\cclogo}
6925 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
6926 \newif\ifcchref\cchreffalse
6927 \AtBeginDocument{
6928   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
6929 }
6930 \def\licensing{
6931   \ifcchref
6932     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
6933   \else
6934     {\usebox{\cclogo}}
6935   \fi
6936 }
6937 \newrobustcmd{\setlicensing}[2][]{
6938   \def\@url{#1}
6939   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
6940   \ifx\@url\@empty
6941     \def\licensing{{\usebox{\cclogo}}}
6942   \else
6943     \def\licensing{
6944       \ifcchref
6945         \href{#1}{\usebox{\cclogo}}
6946       \else
6947         {\usebox{\cclogo}}
6948       \fi
6949     }
6950   \fi
6951 }

```

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

EdN:15 **\slidelabel** Now, we set up the slide label for the article mode.<sup>15</sup>

```

6952 \newrobustcmd\miko@slidelabel{
6953   \vbox to \slidelogoheight{

```

---

<sup>15</sup>EdNOTE: see that we can use the themes for the slides some day. This is all fake.

```

6954 \vss\hbox to \slidewidth
6955 {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
6956 }
6957 }

```

(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.

```

6958 \def\Gin@mhrepos{}
6959 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
6960 \define@key{Gin}{label}{\def\currentlabel{\arabic{slide}}\label{#1}}
6961 \newrobustcmd\frameimage[2][{}{
6962   \stepcounter{slide}
6963   \bool_if:NT \c__notesslides_frameimages_bool {
6964     \def\Gin@ewidth{} \setkeys{Gin}{#1}
6965     \bool_if:NF \c__notesslides_notes_bool { \vfill }
6966     \begin{center}
6967       \bool_if:NTF \c__notesslides_fiboxed_bool {
6968         \fbox{
6969           \ifx\Gin@ewidth\@empty
6970             \ifx\Gin@mhrepos\@empty
6971               \mhgraphics[width=\slidewidth,#1]{#2}
6972             \else
6973               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
6974             \fi
6975           \else% Gin@ewidth empty
6976             \ifx\Gin@mhrepos\@empty
6977               \mhgraphics[#1]{#2}
6978             \else
6979               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
6980             \fi
6981           \fi% Gin@ewidth empty
6982         }
6983       }{
6984         \ifx\Gin@ewidth\@empty
6985           \ifx\Gin@mhrepos\@empty
6986             \mhgraphics[width=\slidewidth,#1]{#2}
6987           \else
6988             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
6989           \fi
6990         \ifx\Gin@mhrepos\@empty
6991           \mhgraphics[#1]{#2}
6992         \else
6993           \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
6994         \fi
6995         \fi% Gin@ewidth empty
6996       }
6997     \end{center}
6998   \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
6999   \bool_if:NF \c__notesslides_notes_bool { \vfill }

```

```

7000 }
7001 } % ifmks@sty@frameimages

```

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

## 38.5 Colors and Highlighting

We first specify sans serif fonts as the default.

```

7002 \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

```

7003 \AddToHook{begindocument}{
7004   \definecolor{green}{rgb}{0,.5,0}
7005   \definecolor{purple}{cmk}{.3,1,0,.17}
7006 }

```

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`.

```

7007 % \def\STpresent#1{\textcolor{blue}{#1}}
7008 \def\defemph#1{\textcolor{magenta}{#1}}
7009 \def\symrefemph#1{\textcolor{cyan}{#1}}
7010 \def\compemph#1{\textcolor{blue}{#1}}
7011 \def\titleemph#1{\textcolor{blue}{#1}}
7012 \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.

```

7013 \pgfdeclareimage[width=.8em]{miko@small@dbend}{stex-dangerous-bend}
7014 \def\smalltextwarning{
7015   \pgfuseimage{miko@small@dbend}
7016   \xspace
7017 }
7018 \pgfdeclareimage[width=1.2em]{miko@dbend}{stex-dangerous-bend}
7019 \newrobustcmd\textwarning{
7020   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
7021   \xspace
7022 }
7023 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{stex-dangerous-bend}
7024 \newrobustcmd\bigtextwarning{
7025   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
7026   \xspace
7027 }

```

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

```

7028 \newrobustcmd\putgraphicsat[3]{
7029   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
7030 }
7031 \newrobustcmd\putat[2]{
7032   \begin{picture}(0,0)\put(#1){#2}\end{picture}
7033 }

```

## 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`.

```

7034 \bool_if:NT \c__notesslides_sectocframes_bool {
7035   \str_if_eq:VnTF \__notesslidesstopsect{part}{
7036     \newcounter{chapter}\counterwithin*{section}{chapter}
7037   }{
7038     \str_if_eq:VnTF \__notesslidesstopsect{chapter}{
7039       \newcounter{chapter}\counterwithin*{section}{chapter}
7040     }
7041   }
7042 }
```

`\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

7043 \def\part@prefix{}
7044 \@ifpackageloaded{document-structure}{
7045   \str_case:VnF \__notesslidesstopsect {
7046     {part}{
7047       \int_set:Nn \l_document_structure_section_level_int {0}
7048       \def\thesection{\arabic{chapter}.\arabic{section}}
7049       \def\part@prefix{\arabic{chapter}.}
7050     }
7051     {chapter}{
7052       \int_set:Nn \l_document_structure_section_level_int {1}
7053       \def\thesection{\arabic{chapter}.\arabic{section}}
7054       \def\part@prefix{\arabic{chapter}.}
7055     }
7056   }{
7057     \int_set:Nn \l_document_structure_section_level_int {2}
7058     \def\part@prefix{}
7059   }
7060 }
7061
7062 \bool_if:NF \c__notesslides_notes_bool { % only in slides
```

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

The new counters are used in the `omgroup` environment that chooses the L<sup>A</sup>T<sub>E</sub>X sectioning macros according to `\section@level`.

**sfragment**

```

7063 \renewenvironment{sfragment}[2][
7064   \__document_structure_omgroup_args:n { #1 }
7065   \int_incr:N \l_document_structure_section_level_int
7066   \bool_if:NT \c__notesslides_sectocframes_bool {
7067     \stepcounter{slide}
7068     \begin{frame}[noframenumbering]
7069     \vfill\Large\centering
7070     \red{
7071       \ifcase\l_document_structure_section_level_int\or
```

```

7072         \stepcounter{part}
7073         \def\__notesslideslabel{\{ \omdoc@part@kw\}~\Roman{part}}
7074         \def\currentsectionlevel{\omdoc@part@kw}
7075     \or
7076         \stepcounter{chapter}
7077         \def\__notesslideslabel{\{ \omdoc@chapter@kw\}~\arabic{chapter}}
7078         \def\currentsectionlevel{\omdoc@chapter@kw}
7079     \or
7080         \stepcounter{section}
7081         \def\__notesslideslabel{\part@prefix\arabic{section}}
7082         \def\currentsectionlevel{\omdoc@section@kw}
7083     \or
7084         \stepcounter{subsection}
7085         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
7086         \def\currentsectionlevel{\omdoc@subsection@kw}
7087     \or
7088         \stepcounter{subsubsection}
7089         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
7090         \def\currentsectionlevel{\omdoc@subsubsection@kw}
7091     \or
7092         \stepcounter{paragraph}
7093         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
7094         \def\currentsectionlevel{\omdoc@paragraph@kw}
7095     \else
7096         \def\__notesslideslabel{}
7097         \def\currentsectionlevel{\omdoc@paragraph@kw}
7098     \fi% end ifcase
7099     \__notesslideslabel%\sref@label@id\__notesslideslabel
7100     \quad #2%
7101 }%
7102 \vfill%
7103 \end{frame}%
7104 }
7105 \str_if_empty:NF \l__document_structure_omgroup_id_str {
7106     \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
7107 }
7108 }{}
7109 }

```

We set up a beamer template for theorems like ams style, but without a block environment.

```

7110 \def\inserttheorembodyfont{\normalfont}
7111 %\bool_if:NF \c__notesslides_notes_bool {
7112 % \defbeamertemplate{theorem begin}{miko}
7113 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
7114 % \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
7115 % \inserttheorempunctuation\inserttheorembodyfont\xspace}
7116 % \defbeamertemplate{theorem end}{miko}{\}

```

and we set it as the default one.

```

7117 % \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.

```

7118 % \expandafter\def\csname Parent2\endcsname{}

```

```

7119 %}
7120
7121 \AddToHook{begindocument}{% this does not work for some reasons
7122   \setbeamertemplate{theorems}[ams style]
7123 }
7124 \bool_if:NT \c__notesslides_notes_bool {
7125   \renewenvironment{columns}[1][]{%
7126     \par\noindent%
7127     \begin{minipage}%
7128       \slidewidth\centering\leavevmode%
7129   }{%
7130     \end{minipage}\par\noindent%
7131   }%
7132   \newsavebox\columnbox%
7133   \renewenvironment<>{column}[2][]{%
7134     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
7135   }{%
7136     \end{minipage}\end{lrbox}\usebox\columnbox%
7137   }%
7138 }
7139 \bool_if:NTF \c__notesslides_noproblems_bool {
7140   \newenvironment{problems}{}{}
7141 }{
7142   \excludcomment{problems}
7143 }

```

## 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.

```

7144 \gdef\printexcursions{}
7145 \newcommand\excursionref[2]{% label, text
7146   \bool_if:NT \c__notesslides_notes_bool {
7147     \begin{sparagraph}[title=Excursion]
7148       #2 \sref[fallback=the appendix]{#1}.
7149     \end{sparagraph}
7150   }
7151 }
7152 \newcommand\activate@excursion[2][{}{
7153   \gappto\printexcursions{\inputref[#1]{#2}}
7154 }
7155 \newcommand\excursion[4][{}{repos, label, path, text
7156   \bool_if:NT \c__notesslides_notes_bool {
7157     \activate@excursion[#1]{#3}\excursionref{#2}{#4}
7158   }
7159 }

```

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

**\excursiongroup**

```

7160 \keys_define:nn{notesslides / excursiongroup }{

```



```

7161 id .str_set_x:N = \l_notesslides_excursion_id_str,
7162 intro .tl_set:N = \l_notesslides_excursion_intro_tl,
7163 mhrepos .str_set_x:N = \l_notesslides_excursion_mhrepos_str
7164 }
7165 \cs_new_protected:Nn \l_notesslides_excursion_args:n {
7166 \tl_clear:N \l_notesslides_excursion_intro_tl
7167 \str_clear:N \l_notesslides_excursion_id_str
7168 \str_clear:N \l_notesslides_excursion_mhrepos_str
7169 \keys_set:nn {notesslides / excursiongroup} { #1 }
7170 }
7171 \newcommand\excursiongroup[1][ ]{
7172 \l_notesslides_excursion_args:n{ #1 }
7173 \ifdefempty\printexcursions{}% only if there are excursions
7174 {\begin{note}
7175 \begin{sfragment}[#1]{Excursions}%
7176 \ifdefempty\l_notesslides_excursion_intro_tl}{
7177 \inputref[\l_notesslides_excursion_mhrepos_str]{
7178 \l_notesslides_excursion_intro_tl
7179 }
7180 }
7181 \printexcursions%
7182 \end{sfragment}
7183 \end{note}}
7184 }
7185 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{\fi
7186 \</package>

```

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

## 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.

```
7187 <*package>
7188 <@@=problems>
7189 \ProvidesExplPackage{problem}{2022/02/26}{3.0.1}{Semantic Markup for Problems}
7190 \RequirePackage{13keys2e,stex}
7191
7192 \keys_define:nn { problem / pkg }{
7193   notes      .default:n    = { true },
7194   notes      .bool_set:N   = \c__problems_notes_bool,
7195   gnotes     .default:n    = { true },
7196   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
7197   hints      .default:n    = { true },
7198   hints      .bool_set:N   = \c__problems_hints_bool,
7199   solutions  .default:n    = { true },
7200   solutions  .bool_set:N   = \c__problems_solutions_bool,
7201   pts        .default:n    = { true },
7202   pts        .bool_set:N   = \c__problems_pts_bool,
7203   min        .default:n    = { true },
7204   min        .bool_set:N   = \c__problems_min_bool,
7205   boxed      .default:n    = { true },
7206   boxed      .bool_set:N   = \c__problems_boxed_bool,
7207   unknown    .code:n       = {}
7208 }
7209 \newif\ifsolutions
7210
7211 \ProcessKeysOptions{ problem / pkg }
7212 \bool_if:NTF \c__problems_solutions_bool {
7213   \solutionstrue
7214 }{
7215   \solutionsfalse
7216 }
```

Then we make sure that the necessary packages are loaded (in the right versions).

```
7217 \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.

```
7218 \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.

```
7219 \def\prob@problem@kw{Problem}
7220 \def\prob@solution@kw{Solution}
7221 \def\prob@hint@kw{Hint}
7222 \def\prob@note@kw{Note}
7223 \def\prob@gnote@kw{Grading}
7224 \def\prob@pt@kw{pt}
7225 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
7226 \AddToHook{begindocument}{
7227   \ltx@ifpackageloaded{babel}{
7228     \makeatletter
7229     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7230     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7231       \input{problem-ngerman.ldf}
7232     }
7233     \clist_if_in:NnT \l_tmpa_clist {finnish}{
7234       \input{problem-finnish.ldf}
7235     }
7236     \clist_if_in:NnT \l_tmpa_clist {french}{
7237       \input{problem-french.ldf}
7238     }
7239     \clist_if_in:NnT \l_tmpa_clist {russian}{
7240       \input{problem-russian.ldf}
7241     }
7242     \makeatother
7243   }{}
7244 }
```

## 39.2 Problems and Solutions

We now prepare the KeyVal support for problems. The key macros just set appropriate internal macros.

```
7245 \keys_define:nn{ problem / problem }{
7246   id      .str_set_x:N = \l__problems_prob_id_str,
7247   pts     .tl_set:N    = \l__problems_prob_pts_tl,
7248   min     .tl_set:N    = \l__problems_prob_min_tl,
7249   title   .tl_set:N    = \l__problems_prob_title_tl,
7250   type    .tl_set:N    = \l__problems_prob_type_tl,
7251   imports .tl_set:N    = \l__problems_prob_imports_tl,
7252   name    .str_set_x:N = \l__problems_prob_name_str,
7253   refnum  .int_set:N   = \l__problems_prob_refnum_int
```

```

7254 }
7255 \cs_new_protected:Nn \__problems_prob_args:n {
7256   \str_clear:N \l__problems_prob_id_str
7257   \str_clear:N \l__problems_prob_name_str
7258   \tl_clear:N \l__problems_prob_pts_tl
7259   \tl_clear:N \l__problems_prob_min_tl
7260   \tl_clear:N \l__problems_prob_title_tl
7261   \tl_clear:N \l__problems_prob_type_tl
7262   \tl_clear:N \l__problems_prob_imports_tl
7263   \int_zero_new:N \l__problems_prob_refnum_int
7264   \keys_set:nn { problem / problem }{ #1 }
7265   \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
7266     \let\l__problems_prob_refnum_int\undefined
7267   }
7268 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

7269 \newcounter{problem}[section]
7270 \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.

```

7271 \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

```

7272 \newcommand\prob@number{
7273   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
7274     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
7275   }{
7276     \int_if_exist:NTF \l__problems_prob_refnum_int {
7277       \prob@label{\int_use:N \l__problems_prob_refnum_int }
7278     }{
7279       \prob@label\theproblem
7280     }
7281   }
7282 }

```

(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.

```

7283 \newcommand\prob@title[3]{%
7284   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
7285     #2 \l__problems_inclprob_title_tl #3
7286   }{
7287     \tl_if_exist:NTF \l__problems_prob_title_tl {
7288       #2 \l__problems_prob_title_tl #3
7289     }{
7290       #1

```

```

7291     }
7292   }
7293 }

```

(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.

```

7294 \def\prob@heading{
7295   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
7296   %\sref@label@id{\prob@problem@kw~\prob@number}{~}
7297 }

```

(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`

```

7298 \newenvironment{sproblem}[1][{}]{
7299   \__problems_prob_args:n{#1}%\sref@target%
7300   \@in@omtexttrue% we are in a statement (for inline definitions)
7301   \stepcounter{problem}\record@problem
7302   \def\current@section@level{\prob@problem@kw}
7303
7304   \str_if_empty:NT \l__problems_prob_name_str {
7305     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
7306     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
7307     \seq_get_left:NN \l_tmpa_seq \l__problems_prob_name_str
7308   }
7309   \exp_args:Nno\stex_module_setup:nn{type=problem}\l__problems_prob_name_str
7310
7311   \stex_reactivate_macro:N \STEXexport
7312   \stex_reactivate_macro:N \importmodule
7313   \stex_reactivate_macro:N \symdecl
7314   \stex_reactivate_macro:N \notation
7315   \stex_reactivate_macro:N \symdef
7316
7317   \stex_if_do_html:T{
7318     \begin{stex_annotate_env} {problem} {
7319       \l_stex_module_ns_str ? \l_stex_module_name_str
7320     }
7321
7322     \stex_annotate_invisible:nnn{header}{~}{~} {
7323       \stex_annotate:nnn{language}{~} \l_stex_module_lang_str }{~}
7324       \stex_annotate:nnn{signature}{~} \l_stex_module_sig_str }{~}
7325       \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
7326         \stex_annotate:nnn{metatheory}{~} \l_stex_module_meta_str }{~}
7327     }
7328   }
7329 }
7330

```

```

7331 \stex_csl_to_imports:No \importmodule \l__problems_prob_imports_tl
7332
7333
7334 \tl_if_exist:NTF \l__problems_inclprob_type_tl {
7335   \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
7336 }{
7337   \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
7338 }
7339 \str_if_exist:NTF \l__problems_inclprob_id_str {
7340   \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
7341 }{
7342   \str_set_eq:NN \sproblemid \l__problems_prob_id_str
7343 }
7344
7345
7346 \stex_if_smsmode:F {
7347   \clist_set:No \l_tmpa_clist \sproblemtype
7348   \tl_clear:N \l_tmpa_tl
7349   \clist_map_inline:Nn \l_tmpa_clist {
7350     \tl_if_exist:cT {__problems_sproblem_##1_start:}{
7351       \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_start:}}
7352     }
7353   }
7354   \tl_if_empty:NTF \l_tmpa_tl {
7355     \__problems_sproblem_start:
7356   }{
7357     \l_tmpa_tl
7358   }
7359 }
7360 \stex_ref_new_doc_target:n \sproblemid
7361 \stex_smsmode_do:
7362 }{
7363   \__stex_modules_end_module:
7364   \stex_if_smsmode:F{
7365     \clist_set:No \l_tmpa_clist \sproblemtype
7366     \tl_clear:N \l_tmpa_tl
7367     \clist_map_inline:Nn \l_tmpa_clist {
7368       \tl_if_exist:cT {__problems_sproblem_##1_end:}{
7369         \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_end:}}
7370       }
7371     }
7372     \tl_if_empty:NTF \l_tmpa_tl {
7373       \__problems_sproblem_end:
7374     }{
7375       \l_tmpa_tl
7376     }
7377   }
7378   \stex_if_do_html:T{
7379     \end{stex_annotate_env}
7380   }
7381
7382   \smallskip
7383 }
7384

```

```

7385 \seq_put_right:Nx\g_stex_smsmode_allowedenvs_seq{\tl_to_str:n{sproblem}}
7386
7387
7388
7389 \cs_new_protected:Nn \__problems_sproblem_start: {
7390   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
7391 }
7392 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
7393
7394 \newcommand\stexpatchproblem[3][] {
7395   \str_set:Nx \l_tmpa_str{ #1 }
7396   \str_if_empty:NTF \l_tmpa_str {
7397     \tl_set:Nn \__problems_sproblem_start: { #2 }
7398     \tl_set:Nn \__problems_sproblem_end: { #3 }
7399   }{
7400     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
7401     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
7402   }
7403 }
7404
7405
7406 \bool_if:NT \c__problems_boxed_bool {
7407   \surroundwithhmdframed{problem}
7408 }

```

**\record@problem** This macro records information about the problems in the \*.aux file.

```

7409 \def\record@problem{
7410   \protected@write\@auxout{}
7411   {
7412     \string\@problem{\prob@number}
7413     {
7414       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7415         \l__problems_inclprob_pts_tl
7416       }{
7417         \l__problems_prob_pts_tl
7418       }
7419     }%
7420     {
7421       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7422         \l__problems_inclprob_min_tl
7423       }{
7424         \l__problems_prob_min_tl
7425       }
7426     }
7427   }
7428 }

```

(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).

```

7429 \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 its own keys that we need to define first.

```

7430 \keys_define:nn { problem / solution }{
7431   id          .str_set_x:N = \l__problems_solution_id_str ,
7432   for         .tl_set:N    = \l__problems_solution_for_tl ,
7433   height      .dim_set:N   = \l__problems_solution_height_dim ,
7434   creators    .clist_set:N = \l__problems_solution_creators_clist ,
7435   contributors .clist_set:N = \l__problems_solution_contributors_clist ,
7436   srccite     .tl_set:N    = \l__problems_solution_srccite_tl
7437 }
7438 \cs_new_protected:Nn \__problems_solution_args:n {
7439   \str_clear:N \l__problems_solution_id_str
7440   \tl_clear:N \l__problems_solution_for_tl
7441   \tl_clear:N \l__problems_solution_srccite_tl
7442   \clist_clear:N \l__problems_solution_creators_clist
7443   \clist_clear:N \l__problems_solution_contributors_clist
7444   \dim_zero:N \l__problems_solution_height_dim
7445   \keys_set:nn { problem / solution }{ #1 }
7446 }

```

the next step is to define a helper macro that does what is needed to start a solution.

```

7447 \newcommand\@startsolution[1][]{
7448   \__problems_solution_args:n { #1 }
7449   \@in@omtexttrue% we are in a statement.
7450   \bool_if:NF \c__problems_boxed_bool { \hrule }
7451   \smallskip\noindent
7452   {\textbf\prob@solution@kw : \enspace}
7453   \begin{small}
7454   \def\current@section@level{\prob@solution@kw}
7455   \ignorespacesandpars
7456 }

```

**\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.

```

7457 \box_new:N \l__problems_solution_box
7458 \newenvironment{solution}{
7459   \stex_html_backend:TF{
7460     \stex_if_do_html:T{
7461       \begin{stex_annotate_env}{solution}{ }
7462     }
7463   }{
7464     \setbox\l__problems_solution_box\vbox\bgroup
7465     \par\smallskip\hrule\smallskip
7466     \noindent\textbf{Solution:}~
7467   }
7468 }{
7469   \stex_html_backend:TF{
7470     \stex_if_do_html:T{
7471       \end{stex_annotate_env}
7472     }
7473   }{

```



```

7474 \smallskip\hrule
7475 \egroup
7476 \bool_if:NT \c__problems_solutions_bool {
7477   \box\l__problems_solution_box
7478 }
7479 }
7480 }
7481
7482 \newcommand\startsolutions{
7483   \bool_set_true:N \c__problems_solutions_bool
7484   % \specialcomment{solution}{\@startsolution}{
7485   %   \bool_if:NF \c__problems_boxed_bool {
7486   %     \hrule\medskip
7487   %   }
7488   %   \end{small}%
7489   % }
7490   % \bool_if:NT \c__problems_boxed_bool {
7491   %   \surroundwithmdframed{solution}
7492   % }
7493 }

```

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

**\stopsolutions**

```

7494 \newcommand\stopsolutions{\bool_set_false:N \c__problems_solutions_bool}%\excludecomment{sol

```

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

so it only remains to start/stop solutions depending on what option was specified.

```

7495 \ifsolutions
7496   \startsolutions
7497 \else
7498   \stopsolutions
7499 \fi

```

**exnote**

```

7500 \bool_if:NTF \c__problems_notes_bool {
7501   \newenvironment{exnote}[1][]{
7502     \par\smallskip\hrule\smallskip
7503     \noindent\textbf{\prob@note@kw : }\small
7504   }{
7505     \smallskip\hrule
7506   }
7507 }{
7508   \excludecomment{exnote}
7509 }

```

**hint**

```

7510 \bool_if:NTF \c__problems_notes_bool {
7511   \newenvironment{hint}[1][]{
7512     \par\smallskip\hrule\smallskip
7513     \noindent\textbf{\prob@hint@kw :~ }\small
7514   }{
7515     \smallskip\hrule
7516   }

```

```

7517 \newenvironment{exhint}[1][]{
7518   \par\smallskip\hrule\smallskip
7519   \noindent\textbf{\prob@hint@kw :~ }\small
7520 }{
7521   \smallskip\hrule
7522 }
7523 }{
7524   \excludecomment{hint}
7525   \excludecomment{exhint}
7526 }

```

gnote

```

7527 \bool_if:NTF \c__problems_notes_bool {
7528   \newenvironment{gnote}[1][]{
7529     \par\smallskip\hrule\smallskip
7530     \noindent\textbf{\prob@gnote@kw : }\small
7531   }{
7532     \smallskip\hrule
7533   }
7534   }{
7535     \excludecomment{gnote}
7536   }

```

### 39.3 Multiple Choice Blocks

EdN:16

mcb 16

```

7537 \newenvironment{mcb}{
7538   \begin{enumerate}
7539 }{
7540   \end{enumerate}
7541 }

```

we define the keys for the mcb macro

```

7542 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
7543   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
7544     \bool_set_true:N #1
7545   }{
7546     \bool_set_false:N #1
7547   }
7548 }
7549 \keys_define:nn { problem / mcb }{
7550   id .str_set:N = \l__problems_mcc_id_str ,
7551   feedback .tl_set:N = \l__problems_mcc_feedback_tl ,
7552   T .default:n = { false } ,
7553   T .bool_set:N = \l__problems_mcc_t_bool ,
7554   F .default:n = { false } ,
7555   F .bool_set:N = \l__problems_mcc_f_bool ,
7556   Ttext .tl_set:N = \l__problems_mcc_Ttext_str ,
7557   Ftext .tl_set:N = \l__problems_mcc_Ftext_str
7558 }
7559 \cs_new_protected:Nn \l__problems_mcc_args:n {

```

---

<sup>16</sup>EdNOTE: MK: maybe import something better here from a dedicated MC package

```

7560 \str_clear:N \l__problems_mcc_id_str
7561 \tl_clear:N \l__problems_mcc_feedback_tl
7562 \bool_set_false:N \l__problems_mcc_t_bool
7563 \bool_set_false:N \l__problems_mcc_f_bool
7564 \tl_clear:N \l__problems_mcc_Ttext_tl
7565 \tl_clear:N \l__problems_mcc_Ftext_tl
7566 \str_clear:N \l__problems_mcc_id_str
7567 \keys_set:nn { problem / mcc }{ #1 }
7568 }

```

`\mcc`

```

7569 \def\mccTrueText{\textbf{(true)~}}
7570 \def\mccFalseText{\textbf{(false)~}}
7571 \newcommand\mcc[2][] {
7572   \l__problems_mcc_args:n{ #1 }
7573   \item[$\Box$] #2
7574   \ifsolutions
7575     \l
7576     \bool_if:NT \l__problems_mcc_t_bool {
7577       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccTrueText\l__problems_mcc_Ttext_tl
7578     }
7579     \bool_if:NT \l__problems_mcc_f_bool {
7580       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccFalseText\l__problems_mcc_Ftext_tl
7581     }
7582     \tl_if_empty:NF \l__problems_mcc_feedback_tl {
7583       \emph{(\l__problems_mcc_feedback_tl)}
7584     }
7585   \fi
7586 } %solutions

```

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

## 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.

```

7587
7588 \keys_define:nn{ problem / inclproblem }{
7589   id      .str_set_x:N = \l__problems_inclprob_id_str,
7590   pts     .tl_set:N    = \l__problems_inclprob_pts_tl,
7591   min     .tl_set:N    = \l__problems_inclprob_min_tl,
7592   title   .tl_set:N    = \l__problems_inclprob_title_tl,
7593   refnum  .int_set:N    = \l__problems_inclprob_refnum_int,
7594   type    .tl_set:N    = \l__problems_inclprob_type_tl,
7595   mhrepos .str_set_x:N = \l__problems_inclprob_mhrepos_str
7596 }
7597 \cs_new_protected:Nn \l__problems_inclprob_args:n {
7598   \str_clear:N \l__problems_prob_id_str
7599   \tl_clear:N \l__problems_inclprob_pts_tl
7600   \tl_clear:N \l__problems_inclprob_min_tl
7601   \tl_clear:N \l__problems_inclprob_title_tl
7602   \tl_clear:N \l__problems_inclprob_type_tl

```

```

7603 \int_zero_new:N \l__problems_inclprob_refnum_int
7604 \str_clear:N \l__problems_inclprob_mhrepos_str
7605 \keys_set:nn { problem / inclproblem }{ #1 }
7606 \tl_if_empty:NT \l__problems_inclprob_pts_tl {
7607   \let\l__problems_inclprob_pts_tl\undefined
7608 }
7609 \tl_if_empty:NT \l__problems_inclprob_min_tl {
7610   \let\l__problems_inclprob_min_tl\undefined
7611 }
7612 \tl_if_empty:NT \l__problems_inclprob_title_tl {
7613   \let\l__problems_inclprob_title_tl\undefined
7614 }
7615 \tl_if_empty:NT \l__problems_inclprob_type_tl {
7616   \let\l__problems_inclprob_type_tl\undefined
7617 }
7618 \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
7619   \let\l__problems_inclprob_refnum_int\undefined
7620 }
7621 }
7622
7623 \cs_new_protected:Nn \__problems_inclprob_clear: {
7624   \let\l__problems_inclprob_id_str\undefined
7625   \let\l__problems_inclprob_pts_tl\undefined
7626   \let\l__problems_inclprob_min_tl\undefined
7627   \let\l__problems_inclprob_title_tl\undefined
7628   \let\l__problems_inclprob_type_tl\undefined
7629   \let\l__problems_inclprob_refnum_int\undefined
7630   \let\l__problems_inclprob_mhrepos_str\undefined
7631 }
7632 \__problems_inclprob_clear:
7633
7634 \newcommand\includeproblem[2][ ]{
7635   \__problems_inclprob_args:n{ #1 }
7636   \exp_args:No \stex_in_repository:nn\l__problems_inclprob_mhrepos_str{
7637     \stex_html_backend:TF {
7638       \str_clear:N \l_tmpa_str
7639       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
7640         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
7641       }
7642       \stex_annotate_invisible:nnn{includeproblem}{
7643         \l_tmpa_str / #2
7644       }{}
7645     }{
7646       \begingroup
7647         \inputreftrue
7648         \tl_if_empty:nTF{ ##1 }{
7649           \input{#2}
7650         }{
7651           \input{ \c_stex_mathhub_str / ##1 / source / #2 }
7652         }
7653       \endgroup
7654     }
7655   }
7656   \__problems_inclprob_clear:

```

```
7657 }
```

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

## 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.

```
7658 \AddToHook{enddocument}{
7659   \bool_if:NT \c__problems_pts_bool {
7660     \message{Total:~\arabic{pts}~points}
7661   }
7662   \bool_if:NT \c__problems_min_bool {
7663     \message{Total:~\arabic{min}~minutes}
7664   }
7665 }
```

The margin pars are reader-visible, so we need to translate

```
7666 \def\pts#1{
7667   \bool_if:NT \c__problems_pts_bool {
7668     \marginpar{#1~\prob@pt@kw}
7669   }
7670 }
7671 \def\min#1{
7672   \bool_if:NT \c__problems_min_bool {
7673     \marginpar{#1~\prob@min@kw}
7674   }
7675 }
```

`\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.

```
7676 \newcounter{pts}
7677 \def\show@pts{
7678   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7679     \bool_if:NT \c__problems_pts_bool {
7680       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
7681       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
7682     }
7683   }{
7684     \tl_if_exist:NT \l__problems_prob_pts_tl {
7685       \bool_if:NT \c__problems_pts_bool {
7686         \tl_if_empty:NTF \l__problems_prob_pts_tl{
7687           \tl_set:Nn \l__problems_prob_pts_tl {0}
7688         }
7689         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
7690         \addtocounter{pts}{\l__problems_prob_pts_tl}
7691       }
7692     }
7693   }
7694 }
```

(End definition for \show@pts. This function is documented on page ??.)  
and now the same for the minutes

\show@min

```

7695 \newcounter{min}
7696 \def\show@min{
7697   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7698     \bool_if:NT \c__problems_min_bool {
7699       \marginpar{\l__problems_inclprob_pts_tl\ min}
7700       \addtocounter{min}{\l__problems_inclprob_min_tl}
7701     }
7702   }{
7703     \tl_if_exist:NT \l__problems_prob_min_tl {
7704       \bool_if:NT \c__problems_min_bool {
7705         \tl_if_empty:NT\l__problems_prob_min_tl{
7706           \tl_set:Nn \l__problems_prob_min_tl {0}
7707         }
7708         \marginpar{\l__problems_prob_min_tl\ min}
7709         \addtocounter{min}{\l__problems_prob_min_tl}
7710       }
7711     }
7712   }
7713 }
7714 \end{package}

```

(End definition for \show@min. This function is documented on page ??.)

## Chapter 40

# Implementation: The hwexam Package

### 40.1 Package Options

The first step is to declare (a few) package options that handle whether certain information is printed or not. Some come with their own conditionals that are set by the options, the rest is just passed on to the `problems` package.

```
7715 \*package>
7716 \ProvidesExplPackage{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
7717 \RequirePackage{13keys2e}
7718
7719 \newif\iftest\testfalse
7720 \DeclareOption{test}{\testtrue}
7721 \newif\ifmultiple\multiplefalse
7722 \DeclareOption{multiple}{\multipletrue}
7723 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
7724 \ProcessOptions
```

Then we make sure that the necessary packages are loaded (in the right versions).

```
7725 \RequirePackage{keyval}[1997/11/10]
7726 \RequirePackage{problem}
```

`\hwexam@*kw` For multilinguality, we define internal macros for keywords that can be specialized in `*.ldf` files.

```
7727 \newcommand\hwexam@assignment@kw{Assignment}
7728 \newcommand\hwexam@given@kw{Given}
7729 \newcommand\hwexam@due@kw{Due}
7730 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
7731 blank~for~extra~space}
7732 \def\hwexam@minutes@kw{minutes}
7733 \newcommand\correction@probs@kw{prob.}
7734 \newcommand\correction@pts@kw{total}
7735 \newcommand\correction@reached@kw{reached}
7736 \newcommand\correction@sum@kw{Sum}
7737 \newcommand\correction@grade@kw{grade}
7738 \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

```

7739 \AddToHook{begindocument}{
7740 \ltx@ifpackageloaded{babel}{
7741 \makeatletter
7742 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7743 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7744 \input{hwexam-ngerman.ldf}
7745 }
7746 \clist_if_in:NnT \l_tmpa_clist {finnish}{
7747 \input{hwexam-finnish.ldf}
7748 }
7749 \clist_if_in:NnT \l_tmpa_clist {french}{
7750 \input{hwexam-french.ldf}
7751 }
7752 \clist_if_in:NnT \l_tmpa_clist {russian}{
7753 \input{hwexam-russian.ldf}
7754 }
7755 \makeatother
7756 }{}
7757 }
7758

```

## 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.

```

7759 \newcounter{assignment}
7760 %\numberproblemsin{assignment}

We will prepare the keyval support for the assignment environment.

7761 \keys_define:nn { hwexam / assignment } {
7762 id .str_set:N = \l_@@_assign_id_str,
7763 number .int_set:N = \l_@@_assign_number_int,
7764 title .tl_set:N = \l_@@_assign_title_tl,
7765 type .tl_set:N = \l_@@_assign_type_tl,
7766 given .tl_set:N = \l_@@_assign_given_tl,
7767 due .tl_set:N = \l_@@_assign_due_tl,
7768 loadmodules .code:n = {
7769 \bool_set_true:N \l_@@_assign_loadmodules_bool
7770 }
7771 }
7772 \cs_new_protected:Nn \_@@_assignment_args:n {
7773 \str_clear:N \l_@@_assign_id_str
7774 \int_set:Nn \l_@@_assign_number_int {-1}
7775 \tl_clear:N \l_@@_assign_title_tl
7776 \tl_clear:N \l_@@_assign_type_tl
7777 \tl_clear:N \l_@@_assign_given_tl
7778 \tl_clear:N \l_@@_assign_due_tl
7779 \bool_set_false:N \l_@@_assign_loadmodules_bool
7780 \keys_set:nn { hwexam / assignment }{ #1 }
7781 }

```



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.

```

7782 \newcommand\given@due[2]{
7783 \bool_lazy_all:nF {
7784 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
7785 { \tl_if_empty_p:V \l_@@_assign_given_tl }
7786 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
7787 { \tl_if_empty_p:V \l_@@_assign_due_tl }
7788 }{ #1 }
7789
7790 \tl_if_empty:NTF \l_@@_inclasssign_given_tl {
7791 \tl_if_empty:NF \l_@@_assign_given_tl {
7792 \hwexam@given@kw\xspace\l_@@_assign_given_tl
7793 }
7794 }{
7795 \hwexam@given@kw\xspace\l_@@_inclasssign_given_tl
7796 }
7797
7798 \bool_lazy_or:nnF {
7799 \bool_lazy_and_p:nn {
7800 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
7801 }{
7802 \tl_if_empty_p:V \l_@@_assign_due_tl
7803 }
7804 }{
7805 \bool_lazy_and_p:nn {
7806 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
7807 }{
7808 \tl_if_empty_p:V \l_@@_assign_due_tl
7809 }
7810 }{ ,~ }
7811
7812 \tl_if_empty:NTF \l_@@_inclasssign_due_tl {
7813 \tl_if_empty:NF \l_@@_assign_due_tl {
7814 \hwexam@due@kw\xspace \l_@@_assign_due_tl
7815 }
7816 }{
7817 \hwexam@due@kw\xspace \l_@@_inclasssign_due_tl
7818 }
7819
7820 \bool_lazy_all:nF {
7821 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
7822 { \tl_if_empty_p:V \l_@@_assign_given_tl }
7823 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
7824 { \tl_if_empty_p:V \l_@@_assign_due_tl }
7825 }{ #2 }
7826 }

```

`\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.

```

7827 \newcommand\assignment@title[3]{
7828 \tl_if_empty:NTF \l_@@_inclasssign_title_tl {
7829 \tl_if_empty:NTF \l_@@_assign_title_tl {
7830 #1
7831 }{
7832 #2\l_@@_assign_title_tl#3
7833 }
7834 }{
7835 #2\l_@@_inclasssign_title_tl#3
7836 }
7837 }

```

(End definition for \assignment@title. This function is documented on page ??.)

**\assignment@number** Like \assignment@title only for the number, and no around part.

```

7838 \newcommand\assignment@number{
7839 \int_compare:nNnTF \l_@@_inclasssign_number_int = {-1} {
7840 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
7841 \arabic{assignment}
7842 } {
7843 \int_use:N \l_@@_assign_number_int
7844 }
7845 }{
7846 \int_use:N \l_@@_inclasssign_number_int
7847 }
7848 }

```

(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.

```

7849 \newenvironment{assignment}[1][]{
7850 \_@@_assignment_args:n { #1 }
7851 %\sref@target
7852 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
7853 \global\stepcounter{assignment}
7854 }{
7855 \global\setcounter{assignment}{\int_use:N\l_@@_assign_number_int}
7856 }
7857 \setcounter{problem}{0}
7858 \renewcommand\prob@label[1]{\assignment@number.##1}
7859 \def\current@section@level{\document@hwexamtype}
7860 %\sref@label{id}{\document@hwexamtype \thesection}
7861 \begin{@assignment}
7862 }{
7863 \end{@assignment}
7864 }

```

In the multi-assignment case we just use the omdoc environment for suitable sectioning.

```

7865 \def\ass@title{
7866 {\protect\document@hwexamtype}\arabic{assignment}
7867 \assignment@title{}\;\;{}{}\; -- \given@due{}\;}
7868 }
7869 \ifmultiple
7870 \newenvironment{@assignment}{
7871 \bool_if:NTF \l_@@_assign_loadmodules_bool {
7872 \begin{sfragment}[loadmodules]{\ass@title}
7873 }{
7874 \begin{sfragment}{\ass@title}
7875 }
7876 }{
7877 \end{sfragment}
7878 }

```

for the single-page case we make a title block from the same components.

```

7879 \else
7880 \newenvironment{@assignment}{
7881 \begin{center}\bf
7882 \Large@title\strut\
7883 \document@hwexamtype}\arabic{assignment}\assignment@title{}\;\;{}{}\;
7884 \large\given@due{--\;\;{}{}\;}\;
7885 \end{center}
7886 }{}
7887 \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.

```

7888 \keys_define:nn { hwexam / inclassignment } {
7889 %id .str_set_x:N = \l_@@_assign_id_str,
7890 number .int_set:N = \l_@@_inclassign_number_int,
7891 title .tl_set:N = \l_@@_inclassign_title_tl,
7892 type .tl_set:N = \l_@@_inclassign_type_tl,
7893 given .tl_set:N = \l_@@_inclassign_given_tl,
7894 due .tl_set:N = \l_@@_inclassign_due_tl,
7895 mhrepos .str_set_x:N = \l_@@_inclassign_mhrepos_str
7896 }
7897 \cs_new_protected:Nn \l_@@_inclassignment_args:n {
7898 \int_set:Nn \l_@@_inclassign_number_int {-1}
7899 \tl_clear:N \l_@@_inclassign_title_tl
7900 \tl_clear:N \l_@@_inclassign_type_tl
7901 \tl_clear:N \l_@@_inclassign_given_tl
7902 \tl_clear:N \l_@@_inclassign_due_tl
7903 \str_clear:N \l_@@_inclassign_mhrepos_str
7904 \keys_set:nn { hwexam / inclassignment }{ #1 }
7905 }
7906 \l_@@_inclassignment_args:n {}
7907
7908 \newcommand\inputassignment[2][]{

```

```

7909 \_@@_inclassassignment_args:n { #1 }
7910 \str_if_empty:NTF \l_@@_inclasssign_mhrepos_str {
7911 \input{#2}
7912 }{
7913 \stex_in_repository:nn{\l_@@_inclasssign_mhrepos_str}{
7914 \input{\mhpath{\l_@@_inclasssign_mhrepos_str}{#2}}
7915 }
7916 }
7917 \_@@_inclassassignment_args:n {}
7918 }
7919 \newcommand\includeassignment[2][]{
7920 \newpage
7921 \inputassignment[#1]{#2}
7922 }

```

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

## 40.4 Typesetting Exams

\quizheading

```

7923 \ExplSyntaxOff
7924 \newcommand\quizheading[1]{%
7925 \def\@tas{#1}%
7926 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
7927 \ifx\@tas\@empty\else%
7928 \noindent TA:~\@for\@I:=\@tas\do{{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
7929 \fi%
7930 }
7931 \ExplSyntaxOn

```

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

\testheading

```

7932
7933 \def\hwexamheader{\input{hwexam-default.header}}
7934
7935 \def\hwexamminutes{
7936 \tl_if_empty:NTF \testheading@duration {
7937 {\testheading@min}~\hwexam@minutes@kw
7938 }{
7939 \testheading@duration
7940 }
7941 }
7942
7943 \keys_define:nn { hwexam / testheading } {
7944 min .tl_set:N = \testheading@min,
7945 duration .tl_set:N = \testheading@duration,
7946 reqpts .tl_set:N = \testheading@reqpts,
7947 tools .tl_set:N = \testheading@tools
7948 }
7949 \cs_new_protected:Nn \_@@_testheading_args:n {
7950 \tl_clear:N \testheading@min
7951 \tl_clear:N \testheading@duration

```

```

7952 \tl_clear:N \testheading@reqpts
7953 \tl_clear:N \testheading@tools
7954 \keys_set:nn { hwexam / testheading }{ #1 }
7955 }
7956 \newenvironment{testheading}[1][ ]{
7957 \_@@_testheading_args:n{ #1 }
7958 \newcount\check@time\check@time=\testheading@min
7959 \advance\check@time by -\theassignment@totalmin
7960 \newif\if@bonuspoints
7961 \tl_if_empty:NTF \testheading@reqpts {
7962 \@bonuspointsfalse
7963 }{
7964 \newcount\bonus@pts
7965 \bonus@pts=\theassignment@totalpts
7966 \advance\bonus@pts by -\testheading@reqpts
7967 \edef\bonus@pts{\the\bonus@pts}
7968 \@bonuspointstrue
7969 }
7970 \edef\check@time{\the\check@time}
7971
7972 \makeatletter\hwexamheader\makeatother
7973 }{
7974 \newpage
7975 }

```

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

\testspace

```

7976 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

7977 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

7978 \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.

```

7979 <@@=problems>
7980 \renewcommand\@problem[3]{
7981 \stepcounter{assignment@probs}
7982 \def\__problemspts{#2}
7983 \ifx\__problemspts\@empty\else
7984 \addtocounter{assignment@totalpts}{#2}
7985 \fi
7986 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
7987 \xdef\correction@probs{\correction@probs & #1}%
7988 \xdef\correction@pts{\correction@pts & #2}
7989 \xdef\correction@reached{\correction@reached &}

```

```

7990 }
7991 <@@=hwexam>

```

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

`\correction@table` This macro generates the correction table

```

7992 \newcounter{assignment@probs}
7993 \newcounter{assignment@totalpts}
7994 \newcounter{assignment@totalmin}
7995 \def\correction@probs{\correction@probs@kw}
7996 \def\correction@pts{\correction@pts@kw}
7997 \def\correction@reached{\correction@reached@kw}
7998 \stepcounter{assignment@probs}
7999 \newcommand\correction@table{
8000 \resizebox{\textwidth}{!}{%
8001 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
8002 &\multicolumn{\theassignment@probs}{c|}||%|
8003 {\footnotesize\correction@forgrading@kw} &\\ \hline
8004 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
8005 \correction@pts & \theassignment@totalpts & \\ \hline
8006 \correction@reached & & \[.7cm]\hline
8007 \end{tabular}}
8008 </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}

```