

# The sTeX3 Package \*

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

2022-03-27

## 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 2022-03-27)

# 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 . . . . .	100
24.6	Auxiliary Methods . . . . .	101
<b>25</b>	<b>gTeX-MathHub Implementation</b>	<b>104</b>
25.1	Generic Path Handling . . . . .	104
25.2	PWD and kpsewhich . . . . .	106
25.3	File Hooks and Tracking . . . . .	107
25.4	MathHub Repositories . . . . .	108
25.5	Using Content in Archives . . . . .	112
<b>26</b>	<b>gTeX-References Implementation</b>	<b>117</b>
26.1	Document URIs and URLs . . . . .	117
26.2	Setting Reference Targets . . . . .	119
26.3	Using References . . . . .	121
<b>27</b>	<b>gTeX-Modules Implementation</b>	<b>124</b>
27.1	The smodule environment . . . . .	128
27.2	Invoking modules . . . . .	133
<b>28</b>	<b>gTeX-Module Inheritance Implementation</b>	<b>135</b>
28.1	SMS Mode . . . . .	135
28.2	Inheritance . . . . .	139

<b>29</b>	<b>STEX-Symbols Implementation</b>	<b>144</b>
29.1	Symbol Declarations . . . . .	144
29.2	Notations . . . . .	151
29.3	Variables . . . . .	161
<b>30</b>	<b>STEX-Terms Implementation</b>	<b>168</b>
30.1	Symbol Invocations . . . . .	168
30.2	Terms . . . . .	175
30.3	Notation Components . . . . .	179
30.4	Variables . . . . .	182
30.5	Sequences . . . . .	183
<b>31</b>	<b>STEX-Structural Features Implementation</b>	<b>185</b>
31.1	Imports with modification . . . . .	186
31.2	The feature environment . . . . .	193
31.3	Structure . . . . .	194
<b>32</b>	<b>STEX-Statements Implementation</b>	<b>205</b>
32.1	Definitions . . . . .	205
32.2	Assertions . . . . .	210
32.3	Examples . . . . .	214
32.4	Logical Paragraphs . . . . .	216
<b>33</b>	<b>The Implementation</b>	<b>222</b>
33.1	Package Options . . . . .	222
33.2	Proofs . . . . .	222
33.3	Justifications . . . . .	233
<b>34</b>	<b>STEX-Others Implementation</b>	<b>235</b>
<b>35</b>	<b>STEX-Metatheory Implementation</b>	<b>236</b>
<b>36</b>	<b>Tikzinput Implementation</b>	<b>239</b>
<b>37</b>	<b>document-structure.sty Implementation</b>	<b>241</b>
37.1	The document-structure Class . . . . .	241
37.2	Class Options . . . . .	241
37.3	Beefing up the document environment . . . . .	242
37.4	Implementation: document-structure Package . . . . .	242
37.5	Package Options . . . . .	242
37.6	Document Structure . . . . .	244
37.7	Front and Backmatter . . . . .	247
37.8	Global Variables . . . . .	249

<b>38 NotesSlides – Implementation</b>	<b>250</b>
38.1 Class and Package Options . . . . .	250
38.2 Notes and Slides . . . . .	252
38.3 Header and Footer Lines . . . . .	256
38.4 Frame Images . . . . .	257
38.5 Colors and Highlighting . . . . .	258
38.6 Sectioning . . . . .	259
38.7 Excursions . . . . .	262
<b>39 The Implementation</b>	<b>263</b>
39.1 Package Options . . . . .	263
39.2 Problems and Solutions . . . . .	264
39.3 Multiple Choice Blocks . . . . .	270
39.4 Including Problems . . . . .	271
39.5 Reporting Metadata . . . . .	272
<b>40 Implementation: The hwexam Class</b>	<b>274</b>
40.1 Class Options . . . . .	274
<b>41 Implementation: The hwexam Package</b>	<b>276</b>
41.1 Package Options . . . . .	276
41.2 Assignments . . . . .	277
41.3 Including Assignments . . . . .	280
41.4 Typesetting Exams . . . . .	281
41.5 Leftovers . . . . .	283

# 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,
- RuSTeX 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>
</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.

### 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/smglom/calculus`.

Each such archive needs two subdirectories:

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

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

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

We recommend 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{paragraph}` [`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 `\inputref`ing 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/smglob/calculus
narration-base: http://mathhub.info/smglob/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:

---

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

---

$\backslash\text{inputref}$	$\backslash\text{inputref}$ [Some/Archive]{some/file} behaves like $\backslash\text{mhinput}$ , but wraps the input in a $\backslash\text{begingroup} \dots \backslash\text{endgroup}$ . When converting to $\text{xhtml}$ , the file is not input at all, and instead an $\text{html}$ -annotation is inserted that references the file. In the majority of cases $\backslash\text{inputref}$ is likely to be preferred over $\backslash\text{mhinput}$ .
-----------------------------	---

---

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

---

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

---

$\backslash\text{libinput}$	$\backslash\text{libinput}$ {some/file} searches for a file some/file in <ul style="list-style-type: none"><li>• the <code>lib</code>-directory of the current archive, and</li><li>• the <code>lib</code>-directory of a <code>meta-inf</code>-archive in (any of) the archive groups containing the current archive</li></ul> and include all found files in reverse order; e.g. $\backslash\text{libinput}\{\text{preamble}\}$ in a <code>.tex</code> -file in <code>smglom/calculus</code> will <i>first</i> input <code>../smglom/meta-inf/lib/preamble.tex</code> and then <code>../smglom/calculus/lib/preamble.tex</code> . Will throw an error if <i>no</i> candidate for some/file is found.
-----------------------------	---

---

$\backslash\text{libusepackage}$	$\backslash\text{libusepackage}$ [package-options]{some/file} searches for a file some/file.sty in the same way that $\backslash\text{libinput}$ does, but will call $\backslash\text{usepackage}$ [package-options]{path/to/some/file} instead of $\backslash\text{input}$ . Will throw an error if not <i>exactly one</i> 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 `<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.

$\hookrightarrow$  **M**  $\rightarrow$  **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  $\rightsquigarrow$  **T**  $\rightsquigarrow$  OMBIND-terms in OMDoc/MMT, rather than OMA.

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

### Example 9

Input:

```
1 \symdef{summation}[args=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 `\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. `\addition{a,b,c,d,e}` rather than having to write something like `\addition{a}{\addition{b}{\addition{c}{\addition{d}{e}}}}`!

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

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

### Example 10

Input:

```

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

```

Output:

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

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

### Example 11

Input:

```

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

```

Output:

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

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

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

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

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

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

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

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

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

## 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 `lcm(a,b)` in *english*, but is called *kleinstes gemeinsames Vielfaches* in *german* and consequently denoted as `kgV(a,b)` there.

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

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

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

### 3.4.2 Simple Inheritance and Namespaces

---

`\importmodule`  
`\usemodule`

---

`\importmodule`[Some/Archive]{path?ModuleName} is only allowed within an `smodule`-environment and makes the symbols declared 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 $\varM!:=\mathstrut{\varM{universe},\varM{op}!,\varM{unit}}{\varM{universe}}$
5 such that
6 $\varM{op}!:\mathstrut{\varM{universe},\varM{universe}}{\varM{universe}}$
7 and...
8
9 \varinstantiate{varMb}{universe = Int}{monoid}{M_2}
10
11 \noindent Let $\varMb!:=\mathstrut{\varMb{universe},\varMb{op}!,\varMb{unit}}{\varMb{universe}}$
12 a \symname{monoid} on $\mathbb{Z}$...

```

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

```

Output:

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

TODO: explain donotclone

### 3.4.5 The interpretmodule Environment

TODO: explain

### Example 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 $\text{\TeX}$ 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  
 $\rightarrow$  M  $\rightarrow$  OMA with `<OMS name="...?addition"/>` as head and `<OMV name="n"/>` and  
 $\rightarrow$  T  $\rightarrow$  `<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 $\text{\addition*}{
3   \arg*{\text{\addition{\svar{n}}{\svar{m}}}}
4   \comp{+}
5   \arg{\svar{k}}
6 }$ yields...
```

Output:

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

## 4.3 Referencing Symbols and Statements

TODO: references documentation

## 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).

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

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,  $\text{\texttt{S}\text{\textsf{T}}\text{\textsf{E}}\text{\textsf{X}}$  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> XML
--------------------------	---

---

---

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

---

---

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

---

---

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

---

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

---

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

---

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

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

### 8.1.2 Babel Languages

---

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

---

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

### 8.1.3 Auxiliary Methods

---

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

---

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

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

---

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

---

## Chapter 9

# STEX-MathHub

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

### 9.1 Macros and Environments

---

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

---

#### 9.1.1 Files, Paths, URIs

---

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

---

---

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

---

---

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

---

---

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

---

---

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

---

---

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

---



---

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

---

### 9.1.2 MathHub Archives

---

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

---

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

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

---

<code>\l_stex_current_repository_prop</code>
--

---

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

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

---

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

---

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

---

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

---



---

<code>\stex_in_repository:nn</code>	<code>\stex_in_repository:nn{&lt;repository-name&gt;}{&lt;code&gt;}</code>
-------------------------------------	--

---

Change the current repository to `{<repository-name>}` (or not, if `{<repository-name>}` is empty), and passes its ID on to `{<code>}` as `#1`. Switches back to the previous repository after executing `{<code>}`.



### 9.1.3 Using Content in Archives

<hr/> <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> additionally 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_modules_ns_str`. Additionally, the sub path relative to the current repository is stored in `\l_stex_modules_subpath_str`.

### 11.1.1 The `smodule` environment

`module` `\begin{module}[\langle options \rangle]{\langle name \rangle}`  
 Opens a new module with name `\langle name \rangle`. Options are:

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

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

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

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

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

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

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

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

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

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

---

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

---

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

---

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

---

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

---

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

---

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

---

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

---

---

**`\stex_activate_module:n`**

---

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

## Chapter 12

# STEX-Module Inheritance

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

### 12.1 Macros and Environments

#### 12.1.1 SMS Mode

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

---

`\g_stex_smsmode_allowedmacros_tl`

---

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

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

---

`\g_stex_smsmode_allowedmacros_escape_tl`

---

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

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

---

`\g_stex_smsmode_allowedenvs_seq`

---

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

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

---

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

---

Tests whether SMS mode is currently active.



---

<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

---

**`\stex_import_module_uri:nn`**

---

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

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

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

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

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

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

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

2. Otherwise:

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

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

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

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

---

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

---

stores the result in these four variables.

---

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

---

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

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

# Chapter 13

## STEX-Symbols

Code related to symbol declarations and notations

### 13.1 Macros and Environments

---

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

---

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

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

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

# Chapter 14

## STEX-Terms

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

### 14.1 Macros and Environments

<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>\stex_highlight_term:nn</code>	<code>\stex_highlight_term:nn{⟨URI⟩}{⟨args⟩}</code>
	Establishes a context for <code>\comp</code> . Stores the URI in a variable so that <code>\comp</code> knows which symbol governs the current notation.
<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  $\S\TeX$  collection, a version of  $\TeX/\text{L}\text{A}\text{T}\text{E}\text{X}$  that allows to markup  $\TeX/\text{L}\text{A}\text{T}\text{E}\text{X}$  documents semantically without leaving the document format, essentially turning  $\TeX/\text{L}\text{A}\text{T}\text{E}\text{X}$  into a document format for mathematical knowledge management (MKM).

This package supplies an infrastructure for writing OMDOC documents in  $\text{L}\text{A}\text{T}\text{E}\text{X}$ . This includes a simple structure sharing mechanism for  $\S\TeX$  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  $\S\TeX$  sources, or after translation.

### 20.1 Introduction

$\S\TeX$  is a version of  $\TeX/\text{L}\text{A}\text{T}\text{E}\text{X}$  that allows to markup  $\TeX/\text{L}\text{A}\text{T}\text{E}\text{X}$  documents semantically without leaving the document format, essentially turning  $\TeX/\text{L}\text{A}\text{T}\text{E}\text{X}$  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  $\S\TeX$  sources, or after translation. Currently, trans-document referencing provided by this package can only be used in the  $\S\TeX$  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

`document` The top-level `document` environment can be given key/value information by the `\documentkeys` macro in the preamble<sup>3</sup>. This can be used to give metadata about the document. For the moment only the `id` key is used to give an identifier to the `omdoc` element resulting from the L<sup>A</sup>T<sub>E</sub>XML transformation.

`sfragment` The structure of the document is given by the `omgroup` environment just like in OMDoc. In the L<sup>A</sup>T<sub>E</sub>X route, the `omgroup` environment is flexibly mapped to sectioning commands, inducing the proper sectioning level from the nesting of `omgroup` environments. Correspondingly, the `omgroup` environment takes an optional key/value argument for metadata followed by a regular argument for the (section) title of the `omgroup`. The optional metadata argument has the keys `id` for an identifier, `creators` and `contributors` for the Dublin Core metadata [DCM03]; see [Koh20a] for details of the format. The `short` allows to give a short title for the generated section. If the title contains semantic macros, they need to be protected by `\protect`, and we need to give the `loadmodules` key it needs no value. For instance we would have

```
\begin{smodule}{foo}
\symdef{bar}{B^a_r}
...
\begin{sfragment}[id=sec.barderviv,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`  
`\STRsemantics` 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>

### 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  $\text{\LaTeX}$  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> | • The options <code>slides</code> and <code>notes</code> switch between slides mode and notes mode (see |
| <code>notes</code>  | Section 21.2.2).  |



<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  $\TeX$  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  $\TeX$ 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=elephants,pts=10,min=2,title=Fitting Elephants]
    How many Elephants 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=elephants,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 0.1 (Fitting Elephants)
How many Elephants can you fit into a Volkswagen beetle?


---


Hint: Think positively, this is simple!


---


Note:Justify your answer


---


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


---


```

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]
  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 0.2 (Functions)**

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

1. def
2. function
3. fun
4. public static void

---

**Problem 0.3 (Functions)**

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

1. def  
!
2. function  
that is for C and C++
3. fun  
that is for Standard ML
4. public static void  
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.



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,rustex}
6 \ProvidesExplClass{stex}{2022/03/03}{3.1.0}{sTeX document class}
7 \rustex_if:TF {
8   \LoadClass{article}
9 }{
10   \LoadClass[border=1px,varwidth]{standalone}
11   \setlength\textwidth{15cm}
12 }
13
14 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
15 \ProcessOptions
16
17 \RequirePackage{stex}
18 </cls>
```

### 24.2 Preliminaries

```
19 <*package>
20
21 %%%%%%%%% basics.dtx %%%%%%%%%
22
23 \RequirePackage{expl3,l3keys2e,ltxcmds}
24 \ProvidesExplPackage{stex}{2022/03/03}{3.1.0}{sTeX package}
25
26 %\RequirePackage{morewrites}
```

```

27 %\RequirePackage{amsmath}
28
    Package options:
29 \keys_define:nn { stex } {
30   debug      .clist_set:N = \c_stex_debug_clist ,
31   lang       .clist_set:N = \c_stex_languages_clist ,
32   mathhub    .tl_set_x:N = \mathhub ,
33   sms        .bool_set:N = \c_stex_persist_mode_bool ,
34   image      .bool_set:N = \c_tikzinput_image_bool ,
35   unknown    .code:n      = {}
36 }
37 \ProcessKeysOptions { stex }

\stex The  $\TeX$  logo:
\TeX
38 \protected\def\stex{
39   \texorpdfstring{\raisebox{-.5ex}{S}\kern-.5ex\TeX}{sTeX}\xspace%
40 }
41 \let\TeX\stex

```

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

## 24.3 Messages and logging

```

42 <@@=stex_log>

    Warnings and error messages
43 \msg_new:nnn{stex}{error/unknownlanguage}{
44   Unknown~language:~#1
45 }
46 \msg_new:nnn{stex}{warning/nomathhub}{
47   MATHHUB~system~variable~not~found~and~no~
48   \detokenize{\mathhub}~value~set!
49 }
50 \msg_new:nnn{stex}{error/deactivated-macro}{
51   The~\detokenize{#1}~command~is~only~allowed~in~#2!
52 }

\stex_debug:nn A simple macro issuing package messages with subpath.
53 \cs_new_protected:Nn \stex_debug:nn {
54   \clist_if_in:NnTF \c_stex_debug_clist { all } {
55     \msg_set:nnn{stex}{debug / #1}{
56       \\Debug~#1:~#2\\
57     }
58     \msg_none:nn{stex}{debug / #1}
59   }{
60     \clist_if_in:NnT \c_stex_debug_clist { #1 } {
61       \msg_set:nnn{stex}{debug / #1}{
62         \\Debug~#1:~#2\\
63       }
64       \msg_none:nn{stex}{debug / #1}
65     }
66   }
67 }

```

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

Redirecting messages:

```

68 \clist_if_in:NnTF \c_stex_debug_clist {all} {
69   \msg_redirect_module:nnn{ stex }{ none }{ term }
70 }{
71   \clist_map_inline:Nn \c_stex_debug_clist {
72     \msg_redirect_name:nnn{ stex }{ debug / ##1 }{ term }
73   }
74 }
75
76 \stex_debug:nn{log}{debug~mode~on}

```

## 24.4 HTML Annotations

```

77 <@=stex_annotate>
78 \RequirePackage{rustex}

```

We add the namespace abbreviation `ns:stex="http://kwarc.info/ns/sTeX"` to `RUSTEX`:

```

79 \rustex_add_Namespace:nn{stex}{http://kwarc.info/ns/sTeX}
80 \rustex_add_Namespace:nn{mmt}{http://uniformal.github.io/MMT}

```

Conditionals for `LATXML`:

`\if@latexml`

```

81 \ifcsname if@latexml\endcsname\else
82   \expandafter\newif\csname if@latexml\endcsname\@latexmlfalse
83 \fi

```

(End definition for `\if@latexml`. This function is documented on page 46.)

`\latexml_if_p:`

`\latexml_if:TF`

```

84 \prg_new_conditional:Nnn \latexml_if: {p, T, F, TF} {
85   \if@latexml
86     \expandafter\prg_return_true:
87   \else:
88     \expandafter\prg_return_false:
89   \fi:
90 }

```

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

`\l__stex_annotate_arg_tl`  
`\c__stex_annotate_emptyarg_tl`

Used by annotation macros to ensure that the HTML output to annotate is not empty.

```

91 \tl_new:N \l__stex_annotate_arg_tl
92 \tl_const:Nx \c__stex_annotate_emptyarg_tl {
93   \rustex_if:TF {
94     \rustex_direct_HTML:n { \c_ampersand_str \c_hash_str 8205; }
95   }{-}
96 }

```

(End definition for `\l__stex_annotate_arg_tl` and `\c__stex_annotate_emptyarg_tl`.)



`\_stex_annotate_checkempty:n`

```

97 \cs_new_protected:Nn \_stex_annotate_checkempty:n {
98   \tl_set:Nn \l__stex_annotate_arg_tl { #1 }
99   \tl_if_empty:NT \l__stex_annotate_arg_tl {
100     \tl_set_eq:NN \l__stex_annotate_arg_tl \c__stex_annotate_emptyarg_tl
101   }
102 }

```

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

`\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:nnx`

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>).

`\stex_annotate_invisible:n`

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.

`\stex_annotate_invisible:nnn`

```

120 \rustex_if:TF{
121   \cs_new_protected:Nn \stex_annotate:nnn {
122     \_stex_annotate_checkempty:n { #3 }
123     \rustex_annotate_HTML:nn {
124       property="stex:#1" ~
125       resource="#2"
126     } {
127       \mode_if_vertical:TF{
128         \tl_use:N \l__stex_annotate_arg_tl\par
129       }{
130         \tl_use:N \l__stex_annotate_arg_tl
131       }
132     }
133   }
134   \cs_new_protected:Nn \stex_annotate_invisible:n {

```

```

135 \__stex_annotate_checkempty:n { #1 }
136 \rustex_annotate_HTML:nn {
137   stex:visible="false" ~
138   style:display="none"
139 } {
140   \mode_if_vertical:TF{
141     \tl_use:N \l__stex_annotate_arg_tl\par
142   }{
143     \tl_use:N \l__stex_annotate_arg_tl
144   }
145 }
146 }
147 \cs_new_protected:Nn \stex_annotate_invisible:nnn {
148   \__stex_annotate_checkempty:n { #3 }
149   \rustex_annotate_HTML:nn {
150     property="stex:#1" ~
151     resource="#2" ~
152     stex:visible="false" ~
153     style:display="none"
154   } {
155     \mode_if_vertical:TF{
156       \tl_use:N \l__stex_annotate_arg_tl\par
157     }{
158       \tl_use:N \l__stex_annotate_arg_tl
159     }
160   }
161 }
162 \NewDocumentEnvironment{stex_annotate_env} { m m } {
163   \par
164   \rustex_annotate_HTML_begin:n {
165     property="stex:#1" ~
166     resource="#2"
167   }
168 }{
169   \par\rustex_annotate_HTML_end:
170 }
171 }{
172   \latexml_if:TF {
173     \cs_new_protected:Nn \stex_annotate:nnn {
174       \__stex_annotate_checkempty:n { #3 }
175       \mode_if_math:TF {
176         \cs:w latexml@annotate@math\cs_end:{#1}{#2}{
177           \tl_use:N \l__stex_annotate_arg_tl
178         }
179       }{
180         \cs:w latexml@annotate@text\cs_end:{#1}{#2}{
181           \tl_use:N \l__stex_annotate_arg_tl
182         }
183       }
184     }
185     \cs_new_protected:Nn \stex_annotate_invisible:n {
186       \__stex_annotate_checkempty:n { #1 }
187       \mode_if_math:TF {
188         \cs:w latexml@invisible@math\cs_end:{

```

```

189         \tl_use:N \l__stex_annotate_arg_tl
190     }
191 } {
192     \cs:w latexml@invisible@text\cs_end:{
193         \tl_use:N \l__stex_annotate_arg_tl
194     }
195 }
196 }
197 \cs_new_protected:Nn \stex_annotate_invisible:nnn {
198     \__stex_annotate_checkempty:n { #3 }
199     \cs:w latexml@annotate@invisible\cs_end:{#1}{#2}{
200         \tl_use:N \l__stex_annotate_arg_tl
201     }
202 }
203 \NewDocumentEnvironment{stex_annotate_env} { m m } {
204     \par\begin{latexml@annotateenv}{#1}{#2}
205 }{
206     \par\end{latexml@annotateenv}
207 }
208 }{
209     \cs_new_protected:Nn \stex_annotate:nnn {#3}
210     \cs_new_protected:Nn \stex_annotate_invisible:n {}
211     \cs_new_protected:Nn \stex_annotate_invisible:nnn {}
212     \NewDocumentEnvironment{stex_annotate_env} { m m } {}{}
213 }
214 }

```

(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

```

215 <@=stex_language>

```

`\c_stex_languages_prop` We store language abbreviations in two (mutually inverse) property lists:  
`\c_stex_language_abbrevs_prop`

```

216 \prop_const_from_keyval:Nn \c_stex_languages_prop {
217     en = english ,
218     de = ngerman ,
219     ar = arabic ,
220     bg = bulgarian ,
221     ru = russian ,
222     fi = finnish ,
223     ro = romanian ,
224     tr = turkish ,
225     fr = french
226 }
227
228 \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop {
229     english = en ,
230     ngerman = de ,
231     arabic = ar ,
232     bulgarian = bg ,
233     russian = ru ,
234     finnish = fi ,

```

```

235   romanian = ro ,
236   turkish  = tr ,
237   french   = fr
238 }
239 % todo: chinese simplified (zhs)
240 %       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:

```

241 \clist_if_empty:NF \c_stex_languages_clist {
242   \clist_clear:N \l_tmpa_clist
243   \clist_map_inline:Nn \c_stex_languages_clist {
244     \prop_get:NnNTF \c_stex_languages_prop { #1 } \l_tmpa_str {
245       \clist_put_right:No \l_tmpa_clist \l_tmpa_str
246     } {
247       \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
248     }
249   }
250   \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
251   \RequirePackage[\clist_use:Nn \l_tmpa_clist,]{babel}
252 }
253 \AtBeginDocument{
254   \bool_lazy_any:nT {
255     {\rustex_if_p:}
256     {\latexml_if_p:}
257   } {
258     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
259     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
260     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
261     \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
262     \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
263       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
264       \stex_debug:nn{basics} {Language~\l_tmpa_str~
265         inferred~from~file~name}
266       \stex_annotate_invisible:nnn{language}{ \l_tmpa_str }{}
267     }
268   }
269 }
270 }

```

## 24.6 Auxiliary Methods

`\stex_deactivate_macro:Nn`

```

271 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
272   \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
273   \def#1{
274     \msg_error:nnnn{stex}{error/deactivated-macro}{\detokenize{#1}}{#2}
275   }
276 }

```

(End definition for `\stex_deactivate_macro:Nn`. This function is documented on page 47.)

`\stex_reactivate_macro:N`

```

277 \cs_new_protected:Nn \stex_reactivate_macro:N {
278   \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname
279 }

```

(End definition for `\stex_reactivate_macro:N`. This function is documented on page 47.)

`\ignorespacesandpars`

```

280 \protected\def\ignorespacesandpars{
281   \begingroup\catcode13=10\relax
282   \@ifnextchar\par{
283     \endgroup\expandafter\ignorespacesandpars\@gobble
284   }{
285     \endgroup
286   }
287 }
288
289 \cs_new:Nn \stex_copy_control_sequence:NNN {
290   \tl_set:Nx \_tmp_args_tl {\cs_argument_spec:N #2}
291   \tl_remove_all:Nn \_tmp_args_tl {\c_hash_str}
292   \int_set:Nn \l_tmpa_int {\tl_count:N \_tmp_args_tl}
293
294   \tl_clear:N \_tmp_args_tl
295   \int_step_inline:nn \l_tmpa_int {
296     \tl_put_right:Nx \_tmp_args_tl {{\exp_not:n{####}\exp_not:n{##1}}}
297   }
298
299   \tl_set:Nn #3 {\cs_generate_from_arg_count:NNnn #1 \cs_set:Npn}
300   \tl_put_right:Nx #3 { {\int_use:N \l_tmpa_int}{
301     \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
302     \exp_after:wN\exp_after:wN\exp_after:wN {
303       \exp_after:wN #2 \_tmp_args_tl
304     }
305   }}
306 } %% TODO check if this works!
307 \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {cNN}
308 \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {NcN}
309 \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {ccN}

```

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

`\MMTrule`

```

310 \NewDocumentCommand \MMTrule {m m}{
311   \seq_set_split:Nnn \l_tmpa_seq , {#2}
312   \int_zero:N \l_tmpa_int
313   \stex_annotate_invisible:nnn{mmtrule}{scala://#1}{
314     $\seq_map_inline:Nn \l_tmpa_seq {
315       \int_incr:N \l_tmpa_int
316       \stex_annotate:nnn{arg}{i\int_use:N \l_tmpa_int}{##1}
317     }$
318   }
319 }
320
321 \NewDocumentCommand \MMTinclude {m}{

```

```

322 \stex_annotate_invisible:nnn{import}{#1}{  

323 }  

324 \endpackage

```

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

## Chapter 25

# STEX -MathHub Implementation

```
325 <*package>
326
327 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
328
329 <@@=stex_path>
330
331 Warnings and error messages
332 \msg_new:nnn{stex}{error/norepository}{
333   No~archive~#1~found~in~#2
334 }
335 \msg_new:nnn{stex}{error/notinarchive}{
336   Not~currently~in~an~archive,~but~\detokenize{#1}~
337   needs~one!
338 }
339 \msg_new:nnn{stex}{error/nofile}{
340   \detokenize{#1}~could~not~find~file~#2
341 }
342 \msg_new:nnn{stex}{error/twofiles}{
343   \detokenize{#1}~found~two~candidates~for~#2
344 }
```

### 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`

```
343 \cs_new_protected:Nn \stex_path_from_string:Nn {
344   \str_set:Nx \l_tmpa_str { #2 }
345   \str_if_empty:NTF \l_tmpa_str {
346     \seq_clear:N #1
347   }{
348     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
349     \sys_if_platform_windows:T{
350       \seq_clear:N \l_tmpa_tl
```

```

351     \seq_map_inline:Nn #1 {
352       \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
353       \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
354     }
355     \seq_set_eq:NN #1 \l_tmpa_tl
356   }
357   \stex_path_canonicalize:N #1
358 }
359 }
360

```

(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`

```

361 \cs_new_protected:Nn \stex_path_to_string:NN {
362   \exp_args:Nne \str_set:Nn #2 { \seq_use:Nn #1 / }
363 }
364
365 \cs_new:Nn \stex_path_to_string:N {
366   \seq_use:Nn #1 /
367 }

```

(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`

```

368 \str_const:Nn \c__stex_path_dot_str {.}
369 \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.

```

370 \cs_new_protected:Nn \stex_path_canonicalize:N {
371   \seq_if_empty:NF #1 {
372     \seq_clear:N \l_tmpa_seq
373     \seq_get_left:NN #1 \l_tmpa_tl
374     \str_if_empty:NT \l_tmpa_tl {
375       \seq_put_right:Nn \l_tmpa_seq {}
376     }
377     \seq_map_inline:Nn #1 {
378       \str_set:Nn \l_tmpa_tl { ##1 }
379       \str_if_eq:NNF \l_tmpa_tl \c__stex_path_dot_str {
380         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
381           \seq_if_empty:NNTF \l_tmpa_seq {
382             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
383               \c__stex_path_up_str
384             }
385           }{
386             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
387             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
388               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
389                 \c__stex_path_up_str
390               }
391             }{

```



```

392         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
393     }
394 }
395 }{
396     \str_if_empty:NF \l_tmpa_tl {
397         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
398     }
399 }
400 }
401 }
402 \seq_gset_eq:NN #1 \l_tmpa_seq
403 }
404 }

```

(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 $\underline{TF}$`

```

405 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
406     \seq_if_empty:NTF #1 {
407         \prg_return_false:
408     }{
409         \seq_get_left:NN #1 \l_tmpa_tl
410         \sys_if_platform_windows:TF{
411             \str_if_in:NnTF \l_tmpa_tl {:}{
412                 \prg_return_true:
413             }{
414                 \prg_return_false:
415             }
416         }{
417             \str_if_empty:NTF \l_tmpa_tl {
418                 \prg_return_true:
419             }{
420                 \prg_return_false:
421             }
422         }
423     }
424 }

```

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

## 25.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

425 \str_new:N\l_stex_kpsewhich_return_str
426 \cs_new_protected:Nn \stex_kpsewhich:n {
427     \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
428     \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
429     \tl_trim_spaces:N \l_stex_kpsewhich_return_str
430 }

```

(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`

```

431 \sys_if_platform_windows:TF{
432   \begingroup\escapechar=-1\catcode'\=12
433   \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
434   \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
435   \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_
436   }}{
437   \stex_kpsewhich:n{-var-value~PWD}
438   }
439
440 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
441 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
442 \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

```

443 <@@=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

```

444 \seq_gclear_new:N\g__stex_files_stack

```

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`  
`\c_stex_mainfile_str`

```

445 \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
446 \stex_path_from_string:Nn \c_stex_mainfile_seq
447   \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`

```

448 \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`

```

449 \cs_new_protected:Nn \stex_filestack_push:n {
450   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
451   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
452     \stex_path_from_string:Nn\g_stex_currentfile_seq{
453       \c_stex_pwd_str/#1
454     }
455   }
456   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
457   \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
458 }

```

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

`\stex_filestack_pop:`

```

459 \cs_new_protected:Nn \stex_filestack_pop: {
460   \seq_if_empty:NF\g__stex_files_stack{
461     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
462   }
463   \seq_if_empty:NTF\g__stex_files_stack{
464     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
465   }{
466     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
467     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
468   }
469 }
```

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

Hooks for the current file:

```

470 \AddToHook{file/before}{
471   \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
472 }
473 \AddToHook{file/after}{
474   \stex_filestack_pop:
475 }
```

## 25.4 MathHub Repositories

476 `<@=stex_mathhub>`

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

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

```

477 \str_if_empty:NTF\mathhub{
478   \sys_if_platform_windows:TF{
479     \begingroup\escapechar=-1\catcode'\=12
480     \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str MATHHUB\c_percent_str}
481     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
482     \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_kpsewhich_return_str}}
483   }{
484     \stex_kpsewhich:n{-var-value-MATHHUB}
485   }
486   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
487 }
488 \str_if_empty:NTF\c_stex_mathhub_str{
489   \msg_warning:nn{stex}{warning/nomathhub}
490 }{
491   \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
492   \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
493 }
494 }{
495   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
496   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
497     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
498       \c_stex_pwd_str/\mathhub
499     }
500   }
```

```

500 }
501 \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
502 \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
503 }

```

(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

```

504 \cs_new_protected:Nn \_stex_mathhub_do_manifest:n {
505   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
506     \str_set:Nx \l_tmpa_str { #1 }
507     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
508     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
509     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
510     \_stex_mathhub_find_manifest:N \l_tmpa_seq
511     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
512       \msg_error:nnxx{stex}{error/norepository}{#1}{
513         \stex_path_to_string:N \c_stex_mathhub_str
514       }
515     } {
516       \exp_args:No \_stex_mathhub_parse_manifest:n { \l_tmpa_str }
517     }
518   }
519 }

```

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

`\l__stex_mathhub_manifest_file_seq`

```

520 \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`:

```

521 \cs_new_protected:Nn \_stex_mathhub_find_manifest:N {
522   \seq_set_eq:NN\l_tmpa_seq #1
523   \bool_set_true:N\l_tmpa_bool
524   \bool_while_do:Nn \l_tmpa_bool {
525     \seq_if_empty:NTF \l_tmpa_seq {
526       \bool_set_false:N\l_tmpa_bool
527     }{
528       \file_if_exist:nTF{
529         \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
530       }{
531         \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
532         \bool_set_false:N\l_tmpa_bool
533       }{
534         \file_if_exist:nTF{
535           \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
536         }{
537           \seq_put_right:Nn\l_tmpa_seq{META-INF}
538           \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}

```

```

539         \bool_set_false:N\l_tmpa_bool
540     }{
541         \file_if_exist:nTF{
542             \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
543         }{
544             \seq_put_right:Nn\l_tmpa_seq{meta-inf}
545             \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
546             \bool_set_false:N\l_tmpa_bool
547         }{
548             \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
549         }
550     }
551 }
552 }
553 }
554 \seq_set_eq:NN\l__stex_mathhub_manifest_file_seq\l_tmpa_seq
555 }

```

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

\c\_\_stex\_mathhub\_manifest\_ior File variable used for MANIFEST-files

```

556 \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:

```

557 \cs_new_protected:Nn \__stex_mathhub_parse_manifest:n {
558     \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
559     \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
560     \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
561         \str_set:Nn \l_tmpa_str {##1}
562         \exp_args:NNoo \seq_set_split:Nnn
563             \l_tmpb_seq \c_colon_str \l_tmpa_str
564         \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
565             \exp_args:NNe \str_set:Nn \l_tmpb_tl {
566                 \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
567             }
568             \exp_args:No \str_case:nnTF \l_tmpa_tl {
569                 {id} {
570                     \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
571                     { id } \l_tmpb_tl
572                 }
573                 {narration-base} {
574                     \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
575                     { narr } \l_tmpb_tl
576                 }
577                 {url-base} {
578                     \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
579                     { docurl } \l_tmpb_tl
580                 }
581                 {source-base} {
582                     \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
583                     { ns } \l_tmpb_tl
584                 }

```

```

585     {ns} {
586       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
587       { ns } \l_tmpb_tl
588     }
589     {dependencies} {
590       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
591       { deps } \l_tmpb_tl
592     }
593   }{}{}
594 }{}
595 }
596 \ior_close:N \c__stex_mathhub_manifest_ior
597 }

```

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

`\stex_set_current_repository:n`

```

598 \cs_new_protected:Nn \stex_set_current_repository:n {
599   \stex_require_repository:n { #1 }
600   \prop_set_eq:Nc \l_stex_current_repository_prop {
601     c_stex_mathhub_#1_manifest_prop
602   }
603 }

```

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

`\stex_require_repository:n`

```

604 \cs_new_protected:Nn \stex_require_repository:n {
605   \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
606     \stex_debug:nn{mathhub}{Opening~archive:~#1}
607     \_stex_mathhub_do_manifest:n { #1 }
608   }
609 }

```

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

`\l_stex_current_repository_prop`

Current MathHub repository

```

610 %\prop_new:N \l_stex_current_repository_prop
611
612 \_stex_mathhub_find_manifest:N \c_stex_pwd_seq
613 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
614   \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
615 } {
616   \_stex_mathhub_parse_manifest:n { main }
617   \prop_get:Nn \c_stex_mathhub_main_manifest_prop {id}
618   \l_tmpa_str
619   \prop_set_eq:cN { c_stex_mathhub\_l_tmpa_str_manifest_prop }
620   \c_stex_mathhub_main_manifest_prop
621   \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
622   \stex_debug:nn{mathhub}{Current~repository:~
623     \prop_item:Nn \l_stex_current_repository_prop {id}
624   }
625 }

```

(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.

```

626 \cs_new_protected:Nn \stex_in_repository:nn {
627   \str_set:Nx \l_tmpa_str { #1 }
628   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
629   \str_if_empty:NTF \l_tmpa_str {
630     \prop_if_exist:NTF \l_stex_current_repository_prop {
631       \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
632       \exp_args:Ne \l_tmpa_cs{
633         \prop_item:Nn \l_stex_current_repository_prop { id }
634       }
635     }{
636       \l_tmpa_cs{}
637     }
638   }{
639     \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}
640     \stex_require_repository:n \l_tmpa_str
641     \str_set:Nx \l_tmpa_str { #1 }
642     \exp_args:Nne \use:nn {
643       \stex_set_current_repository:n \l_tmpa_str
644       \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
645     }{
646       \stex_debug:nn{mathhub}{switching~back~to:~
647       \prop_if_exist:NTF \l_stex_current_repository_prop {
648         \prop_item:Nn \l_stex_current_repository_prop { id }::~
649       \meaning\l_stex_current_repository_prop
650       }{
651         no~repository
652       }
653     }
654     \prop_if_exist:NTF \l_stex_current_repository_prop {
655       \stex_set_current_repository:n {
656         \prop_item:Nn \l_stex_current_repository_prop { id }
657       }
658     }{
659       \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
660     }
661   }
662 }
663 }

```

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

## 25.5 Using Content in Archives

`\mhpath`

```

664 \def \mhpath #1 #2 {
665   \exp_args:Ne \tl_if_empty:nTF{#1}{
666     \c_stex_mathhub_str /
667     \prop_item:Nn \l_stex_current_repository_prop { id }
668     / source / #2
669   }{
670     \c_stex_mathhub_str / #1 / source / #2

```

```

671 }
672 }

```

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

`\inputref`  
`\mhinput`

```

673 \newif \ifinputref \inputreffalse
674
675 \cs_new_protected:Nn \__stex_mathhub_mhinput:nn {
676   \stex_in_repository:nn {#1} {
677     \ifinputref
678       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
679     \else
680       \inputreftrue
681       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
682       \inputreffalse
683     \fi
684   }
685 }
686 \NewDocumentCommand \mhinput { 0{} m}{
687   \stex_mhinput:nn{ #1 }{ #2 }
688 }
689
690 \cs_new_protected:Nn \__stex_mathhub_inputref:nn {
691   \stex_in_repository:nn {#1} {
692     \bool_lazy_any:nTF {
693       {\rustex_if_p:}
694       {\latexml_if_p:}
695     } {
696       \str_clear:N \l_tmpa_str
697       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
698         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
699       }
700       \stex_annotate_invisible:nnn{inputref}{
701         \l_tmpa_str / #2
702       }{}
703     }{
704       \begingroup
705         \inputreftrue
706         \tl_if_empty:nTF{ ##1 }{
707           \input{#2}
708         }{
709           \input{ \c_stex_mathhub_str / ##1 / source / #2 }
710         }
711       \endgroup
712     }
713   }
714 }
715 \NewDocumentCommand \inputref { 0{} m}{
716   \__stex_mathhub_inputref:nn{ #1 }{ #2 }
717 }

```

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



**\addmhbibresource**

```

718 \cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {
719   \stex_in_repository:nn {#1} {
720     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
721   }
722 }
723 \newcommand\addmhbibresource[2][]{
724   \__stex_mathhub_mhbibresource:nn{ #1 }{ #2 }
725 }

```

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

**\libinput**

```

726 \cs_new_protected:Npn \libinput #1 {
727   \prop_if_exist:NF \l_stex_current_repository_prop {
728     \msg_error:nnn{stex}{error/notinarchive}\libinput
729   }
730   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
731     \msg_error:nnn{stex}{error/notinarchive}\libinput
732   }
733   \seq_clear:N \l__stex_mathhub_libinput_files_seq
734   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
735   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
736
737   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
738     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}
739     \IfFileExists{ \l_tmpa_str }{
740       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
741     }{}
742     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
743     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
744   }
745
746   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
747   \IfFileExists{ \l_tmpa_str }{
748     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
749   }{}
750
751   \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
752     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
753   }{
754     \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
755       \input{ ##1 }
756     }
757   }
758 }

```

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

**\libusepackage**

```

759 \NewDocumentCommand \libusepackage {0{} m} {
760   \prop_if_exist:NF \l_stex_current_repository_prop {
761     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
762   }

```

```

763 \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
764   \msg_error:nnn{stex}{error/notinarchive}\libusepackage
765 }
766 \seq_clear:N \l__stex_mathhub_libinput_files_seq
767 \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
768 \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
769
770 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
771   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2}
772   \IfFileExists{ \l_tmpa_str.sty }{
773     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
774   }{
775     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
776     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
777   }
778
779   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2}
780   \IfFileExists{ \l_tmpa_str.sty }{
781     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
782   }{
783
784     \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
785       \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
786     }{
787       \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
788         \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
789           \usepackage[#1]{ ##1 }
790         }
791       }{
792         \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
793       }
794     }
795   }

```

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

`\mhgraphics`  
`\cmhgraphics`

```

796
797 \AddToHook{begindocument}{
798   \ltx@ifpackageloaded{graphicx}{
799     \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
800     \newcommand\mhgraphics[2][\%
801       \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
802       \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}
803     \newcommand\cmhgraphics[2][\begin{center}\mhgraphics{#1}{#2}\end{center}}
804   }{

```

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

`\lstinputmhlisting`  
`\cmlstinputmhlisting`

```

805 \ltx@ifpackageloaded{listings}{
806   \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
807   \newcommand\lstinputmhlisting[2][\%
808     \def\lst@mhrepos{}\setkeys{lst}{#1}%
809     \lstinputlisting[#1]{\mhp\lst@mhrepos{#2}}

```

```

810     \newcommand\clstinputmhlisting[2][]{\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
811   }{}
812 }
813
814 \end{package}

```

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

## Chapter 26

# STEX -References Implementation

```
815 <*package>
816
817 %%%%%%%%%% references.dtx %%%%%%%%%%
818
819 <@@=stex_refs>
      Warnings and error messages
820
```

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

```
821 %\iow_new:N \c__stex_refs_refs_iow
822 \AddToHook{begindocument}{
823 % \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
824 }
825 \AddToHook{enddocument}{
826 % \iow_close:N \c__stex_refs_refs_iow
827 }
```

`\STEXreftitle`

```
828 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
829
830 \NewDocumentCommand \STEXreftitle { m } {
831 \tl_gset:Nx \g__stex_refs_title_tl { #1 }
832 }
```

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

### 26.1 Document URIs and URLs

`\l_stex_current_docns_str`

```
833 \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:`

```
834 \cs_new_protected:Nn \stex_get_document_uri: {
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 { narr } \l_tmpa_str {
844       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
845     }
846   }
847
848   \str_if_empty:NTF \l_tmpa_str {
849     \str_set:Nx \l_stex_current_docns_str {
850       file:/\stex_path_to_string:N \l_tmpa_seq
851     }
852   }{
853     \bool_set_true:N \l_tmpa_bool
854     \bool_while_do:Nn \l_tmpa_bool {
855       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
856       \exp_args:No \str_case:nnTF { \l_tmpb_str } {
857         {source} { \bool_set_false:N \l_tmpa_bool }
858       }{}{
859         \seq_if_empty:NT \l_tmpa_seq {
860           \bool_set_false:N \l_tmpa_bool
861         }
862       }
863     }
864
865     \seq_if_empty:NTF \l_tmpa_seq {
866       \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
867     }{
868       \str_set:Nx \l_stex_current_docns_str {
869         \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
870       }
871     }
872   }
873 }
```

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

`\l_stex_current_docurl_str`

```
874 \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:`

```
875 \cs_new_protected:Nn \stex_get_document_url: {
876   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
877   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
878   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
```

```

879 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
880 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
881
882 \str_clear:N \l_tmpa_str
883 \prop_if_exist:NT \l_stex_current_repository_prop {
884   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
885     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
886       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
887     }
888   }
889 }
890
891 \str_if_empty:NTF \l_tmpa_str {
892   \str_set:Nx \l_stex_current_docurl_str {
893     file:/\stex_path_to_string:N \l_tmpa_seq
894   }
895 }{
896   \bool_set_true:N \l_tmpa_bool
897   \bool_while_do:Nn \l_tmpa_bool {
898     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
899     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
900       {source} { \bool_set_false:N \l_tmpa_bool }
901     }{}{
902       \seq_if_empty:NT \l_tmpa_seq {
903         \bool_set_false:N \l_tmpa_bool
904       }
905     }
906   }
907
908   \seq_if_empty:NTF \l_tmpa_seq {
909     \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
910   }{
911     \str_set:Nx \l_stex_current_docurl_str {
912       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
913     }
914   }
915 }
916 }

```

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

## 26.2 Setting Reference Targets

```

917 \str_const:Nn \c__stex_refs_url_str{URL}
918 \str_const:Nn \c__stex_refs_ref_str{REF}
919 \str_new:N \l__stex_refs_curr_label_str
920 % @currentlabel -> number
921 % @currentlabelname -> title
922 % @currentHref -> name.number <- id of some kind
923 % \theH# -> \arabic{section}
924 % \the# -> number
925 % \hyper@makecurrent{#}
926 \int_new:N \l__stex_refs_unnamed_counter_int

```

`\stex_ref_new_doc_target:n`

```

927 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
928   \stex_get_document_uri:
929   \str_clear:N \l__stex_refs_curr_label_str
930   \str_set:Nx \l_tmpa_str { #1 }
931   \str_if_empty:NT \l_tmpa_str {
932     \int_incr:N \l__stex_refs_unnamed_counter_int
933     \str_set:Nx \l_tmpa_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
934   }
935   \str_set:Nx \l__stex_refs_curr_label_str {
936     \l_stex_current_docns_str?\l_tmpa_str
937   }
938   \seq_if_exist:cF{g__stex_refs_labels_\l_tmpa_str_seq}{
939     \seq_new:c {g__stex_refs_labels_\l_tmpa_str_seq}
940   }
941   \seq_if_in:coF{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str {
942     \seq_gput_right:co{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str
943   }
944   \stex_if_smsmode:TF {
945     \stex_get_document_url:
946     \str_gset_eq:cN {sref_url_\l__stex_refs_curr_label_str_str}\l_stex_current_docurl_str
947     \str_gset_eq:cN {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_url_str
948   }{
949     %\iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter\unexpanded\expandafter{
950     \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
951     \immediate\write\@auxout{\stexauxadddocref{\l_stex_current_docns_str}{\l_tmpa_str}}
952     \str_gset:cx {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_ref_str
953   }
954 }

```

(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.

```

955 \cs_new_protected:Npn \stexauxadddocref #1 #2 {
956   \str_set:Nn \l_tmpa_str {#1?#2}
957   \str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
958   \seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
959     \seq_new:c {g__stex_refs_labels_#2_seq}
960   }
961   \seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
962     \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
963   }
964 }

```

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

```

965 \AtEndDocument{
966   \def\stexauxadddocref#1 #2 {}{}
967 }

```

`\stex_ref_new_sym_target:n`

```

968 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
969   \stex_if_smsmode:TF {
970     \str_if_exist:cF{sref_sym_#1_type}{
971       \stex_get_document_url:
972       \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str

```

```

973     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
974   }
975 }{
976   \str_if_empty:NF \l__stex_refs_curr_label_str {
977     \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
978     \immediate\write\@auxout{
979       \exp_not:N\expandafter\def\exp_not:N\csname \exp_not:N\detokenize{sref_sym_#1_label_
980         \l__stex_refs_curr_label_str
981       }
982     }
983   }
984 }
985 }

```

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

## 26.3 Using References

```

986 \str_new:N \l__stex_refs_indocument_str

```

**\sref** Optional arguments:

```

987
988 \keys_define:nn { stex / sref } {
989   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
990   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
991   pre           .tl_set:N = \l__stex_refs_pre_tl ,
992   post          .tl_set:N = \l__stex_refs_post_tl ,
993 }
994 \cs_new_protected:Nn \__stex_refs_args:n {
995   \tl_clear:N \l__stex_refs_linktext_tl
996   \tl_clear:N \l__stex_refs_fallback_tl
997   \tl_clear:N \l__stex_refs_pre_tl
998   \tl_clear:N \l__stex_refs_post_tl
999   \str_clear:N \l__stex_refs_repo_str
1000   \keys_set:nn { stex / sref } { #1 }
1001 }

```

The actual macro:

```

1002 \NewDocumentCommand \sref { 0{} m}{
1003   \__stex_refs_args:n { #1 }
1004   \str_if_empty:NTF \l__stex_refs_indocument_str {
1005     \str_set:Nx \l_tmpa_str { #2 }
1006     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
1007     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
1008       \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
1009         \seq_get_left:cNF {g__stex_refs_labels_\l_tmpa_str _seq} \l_tmpa_str {
1010           \str_clear:N \l_tmpa_str
1011         }
1012       }{
1013         \str_clear:N \l_tmpa_str
1014       }
1015     }{
1016       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1017       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str

```



```

1018 \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
1019 \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str_seq}{
1020   \str_set_eq:NN \l_tmpc_str \l_tmpa_str
1021   \str_clear:N \l_tmpa_str
1022   \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str_seq} {
1023     \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
1024       \str_range:nnn { ##1 }{-\l_tmpa_int}{ -1 }
1025     }{
1026       \seq_map_break:n {
1027         \str_set:Nn \l_tmpa_str { ##1 }
1028       }
1029     }
1030   }
1031 }{
1032   \str_clear:N \l_tmpa_str
1033 }
1034 }
1035 \str_if_empty:NTF \l_tmpa_str {
1036   \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_lin
1037 }{
1038   \str_if_eq:cNTF {sref_\l_tmpa_str_type} \c__stex_refs_ref_str {
1039     \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1040       \cs_if_exist:cTF{autoref}{
1041         \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1042       }{
1043         \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1044       }
1045     }{
1046       \ltx@ifpackageloaded{hyperref}{
1047         \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
1048       }{
1049         \l__stex_refs_linktext_tl
1050       }
1051     }
1052   }{
1053     \ltx@ifpackageloaded{hyperref}{
1054       \href{\use:c{sref_url_\l_tmpa_str_str}}{\tl_if_empty:NTF \l__stex_refs_linktext_t
1055     }{
1056       \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs
1057     }
1058   }
1059 }
1060 }{
1061   % TODO
1062 }
1063 }

```

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

## \srefsym

```

1064 \NewDocumentCommand \srefsym { 0{} m }{
1065   \stex_get_symbol:n { #2 }
1066   \__stex_refs_sym_aux:nn{##1}{\l_stex_get_symbol_uri_str}
1067 }

```

```

1068
1069 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1070   \str_if_exist:cTF {sref_sym_#2 _label_str }{
1071     \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1072   }{
1073     \__stex_refs_args:n { #1 }
1074     \str_if_empty:NTF \l__stex_refs_indocument_str {
1075       \tl_if_exist:cTF{sref_sym_#2 _type}{
1076         % doc uri in \l_tmpb_str
1077         \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1078         \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1079           % reference
1080           \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1081             \cs_if_exist:cTF{autoref}{
1082               \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1083             }{
1084               \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1085             }
1086           }{
1087             \ltx@ifpackageloaded{hyperref}{
1088               \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1089             }{
1090               \l__stex_refs_linktext_tl
1091             }
1092           }
1093         }{
1094           % URL
1095           \ltx@ifpackageloaded{hyperref}{
1096             \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl \
1097           }{
1098             \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_re
1099           }
1100         }
1101       }{
1102         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_l
1103       }
1104     }{
1105       % TODO
1106     }
1107   }
1108 }

```

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

**\srefsymuri**

```

1109 \cs_new_protected:Npn \srefsymuri #1 #2 {
1110   \__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1111 }

```

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

```

1112 </package>

```

## Chapter 27

# STEX -Modules Implementation

```
1113 <*package>
1114
1115 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1116
1117 <@@=stex_modules>
1118
1119 Warnings and error messages
1120 \msg_new:nnn{stex}{error/unknownmodule}{
1121   No~module~#1~found
1122 }
1123 \msg_new:nnn{stex}{error/syntax}{
1124   Syntax~error:~#1
1125 }
1126 \msg_new:nnn{stex}{error/siglanguage}{
1127   Module~#1~declares~signature~#2,~but~does~not~
1128   declare~its~language
1129 }
1130 \msg_new:nnn{stex}{warning/deprecated}{
1131   #1~is~deprecated;~please~use~#2~instead!
1132 }
1133 \msg_new:nnn{stex}{error/conflictingmodules}{
1134   Conflicting~imports~for~module~#1
1135 }
```

**\l\_stex\_current\_module\_str** The current module:

```
1135 \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

```
1136 \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
1137 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1138   \str_if_empty:NTF \l_stex_current_module_str
1139   \prg_return_false: \prg_return_true:
1140 }

(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
1141 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1142   \prop_if_exist:cTF { c_stex_module_#1_prop }
1143   \prg_return_true: \prg_return_false:
1144 }

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

```

```

\stex_add_to_current_module:n
\STEXexport
1145 \cs_new_protected:Nn \stex_add_to_current_module:n {
1146   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1147 }
1148 \cs_new_protected:Npn \STEXexport {
1149   \begingroup
1150   \newlinechar=-1\relax
1151   \endlinechar=-1\relax
1152   %\catcode'\ = 9\relax
1153   \expandafter\endgroup\__stex_modules_export:n
1154 }
1155 \cs_new_protected:Nn \__stex_modules_export:n {
1156   \ignorespaces #1
1157   \stex_add_to_current_module:n { \ignorespaces #1 }
1158   \stex_smsmode_do:
1159 }
1160 \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
1161 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1162   \str_set:Nx \l_tmpa_str { #1 }
1163   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1164 }

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

```

```

\stex_add_import_to_current_module:n
1165 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1166   \str_set:Nx \l_tmpa_str { #1 }
1167   \exp_args:Nno
1168   \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1169     \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1170   }
1171 }

```

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

`\stex_collect_imports:n`

```

1172 \cs_new_protected:Nn \stex_collect_imports:n {
1173   \seq_clear:N \l_stex_collect_imports_seq
1174   \__stex_modules_collect_imports:n {#1}
1175 }
1176 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1177   \seq_map_inline:cn {c_stex_module_#1_imports} {
1178     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1179       \__stex_modules_collect_imports:n { ##1 }
1180     }
1181   }
1182   \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1183     \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1184   }
1185 }

```

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

`\stex_do_up_to_module:n`

```

1186 \int_new:N \l__stex_modules_group_depth_int
1187 \cs_new_protected:Nn \stex_do_up_to_module:n {
1188   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1189     #1
1190   }{
1191     #1
1192     \expandafter \tl_gset:Nn
1193     \csname l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1194     \expandafter\expandafter\expandafter\endcsname
1195     \expandafter\expandafter\expandafter { \csname
1196       l__stex_modules_aftergroup_\l_stex_current_module_str _tl\endcsname #1 }
1197     \aftergroup\__stex_modules_aftergroup_do:
1198   }
1199 }
1200 \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1201   \stex_debug:nn{aftergroup}{\cs_meaning:c{
1202     l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1203   }}
1204   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1205     \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1206     \tl_gclear:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1207   }{
1208     \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1209     \aftergroup\__stex_modules_aftergroup_do:
1210   }
1211 }
1212 \cs_new_protected:Nn \stex_reset_up_to_module:n {
1213   \expandafter\let\csname l__stex_modules_aftergroup_#1_tl\endcsname\undefined
1214 }

```

(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).

1215

(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.

```

1216 \str_new:N \l_stex_modules_ns_str
1217 \str_new:N \l_stex_modules_subpath_str
1218 \cs_new_protected:Nn __stex_modules_compute_namespace:nN {
1219   \str_set:Nx \l_tmpa_str { #1 }
1220   \seq_set_eq:NN \l_tmpa_seq #2
1221   % split off file extension
1222   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1223   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1224   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1225   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1226
1227   \bool_set_true:N \l_tmpa_bool
1228   \bool_while_do:Nn \l_tmpa_bool {
1229     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1230     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1231       {source} { \bool_set_false:N \l_tmpa_bool }
1232     }{}{
1233       \seq_if_empty:NT \l_tmpa_seq {
1234         \bool_set_false:N \l_tmpa_bool
1235       }
1236     }
1237   }
1238
1239   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
1240   \str_if_empty:NTF \l_stex_modules_subpath_str {
1241     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
1242   }{
1243     \str_set:Nx \l_stex_modules_ns_str {
1244       \l_tmpa_str/\l_stex_modules_subpath_str
1245     }
1246   }
1247 }
1248
1249 \cs_new_protected:Nn \stex_modules_current_namespace: {
1250   \str_clear:N \l_stex_modules_subpath_str
1251   \prop_if_exist:NTF \l_stex_current_repository_prop {
1252     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1253     __stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1254   }{
1255     % split off file extension
1256     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1257     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1258     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1259     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1260     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1261     \str_set:Nx \l_stex_modules_ns_str {

```

```

1262     file:/\stex_path_to_string:N \l_tmpa_seq
1263   }
1264 }
1265 }

```

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

## 27.1 The smodule environment

smodule arguments:

```

1266 \keys_define:nn { stex / module } {
1267   title      .tl_set:N      = \smoduletitle ,
1268   type       .str_set_x:N    = \smoduletype ,
1269   id         .str_set_x:N    = \smoduleid ,
1270   deprecate  .str_set_x:N    = \l_stex_module_deprecate_str ,
1271   ns         .str_set_x:N    = \l_stex_module_ns_str ,
1272   lang       .str_set_x:N    = \l_stex_module_lang_str ,
1273   sig        .str_set_x:N    = \l_stex_module_sig_str ,
1274   creators   .str_set_x:N    = \l_stex_module_creators_str ,
1275   contributors .str_set_x:N  = \l_stex_module_contributors_str ,
1276   meta       .str_set_x:N    = \l_stex_module_meta_str ,
1277   srccite    .str_set_x:N    = \l_stex_module_srccite_str
1278 }
1279
1280 \cs_new_protected:Nn \__stex_modules_args:n {
1281   \str_clear:N \smoduletitle
1282   \str_clear:N \smoduletype
1283   \str_clear:N \smoduleid
1284   \str_clear:N \l_stex_module_ns_str
1285   \str_clear:N \l_stex_module_deprecate_str
1286   \str_clear:N \l_stex_module_lang_str
1287   \str_clear:N \l_stex_module_sig_str
1288   \str_clear:N \l_stex_module_creators_str
1289   \str_clear:N \l_stex_module_contributors_str
1290   \str_clear:N \l_stex_module_meta_str
1291   \str_clear:N \l_stex_module_srccite_str
1292   \keys_set:nn { stex / module } { #1 }
1293 }
1294
1295 % module parameters here? In the body?
1296

```

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

```

1297 \cs_new_protected:Nn \stex_module_setup:nn {
1298   \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1299   \str_set:Nx \l_stex_module_name_str { #2 }
1300   \__stex_modules_args:n { #1 }

```

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

Are we in a nested module?

```

1301 \stex_if_in_module:TF {
1302   % Nested module
1303   \prop_get:cnN {c_stex_module_\l_stex_current_module_str _prop}

```

```

1304     { ns } \l_stex_module_ns_str
1305     \str_set:Nx \l_stex_module_name_str {
1306       \prop_item:cn {c_stex_module\_l_stex_current_module_str _prop}
1307       { name } / \l_stex_module_name_str
1308   }
1309 }{
1310   % not nested:
1311   \str_if_empty:NT \l_stex_module_ns_str {
1312     \stex_modules_current_namespace:
1313     \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1314     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1315       / {\l_stex_module_ns_str}
1316     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1317     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1318       \str_set:Nx \l_stex_module_ns_str {
1319         \stex_path_to_string:N \l_tmpa_seq
1320       }
1321     }
1322   }
1323 }

```

Next, we determine the language of the module:

```

1324   \str_if_empty:NT \l_stex_module_lang_str {
1325     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1326     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1327     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1328     \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1329     \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1330       \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1331       \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1332         inferred~from~file~name}
1333     }
1334   }
1335
1336   \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
1337     \prop_get:NVNTF {c_stex_languages_prop \l_stex_module_lang_str
1338       \l_tmpa_str {
1339       \ltx@ifpackageloaded{babel}{
1340         \exp_args:Nx \selectlanguage { \l_tmpa_str }
1341       }{}
1342     } {
1343       \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
1344     }
1345   }}

```

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

```

1346   \str_if_empty:NTF \l_stex_module_sig_str {
1347     \exp_args:Nnx \prop_gset_from_keyval:cn {
1348       c_stex_module\_l_stex_module_ns_str?\l_stex_module_name_str _prop
1349     } {
1350       name      = \l_stex_module_name_str ,
1351       ns        = \l_stex_module_ns_str ,
1352       file      = \exp_not:o { \g_stex_currentfile_seq } ,

```



```

1353     lang      = \l_stex_module_lang_str ,
1354     sig       = \l_stex_module_sig_str ,
1355     deprecate = \l_stex_module_deprecate_str ,
1356     meta      = \l_stex_module_meta_str
1357 }
1358 \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1359 \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1360 \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _copymodules}
1361 \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1362 \str_set:Nx \l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1363 \str_if_empty:NT \l_stex_module_meta_str {
1364   \str_set:Nx \l_stex_module_meta_str {
1365     \c_stex_metatheory_ns_str ? Metatheory
1366   }
1367 }
1368 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1369   \bool_set_true:N \l_stex_in_meta_bool
1370   \exp_args:Nx \stex_add_to_current_module:n {
1371     \bool_set_true:N \l_stex_in_meta_bool
1372     \stex_activate_module:n {\l_stex_module_meta_str}
1373     \bool_set_false:N \l_stex_in_meta_bool
1374   }
1375   \stex_activate_module:n {\l_stex_module_meta_str}
1376   \bool_set_false:N \l_stex_in_meta_bool
1377 }
1378 }{
1379   \str_if_empty:NT \l_stex_module_lang_str {
1380     \msg_error:nnxx{stex}{error/siglanguage}{
1381       \l_stex_module_ns_str?\l_stex_module_name_str
1382     }\l_stex_module_sig_str}
1383   }
1384   \stex_debug:nn{modules}{Signature~\l_stex_module_sig_str~for~\l_stex_module_ns_str?\l_st
1385   \stex_if_module_exists:nTF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1386     \stex_debug:nn{modules}{(already exists)}
1387   }{
1388     \stex_debug:nn{modules}{(needs loading)}
1389     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1390     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1391     \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1392     \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1393     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1394     \str_set:Nx \l_tmpa_str {
1395       \stex_path_to_string:N \l_tmpa_seq /
1396       \l_tmpa_str . \l_stex_module_sig_str .tex
1397     }
1398     \IfFileExists \l_tmpa_str {
1399       \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1400         \str_clear:N \l_stex_current_module_str
1401         \seq_clear:N \l_stex_all_modules_seq
1402         \stex_debug:nn{modules}{Loading~signature}
1403       }
1404     }{

```

```

1405     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1406   }
1407 }
1408 \stex_if_smsmode:F {
1409   \stex_activate_module:n {
1410     \l_stex_module_ns_str ? \l_stex_module_name_str
1411   }
1412 }
1413 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1414 }
1415 \str_if_empty:NF \l_stex_module_deprecate_str {
1416   \msg_warning:nnxx{stex}{warning/deprecated}{
1417     Module~\l_stex_current_module_str
1418   }{
1419     \l_stex_module_deprecate_str
1420   }
1421 }
1422 \seq_put_right:Nx \l_stex_all_modules_seq {
1423   \l_stex_module_ns_str ? \l_stex_module_name_str
1424 }
1425 \tl_clear:c{l__stex_modules_aftergroup_\l_stex_module_ns_str ? \l_stex_module_name_str _tl}
1426 }

```

(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}
1427 \cs_new_protected:Nn \__stex_modules_begin_module: {
1428   \stex_reactivate_macro:N \STEXexport
1429   \stex_reactivate_macro:N \importmodule
1430   \stex_reactivate_macro:N \symdecl
1431   \stex_reactivate_macro:N \notation
1432   \stex_reactivate_macro:N \symdef
1433 }
1434 \stex_debug:nn{modules}{
1435   New~module:\\
1436   Namespace:~\l_stex_module_ns_str\\
1437   Name:~\l_stex_module_name_str\\
1438   Language:~\l_stex_module_lang_str\\
1439   Signature:~\l_stex_module_sig_str\\
1440   Metatheory:~\l_stex_module_meta_str\\
1441   File:~\stex_path_to_string:N \g_stex_currentfile_seq
1442 }
1443
1444 \stex_if_smsmode:F{
1445   \begin{stex_annotate_env} {theory} {
1446     \l_stex_module_ns_str ? \l_stex_module_name_str
1447   }
1448
1449   \stex_annotate_invisible:nnn{header}{} {
1450     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1451     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1452     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {

```

```

1453     \stex_annotate:nnn{metatheory}{\l_stex_module_meta_str }{}
1454   }
1455   \str_if_empty:NF \smoduletype {
1456     \stex_annotate:nnn{type}{\smoduletype}{}
1457   }
1458 }
1459 }
1460 % TODO: Inherit metatheory for nested modules?
1461 }
1462 \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}

```

1463 \cs_new_protected:Nn \_stex_modules_end_module: {
1464   \stex_debug:nn{modules}{Closing~module~\prop_item:cn {c_stex_module\_l_stex_current_module}}
1465   \stex_reset_up_to_module:n \l_stex_current_module_str
1466 }

```

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

The core environment

```

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

```

```

1500     \tl_if_empty:NTF \l_tmpa_tl {
1501         \__stex_modules_smodule_end:
1502     }{
1503         \l_tmpa_tl
1504     }
1505 }
1506 }

```

## `\stexpatchmodule`

```

1507 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1508 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1509
1510 \newcommand\stexpatchmodule[3] [] {
1511     \str_set:Nx \l_tmpa_str{ #1 }
1512     \str_if_empty:NTF \l_tmpa_str {
1513         \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1514         \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1515     }{
1516         \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1517         \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1518     }
1519 }

```

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

## 27.2 Invoking modules

### `\STEXModule` `\stex_invoke_module:n`

```

1520 \NewDocumentCommand \STEXModule { m } {
1521     \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1522     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1523     \tl_set:Nn \l_tmpa_tl {
1524         \msg_error:nnx{stex}{error/unknownmodule}{#1}
1525     }
1526     \seq_map_inline:Nn \l_stex_all_modules_seq {
1527         \str_set:Nn \l_tmpb_str { ##1 }
1528         \str_if_eq:eeT { \l_tmpa_str } {
1529             \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1530         } {
1531             \seq_map_break:n {
1532                 \tl_set:Nn \l_tmpa_tl {
1533                     \stex_invoke_module:n { ##1 }
1534                 }
1535             }
1536         }
1537     }
1538     \l_tmpa_tl
1539 }
1540
1541 \cs_new_protected:Nn \stex_invoke_module:n {
1542     \stex_debug:nn{modules}{Invoking~module~#1}
1543     \peek_charcode_remove:NTF ! {
1544         \__stex_modules_invoke_uri:nN { #1 }

```

```

1545 } {
1546   \peek_charcode_remove:NTF ? {
1547     \__stex_modules_invoke_symbol:nn { #1 }
1548   } {
1549     \msg_error:nnx{stex}{error/syntax}{
1550       ?~or~!~expected~after~
1551       \c_backslash_str STEXModule{#1}
1552     }
1553   }
1554 }
1555 }
1556
1557 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1558   \str_set:Nn #2 { #1 }
1559 }
1560
1561 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1562   \stex_invoke_symbol:n{#1?#2}
1563 }

```

(End definition for `\STEXModule` and `\stex_invoke_module:n`. These functions are documented on page 55.)

`\stex_activate_module:n`

```

1564 \bool_new:N \l_stex_in_meta_bool
1565 \bool_set_false:N \l_stex_in_meta_bool
1566 \cs_new_protected:Nn \stex_activate_module:n {
1567   \stex_debug:nn{modules}{Activating~module~#1}
1568   \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1569     \msg_error:nnn{stex}{error/conflictingmodules}{ #1 }
1570   }
1571   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1572     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1573     \use:c{ c_stex_module_#1_code }
1574   }
1575 }

```

(End definition for `\stex_activate_module:n`. This function is documented on page 56.)

```

1576 </package>

```

## Chapter 28

# STEX -Module Inheritance Implementation

```
1577 <*package>
1578
1579 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1580
```

### 28.1 SMS Mode

```
1581 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1582 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1583 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1584 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1585
1586 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1587   \makeatletter
1588   \makeatother
1589   \ExplSyntaxOn
1590   \ExplSyntaxOff
1591   \rustexBREAK
1592 }
1593
1594 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1595   \symdef
1596   \importmodule
1597   \notation
1598   \symdecl
1599   \STEXexport
1600   \inlineass
1601   \inlinedef
1602   \inlineex
1603   \endinput
1604   \setnotation
```

```

1605 \copynotation
1606 \assign
1607 \renamedekl
1608 \donotcopy
1609 \instantiate
1610 }
1611
1612 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1613   \tl_to_str:n {
1614     smodule,
1615     copymodule,
1616     interpretmodule,
1617     sdefinition,
1618     sexample,
1619     sassertion,
1620     sparagraph,
1621     mathstructure
1622   }
1623 }

```

(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
1624 \bool_new:N \g__stex_smsmode_bool
1625 \bool_set_false:N \g__stex_smsmode_bool
1626 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1627   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1628 }

```

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

```

\__stex_smsmode_in_smsmode:nn
1629 \cs_new_protected:Nn \__stex_smsmode_in_smsmode:nn {
1630   \vbox_set:Nn \l_tmpa_box {
1631     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1632     \bool_gset_true:N \g__stex_smsmode_bool
1633     #2
1634     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1635   }
1636   \box_clear:N \l_tmpa_box
1637 }

```

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

```

\stex_file_in_smsmode:nn
1638 \quark_new:N \q__stex_smsmode_break
1639
1640 \NewDocumentCommand \__stex_smsmode_importmodule: { 0{} m } {
1641   \seq_gput_right:Nn \l__stex_smsmode_importmodules_seq { {#1}{#2} }
1642   \stex_smsmode_do:
1643 }
1644
1645 \cs_new_protected:Nn \__stex_smsmode_module:nn {
1646   \__stex_modules_args:n{#1}

```

```

1647 \str_if_empty:NF \l_stex_module_sig_str {
1648   \stex_if_module_exists:nF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1649     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1650     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1651     \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1652     \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1653     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1654     \str_set:Nx \l_tmpa_str {
1655       \stex_path_to_string:N \l_tmpa_seq /
1656       \l_tmpa_str . \l_stex_module_sig_str .tex
1657     }
1658     \IfFileExists \l_tmpa_str {
1659       \exp_args:NNx \seq_gput_right:Nn \l__stex_smsmode_sigmodules_seq \l_tmpa_str
1660     }{
1661       \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1662     }
1663   }
1664 }
1665 }
1666
1667 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1668   \stex_filestack_push:n{#1}
1669   \seq_gclear:N \l__stex_smsmode_importmodules_seq
1670   \seq_gclear:N \l__stex_smsmode_sigmodules_seq
1671   % ----- new -----
1672   \__stex_smsmode_in_smsmode:nn{#1}{
1673     \let\importmodule\__stex_smsmode_importmodule:
1674     \let\stex_module_setup:nn\__stex_smsmode_module:nn
1675     \let\__stex_modules_begin_module:\relax
1676     \let\__stex_modules_end_module:\relax
1677     \seq_clear:N \g_stex_smsmode_allowedenvs_seq
1678     \exp_args:NNx \seq_put_right:Nn \g_stex_smsmode_allowedenvs_seq {\tl_to_str:n{module}}
1679     \tl_clear:N \g_stex_smsmode_allowedmacros_tl
1680     \tl_clear:N \g_stex_smsmode_allowedmacros_escape_tl
1681     \tl_put_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\importmodule}
1682     \everyeof{\q__stex_smsmode_break\noexpand}
1683     \expandafter\expandafter\expandafter
1684     \stex_smsmode_do:
1685     \csname @ @ input\endcsname "#1"\relax
1686
1687     \seq_map_inline:Nn \l__stex_smsmode_sigmodules_seq {
1688       \stex_filestack_push:n{##1}
1689       \expandafter\expandafter\expandafter
1690       \stex_smsmode_do:
1691       \csname @ @ input\endcsname "##1"\relax
1692       \stex_filestack_pop:
1693     }
1694   }
1695   % ----- new -----
1696   \__stex_smsmode_in_smsmode:nn{#1} {
1697     #2
1698     % ----- new -----
1699     \begingroup
1700     %\stex_debug:nn{smsmode}{Here:~\seq_use:Nn\l__stex_smsmode_importmodules_seq, }

```



```

1701 \seq_map_inline:Nn \l__stex_smsmode_importmodules_seq {
1702 \stex_import_module_uri:nn ##1
1703 \stex_import_require_module:nnnn
1704 \l_stex_import_ns_str
1705 \l_stex_import_archive_str
1706 \l_stex_import_path_str
1707 \l_stex_import_name_str
1708 }
1709 \endgroup
1710 \stex_debug:nn{smsmode}{Actually~loading~file~#1}
1711 % ----- new -----
1712 \everyeof{\q__stex_smsmode_break\noexpand}
1713 \expandafter\expandafter\expandafter
1714 \stex_smsmode_do:
1715 \csname @ @ input\endcsname "#1"\relax
1716 }
1717 \stex_filestack_pop:
1718 }

```

(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:

```

1719 \cs_new_protected:Npn \stex_smsmode_do: {
1720 \stex_if_smsmode:T {
1721 \__stex_smsmode_do:w
1722 }
1723 }
1724 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1725 \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
1726 \expandafter\if\expandafter\relax\noexpand#1
1727 \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1728 \else\expandafter\__stex_smsmode_do:w\fi
1729 }{
1730 \__stex_smsmode_do:w % #1
1731 }
1732 }
1733 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1734 \cs_if_eq:NNF #1 \q__stex_smsmode_break {
1735 \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1736 #1\__stex_smsmode_do:w
1737 }{
1738 \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1739 #1
1740 }{
1741 \cs_if_eq:NNTF \begin #1 {
1742 \__stex_smsmode_check_begin:n
1743 }{
1744 \cs_if_eq:NNTF \end #1 {
1745 \__stex_smsmode_check_end:n
1746 }{
1747 \__stex_smsmode_do:w
1748 }
1749 }

```

```

1750     }
1751   }
1752 }
1753 }
1754
1755 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1756   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1757     \begin{#1}
1758   }{
1759     \__stex_smsmode_do:w
1760   }
1761 }
1762 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1763   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1764     \end{#1}\__stex_smsmode_do:w
1765   }{
1766     \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1767   }
1768 }

```

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

## 28.2 Inheritance

```

1769 <@@=stex_importmodule>

```

`\stex_import_module_uri:nn`

```

1770 \cs_new_protected:Nn \stex_import_module_uri:nn {
1771   \str_set:Nx \l_stex_import_archive_str { #1 }
1772   \str_set:Nn \l_stex_import_path_str { #2 }
1773
1774   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1775   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1776   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1777
1778   \stex_modules_current_namespace:
1779   \bool_lazy_all:nTF {
1780     {\str_if_empty_p:N \l_stex_import_archive_str}
1781     {\str_if_empty_p:N \l_stex_import_path_str}
1782     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1783   }{
1784     \str_set_eq:NN \l_stex_import_path_str \l_stex_modules_subpath_str
1785     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1786   }{
1787     \str_if_empty:NT \l_stex_import_archive_str {
1788       \prop_if_exist:NT \l_stex_current_repository_prop {
1789         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1790       }
1791     }
1792     \str_if_empty:NTF \l_stex_import_archive_str {
1793       \str_if_empty:NF \l_stex_import_path_str {
1794         \str_set:Nx \l_stex_import_ns_str {
1795           \l_stex_module_ns_str / \l_stex_import_path_str
1796         }

```

```

1797     }
1798   }{
1799     \stex_require_repository:n \l_stex_import_archive_str
1800     \prop_get:cnN { c_stex_mathhub\l_stex_import_archive_str _manifest_prop } { ns }
1801     \l_stex_import_ns_str
1802     \str_if_empty:NF \l_stex_import_path_str {
1803       \str_set:Nx \l_stex_import_ns_str {
1804         \l_stex_import_ns_str / \l_stex_import_path_str
1805       }
1806     }
1807   }
1808 }
1809 }

```

(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 1810 \str_new:N \l_stex_import_name_str
\l_stex_import_path_str 1811 \str_new:N \l_stex_import_archive_str
\l_stex_import_ns_str 1812 \str_new:N \l_stex_import_path_str
1813 \str_new:N \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:nnnn {<ns>} {<archive-ID>} {<path>} {<name>}
1814 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1815   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1816
1817     %\stex_debug:nn{requiremodule}{Here:\\~1::~~2::~~3::~~4:~#4}
1818
1819     \exp_args:NNxx \seq_set_split:Nnn \l_tmpa_seq {\tl_to_str:n{/}} {#4}
1820     \seq_get_left:NN \l_tmpa_seq \l_tmpc_str
1821
1822     %\stex_debug:nn{requiremodule}{Top~module:\l_tmpc_str}
1823
1824     % archive
1825     \str_set:Nx \l_tmpa_str { #2 }
1826     \str_if_empty:NTF \l_tmpa_str {
1827       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1828     } {
1829       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1830       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1831       \seq_put_right:Nn \l_tmpa_seq { source }
1832     }
1833
1834     % path
1835     \str_set:Nx \l_tmpb_str { #3 }
1836     \str_if_empty:NTF \l_tmpb_str {
1837       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / \l_tmpc_str }
1838
1839       \ltx@ifpackageloaded{babel} {
1840         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1841           { \language } \l_tmpb_str {
1842           \msg_error:nnx{stex}{error/unknownlanguage}{\language}

```

```

1843     }
1844 } {
1845     \str_clear:N \l_tmpb_str
1846 }
1847
1848 %\stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1849 \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1850     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1851 }{
1852     %\stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1853     \IfFileExists{ \l_tmpa_str.tex }{
1854         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1855     }{
1856         % try english as default
1857         %\stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1858         \IfFileExists{ \l_tmpa_str.en.tex }{
1859             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1860         }{
1861             \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1862         }
1863     }
1864 }
1865
1866 } {
1867     \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1868     \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1869
1870     \ltx@ifpackageloaded{babel} {
1871         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1872             { \language } \l_tmpb_str {
1873             \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1874         }
1875     } {
1876         \str_clear:N \l_tmpb_str
1877     }
1878
1879     \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1880
1881 %\stex_debug:nn{modules}{Checking~\l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex}
1882 \IfFileExists{ \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }{
1883     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }
1884 }{
1885     %\stex_debug:nn{modules}{Checking~\l_tmpa_str/\l_tmpc_str.tex}
1886     \IfFileExists{ \l_tmpa_str/\l_tmpc_str.tex }{
1887         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.tex }
1888     }{
1889         % try english as default
1890         %\stex_debug:nn{modules}{Checking~\l_tmpa_str/\l_tmpc_str.en.tex}
1891         \IfFileExists{ \l_tmpa_str/\l_tmpc_str.en.tex }{
1892             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.en.tex }
1893         }{
1894             %\stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1895             \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1896                 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }

```

```

1897     }{
1898         %\stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1899         \IfFileExists{ \l_tmpa_str.tex }{
1900             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1901         }{
1902             % try english as default
1903             %\stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1904             \IfFileExists{ \l_tmpa_str.en.tex }{
1905                 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1906             }{
1907                 \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1908             }
1909         }
1910     }
1911 }
1912 }
1913 }
1914 }
1915
1916 \str_if_eq:eeF{\g__stex_importmodule_file_str}{\seq_use:Nn \g_stex_currentfile_seq /}{
1917     \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
1918         \seq_clear:N \l_stex_all_modules_seq
1919         \str_clear:N \l_stex_current_module_str
1920         \str_set:Nx \l_tmpb_str { #2 }
1921         \str_if_empty:NF \l_tmpb_str {
1922             \stex_set_current_repository:n { #2 }
1923         }
1924         \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1925     }
1926
1927     \stex_if_module_exists:nF { #1 ? #4 } {
1928         \msg_error:nnx{stex}{error/unknownmodule}{
1929             #1?#4~(in~file~\g__stex_importmodule_file_str)
1930         }
1931     }
1932 }
1933
1934 }
1935 \stex_activate_module:n { #1 ? #4 }
1936 }

```

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

## `\importmodule`

```

1937 \NewDocumentCommand \importmodule { 0{} m } {
1938     \stex_import_module_uri:nn { #1 } { #2 }
1939     \stex_debug:nn{modules}{Importing~module:~
1940         \l_stex_import_ns_str ? \l_stex_import_name_str
1941     }
1942     \stex_import_require_module:nnnn
1943     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1944     { \l_stex_import_path_str } { \l_stex_import_name_str }
1945     \stex_if_smsmode:F {
1946         \stex_annotate_invisible:nnn

```

```

1947     {import} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1948   }
1949   \exp_args:Nx \stex_add_to_current_module:n {
1950     \stex_import_require_module:nnnn
1951     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1952     { \l_stex_import_path_str } { \l_stex_import_name_str }
1953   }
1954   \exp_args:Nx \stex_add_import_to_current_module:n {
1955     \l_stex_import_ns_str ? \l_stex_import_name_str
1956   }
1957   \stex_smsmode_do:
1958   \ignorespacesandpars
1959 }
1960 \stex_deactivate_macro:Nn \importmodule {module-environments}

```

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

### `\usemodule`

```

1961 \NewDocumentCommand \usemodule { 0{} m } {
1962   \stex_if_smsmode:F {
1963     \stex_import_module_uri:nn { #1 } { #2 }
1964     \stex_import_require_module:nnnn
1965     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1966     { \l_stex_import_path_str } { \l_stex_import_name_str }
1967     \stex_annotate_invisible:nnn
1968     {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1969   }
1970   \stex_smsmode_do:
1971   \ignorespacesandpars
1972 }

```

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

```

1973 \</package>

```

## Chapter 29

# STEX -Symbols Implementation

```
1974 <*package>
1975
1976 %%%%%%%%%% symbols.dtx %%%%%%%%%%
1977
    Warnings and error messages
1978 \msg_new:nnn{stex}{error/wrongargs}{
1979   args~value~in~symbol~declaration~for~#1~
1980   needs~to~be~i,~a,~b~or~B,~but~#2~given
1981 }
1982 \msg_new:nnn{stex}{error/unknownsymbol}{
1983   No~symbol~#1~found!
1984 }
1985 \msg_new:nnn{stex}{error/seqlength}{
1986   Expected~#1~arguments;~got~#2!
1987 }
1988 \msg_new:nnn{stex}{error/unknownnotation}{
1989   Unknown~notation~#1~for~#2!
1990 }
```

### 29.1 Symbol Declarations

```
1991 <@@=stex_symdecl>
\stex_all_symbols:n Map over all available symbols
1992 \cs_new_protected:Nn \stex_all_symbols:n {
1993   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
1994   \seq_map_inline:Nn \l_stex_all_modules_seq {
1995     \seq_map_inline:cn{c_stex_module_##1_constants}{
1996       \__stex_symdecl_all_symbols_cs{##1?####1}
1997     }
1998   }
1999 }
```

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

## `\STEXsymbol`

```
2000 \NewDocumentCommand \STEXsymbol { m } {
2001   \stex_get_symbol:n { #1 }
2002   \exp_args:No
2003   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
2004 }
```

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

`symdecl` arguments:

```
2005 \keys_define:nn { stex / symdecl } {
2006   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2007   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2008   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2009   type      .tl_set:N = \l_stex_symdecl_type_tl ,
2010   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,
2011   align     .str_set:N = \l_stex_symdecl_align_str , % TODO(?)
2012   gfc       .str_set:N = \l_stex_symdecl_gfc_str , % TODO(?)
2013   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
2014   def       .tl_set:N = \l_stex_symdecl_definiens_tl ,
2015   assoc     .choices:nn =
2016             {bin,binl,binr,pre,conj,pwconj}
2017             {\str_set:Nx \l_stex_symdecl_astype_str {\l_keys_choice_tl}}
2018 }
2019
2020 \bool_new:N \l_stex_symdecl_make_macro_bool
2021
2022 \cs_new_protected:Nn \__stex_symdecl_args:n {
2023   \str_clear:N \l_stex_symdecl_name_str
2024   \str_clear:N \l_stex_symdecl_args_str
2025   \str_clear:N \l_stex_symdecl_deprecate_str
2026   \str_clear:N \l_stex_symdecl_astype_str
2027   \bool_set_false:N \l_stex_symdecl_local_bool
2028   \tl_clear:N \l_stex_symdecl_type_tl
2029   \tl_clear:N \l_stex_symdecl_definiens_tl
2030
2031   \keys_set:nn { stex / symdecl } { #1 }
2032 }
```

`\symdecl` Parses the optional arguments and passes them on to `\stex_symdecl_do:` (so that `\symdef` can do the same)

```
2033
2034 \NewDocumentCommand \symdecl { s m O{} } {
2035   \__stex_symdecl_args:n { #3 }
2036   \IfBooleanTF #1 {
2037     \bool_set_false:N \l_stex_symdecl_make_macro_bool
2038   } {
2039     \bool_set_true:N \l_stex_symdecl_make_macro_bool
2040   }
2041   \stex_symdecl_do:n { #2 }
2042   \stex_smsmode_do:
2043 }
2044
2045 \cs_new_protected:Nn \stex_symdecl_do:nn {
```



```

2046 \__stex_symdecl_args:n{#1}
2047 \bool_set_false:N \l_stex_symdecl_make_macro_bool
2048 \stex_symdecl_do:n{#2}
2049 }
2050
2051 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

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

```

2052 \cs_new_protected:Nn \stex_symdecl_do:n {
2053   \str_if_in_module:F {
2054     % TODO throw error? some default namespace?
2055   }
2056
2057   \str_if_empty:NT \l_stex_symdecl_name_str {
2058     \str_set:Nx \l_stex_symdecl_name_str { #1 }
2059   }
2060
2061   \prop_if_exist:cT { l_stex_symdecl_
2062     \l_stex_current_module_str ?
2063     \l_stex_symdecl_name_str
2064   }_prop
2065   ){
2066     % TODO throw error (beware of circular dependencies)
2067   }
2068
2069   \prop_clear:N \l_tmpa_prop
2070   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
2071   \seq_clear:N \l_tmpa_seq
2072   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
2073   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
2074
2075   \str_if_empty:NT \l_stex_symdecl_deprecate_str {
2076     \str_if_empty:NF \l_stex_module_deprecate_str {
2077       \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
2078     }
2079   }
2080   \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
2081
2082   \exp_args:No \stex_add_constant_to_current_module:n {
2083     \l_stex_symdecl_name_str
2084   }
2085
2086   % arity/args
2087   \int_zero:N \l_tmpb_int
2088
2089   \bool_set_true:N \l_tmpa_bool
2090   \str_map_inline:Nn \l_stex_symdecl_args_str {
2091     \token_case_meaning:NnF ##1 {
2092       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2093       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2094       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2095       {\tl_to_str:n a} {

```

```

2096         \bool_set_false:N \l_tmpa_bool
2097         \int_incr:N \l_tmpb_int
2098     }
2099     {\tl_to_str:n B} {
2100         \bool_set_false:N \l_tmpa_bool
2101         \int_incr:N \l_tmpb_int
2102     }
2103 }{
2104     \msg_error:nnxx{stex}{error/wrongargs}{
2105         \l_stex_current_module_str ?
2106         \l_stex_symdecl_name_str
2107     }{##1}
2108 }
2109 }
2110 \bool_if:NTF \l_tmpa_bool {
2111     % possibly numeric
2112     \str_if_empty:NTF \l_stex_symdecl_args_str {
2113         \prop_put:Nnn \l_tmpa_prop { args } {}
2114         \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2115     }{
2116         \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
2117         \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2118         \str_clear:N \l_tmpa_str
2119         \int_step_inline:nn \l_tmpa_int {
2120             \str_put_right:Nn \l_tmpa_str i
2121         }
2122         \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
2123     }
2124 } {
2125     \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2126     \prop_put:Nnx \l_tmpa_prop { arity }
2127     { \str_count:N \l_stex_symdecl_args_str }
2128 }
2129 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2130
2131 \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
2132     \prop_put:Nnx \l_tmpa_prop { defined }{ false }
2133 }{
2134     \prop_put:Nnx \l_tmpa_prop { defined }{ true }
2135 }
2136
2137 % semantic macro
2138
2139 \bool_if:NT \l_stex_symdecl_make_macro_bool {
2140     \exp_args:Nx \stex_do_up_to_module:n {
2141         \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2142             \l_stex_current_module_str ? \l_stex_symdecl_name_str
2143         }}
2144     }
2145
2146     \bool_if:NF \l_stex_symdecl_local_bool {
2147         \exp_args:Nx \stex_add_to_current_module:n {
2148             \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2149                 \l_stex_current_module_str ? \l_stex_symdecl_name_str

```

```

2150     } }
2151   }
2152 }
2153 }
2154
2155 \stex_debug:nn{symbols}{New~symbol:~
2156   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2157   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2158   Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
2159   Definiens:~\exp_not:o {\l_stex_symdecl_definiens_tl}
2160 }
2161
2162 % circular dependencies require this:
2163
2164 \prop_if_exist:cF {
2165   \l_stex_symdecl_
2166   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2167   _prop
2168 } {
2169   \exp_args:Nx \stex_do_up_to_module:n {
2170     \prop_set_from_keyval:cn {
2171       \l_stex_symdecl_
2172       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2173       _prop
2174     } {\prop_to_keyval:N \l_tmpa_prop}
2175     \seq_clear:c {
2176       \l_stex_symdecl_
2177       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2178       _notations
2179     }
2180   }
2181 }
2182
2183 \bool_if:NF \l_stex_symdecl_local_bool {
2184   \exp_args:Nx
2185   \stex_add_to_current_module:n {
2186     \seq_clear:c {
2187       \l_stex_symdecl_
2188       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2189       _notations
2190     }
2191     \prop_set_from_keyval:cn {
2192       \l_stex_symdecl_
2193       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2194       _prop
2195     } {
2196       name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2197       module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2198       type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2199       args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2200       arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2201       assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
2202       defined   = \prop_item:Nn \l_tmpa_prop { defined }   ,
2203     }

```

```

2204     }
2205 }
2206
2207 \stex_if_smsmode:F {
2208 %   \exp_args:Nx \stex_do_up_to_module:n {
2209 %     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2210 %       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2211 %     }
2212 %   }
2213 \stex_if_do_html:T {
2214   \stex_annotate_invisible:nnn {symdecl} {
2215     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2216   } {
2217     \tl_if_empty:NF \l_stex_symdecl_type_tl {
2218       \stex_annotate_invisible:nnn{type}{}{\l_stex_symdecl_type_tl$}
2219     }
2220     \stex_annotate_invisible:nnn{args}{}{
2221       \prop_item:Nn \l_tmpa_prop { args }
2222     }
2223     \stex_annotate_invisible:nnn{macroname}{#1}{}
2224     \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2225       \stex_annotate_invisible:nnn{definiens}{}
2226       {\l_stex_symdecl_definiens_tl$}
2227     }
2228     \str_if_empty:NF \l_stex_symdecl_assocotype_str {
2229       \stex_annotate_invisible:nnn{assocotype}{\l_stex_symdecl_assocotype_str}{}
2230     }
2231   }
2232 }
2233 }
2234 }

```

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

`\stex_get_symbol:n`

```

2235 \str_new:N \l_stex_get_symbol_uri_str
2236
2237 \cs_new_protected:Nn \stex_get_symbol:n {
2238   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2239     \tl_set:Nn \l_tmpa_tl { #1 }
2240     \__stex_symdecl_get_symbol_from_cs:
2241   }{
2242     % argument is a string
2243     % is it a command name?
2244     \cs_if_exist:cTF { #1 }{
2245       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2246       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2247       \str_if_empty:NNTF \l_tmpa_str {
2248         \exp_args:Nx \cs_if_eq:NNTF {
2249           \tl_head:N \l_tmpa_tl
2250         } \stex_invoke_symbol:n {
2251           \__stex_symdecl_get_symbol_from_cs:
2252         }{
2253           \__stex_symdecl_get_symbol_from_string:n { #1 }

```

```

2254     }
2255   } {
2256     \__stex_symdecl_get_symbol_from_string:n { #1 }
2257   }
2258   }{
2259     % argument is not a command name
2260     \__stex_symdecl_get_symbol_from_string:n { #1 }
2261     % \l_stex_all_symbols_seq
2262   }
2263 }
2264 \str_if_eq:eeF {
2265   \prop_item:cn {
2266     l_stex_symdecl\l_stex_get_symbol_uri_str _prop
2267   }{ deprecate }
2268 }{ }{
2269   \msg_warning:nnxx{stex}{warning/deprecated}{
2270     Symbol~\l_stex_get_symbol_uri_str
2271   }{
2272     \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{ deprecate }
2273   }
2274 }
2275 }
2276
2277 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2278   \tl_set:Nn \l_tmpa_tl {
2279     \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2280   }
2281   \str_set:Nn \l_tmpa_str { #1 }
2282   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2283
2284   \stex_all_symbols:n {
2285     \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2286       \seq_map_break:n{\seq_map_break:n{
2287         \tl_set:Nn \l_tmpa_tl {
2288           \str_set:Nn \l_stex_get_symbol_uri_str { ##1 }
2289         }
2290       }}
2291     }
2292   }
2293
2294   \l_tmpa_tl
2295 }
2296
2297 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
2298   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2299   { \tl_tail:N \l_tmpa_tl }
2300   \tl_if_single:NTF \l_tmpa_tl {
2301     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2302       \exp_after:wN \str_set:Nn \exp_after:wN
2303       \l_stex_get_symbol_uri_str \l_tmpa_tl
2304     }{
2305       % TODO
2306       % tail is not a single group
2307     }

```

```

2308 }{
2309   % TODO
2310   % tail is not a single group
2311 }
2312 }

```

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

## 29.2 Notations

```

2313 <@@=stex_notation>
      notation arguments:
2314 \keys_define:nn { stex / notation } {
2315   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2316   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2317   prec .str_set_x:N = \l__stex_notation_prec_str ,
2318   op .tl_set:N = \l__stex_notation_op_tl ,
2319   primary .bool_set:N = \l__stex_notation_primary_bool ,
2320   primary .default:n = {true} ,
2321   unknown .code:n = \str_set:Nx
2322     \l__stex_notation_variant_str \l_keys_key_str
2323 }
2324
2325 \cs_new_protected:Nn \_stex_notation_args:n {
2326   \str_clear:N \l__stex_notation_lang_str
2327   \str_clear:N \l__stex_notation_variant_str
2328   \str_clear:N \l__stex_notation_prec_str
2329   \tl_clear:N \l__stex_notation_op_tl
2330   \bool_set_false:N \l__stex_notation_primary_bool
2331
2332   \keys_set:nn { stex / notation } { #1 }
2333 }

```

**\notation**

```

2334 \NewDocumentCommand \notation { s m O{}} {
2335   \_stex_notation_args:n { #3 }
2336   \tl_clear:N \l_stex_symdecl_definiens_tl
2337   \stex_get_symbol:n { #2 }
2338   \tl_set:Nn \l_stex_notation_after_do_tl {
2339     \__stex_notation_final:
2340     \IfBooleanTF#1{
2341       \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2342     }{}
2343     \stex_smsmode_do:\ignorespacesandpars
2344   }
2345   \stex_notation_do:nnnnn
2346   { \prop_item:cn {\l_stex_symdecl\_l_stex_get_symbol_uri_str _prop } { args } }
2347   { \prop_item:cn { \l_stex_symdecl\_l_stex_get_symbol_uri_str _prop } { arity } }
2348   { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2349   { \l__stex_notation_prec_str }
2350 }
2351 \stex_deactivate_macro:Nn \notation {module~environments}

```

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

\stex\_notation\_do:nnnnn

```
2352 \seq_new:N \l__stex_notation_precedences_seq
2353 \tl_new:N \l__stex_notation_opprec_tl
2354 \int_new:N \l__stex_notation_currarg_int
2355 \tl_new:N \stex_symbol_after_invokation_tl
2356
2357 \cs_new_protected:Nn \stex_notation_do:nnnnn {
2358   \let\l_stex_current_symbol_str\relax
2359   \seq_clear:N \l__stex_notation_precedences_seq
2360   \tl_clear:N \l__stex_notation_opprec_tl
2361   \str_set:Nx \l__stex_notation_args_str { #1 }
2362   \str_set:Nx \l__stex_notation_arity_str { #2 }
2363   \str_set:Nx \l__stex_notation_suffix_str { #3 }
2364   \str_set:Nx \l__stex_notation_prec_str { #4 }
2365
2366   % precedences
2367   \str_if_empty:NTF \l__stex_notation_prec_str {
2368     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2369       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2370     }{
2371       \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2372     }
2373   } {
2374     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2375       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2376       \int_step_inline:nn { \l__stex_notation_arity_str } {
2377         \exp_args:NNo
2378         \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2379       }
2380     }{
2381       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2382       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2383         \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2384         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2385           \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2386             \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2387           \seq_map_inline:Nn \l_tmpa_seq {
2388             \seq_put_right:Nn \l_tmpb_seq { ##1 }
2389           }
2390         }
2391       }{
2392         \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2393           \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2394         }{
2395           \tl_set:No \l__stex_notation_opprec_tl { 0 }
2396         }
2397       }
2398     }
2399   }
2400
2401   \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2402   \int_step_inline:nn { \l__stex_notation_arity_str } {
2403     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpb_str {
2404       \exp_args:NNo
```

```

2405     \seq_put_right:No \l__stex_notation_precedences_seq {
2406       \l__stex_notation_opprec_tl
2407     }
2408   }
2409 }
2410 \tl_clear:N \l_stex_notation_dummyargs_tl
2411
2412 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2413   \exp_args:NNe
2414   \cs_set:Npn \l_stex_notation_macrocode_cs {
2415     \_stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2416     { \l__stex_notation_suffix_str }
2417     { \l__stex_notation_opprec_tl }
2418     { \exp_not:n { #5 } }
2419   }
2420   \l_stex_notation_after_do_tl
2421 }{
2422   \str_if_in:NnTF \l__stex_notation_args_str b {
2423     \exp_args:Nne \use:nn
2424     {
2425       \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2426       \cs_set:Npn \l__stex_notation_arity_str } { {
2427         \_stex_term_math_omb: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   }{
2433     \str_if_in:NnTF \l__stex_notation_args_str B {
2434       \exp_args:Nne \use:nn
2435       {
2436         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2437         \cs_set:Npn \l__stex_notation_arity_str } { {
2438           \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2439           { \l__stex_notation_suffix_str }
2440           { \l__stex_notation_opprec_tl }
2441           { \exp_not:n { #5 } }
2442         } }
2443     }{
2444       \exp_args:Nne \use:nn
2445       {
2446         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2447         \cs_set:Npn \l__stex_notation_arity_str } { {
2448           \_stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2449           { \l__stex_notation_suffix_str }
2450           { \l__stex_notation_opprec_tl }
2451           { \exp_not:n { #5 } }
2452         } }
2453     }
2454   }
2455
2456   \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2457   \int_zero:N \l__stex_notation_currarg_int
2458   \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq

```



```

2459     \__stex_notation_arguments:
2460   }
2461 }

```

(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

```

2462 \cs_new_protected:Nn \__stex_notation_arguments: {
2463   \int_incr:N \l__stex_notation_currarg_int
2464   \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2465     \l_stex_notation_after_do_tl
2466   }{
2467     \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2468     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2469     \str_if_eq:NnTF \l_tmpa_str a {
2470       \__stex_notation_argument_assoc:nn{a}
2471     }{
2472       \str_if_eq:NnTF \l_tmpa_str B {
2473         \__stex_notation_argument_assoc:nn{B}
2474       }{
2475         \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpb_str
2476         \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2477           { \_stex_term_math_arg:nnn
2478             { \l_tmpa_str\int_use:N \l__stex_notation_currarg_int }
2479             { \l_tmpb_str }
2480             { ###\int_use:N \l__stex_notation_currarg_int }
2481           }
2482         }
2483         \__stex_notation_arguments:
2484       }
2485     }
2486   }
2487 }

```

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

\\_stex\_notation\_argument\_assoc:nn

```

2488 \cs_new_protected:Nn \_stex_notation_argument_assoc:nn {
2489
2490   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2491     {\l__stex_notation_arity_str}{
2492       #2
2493     }
2494   \int_zero:N \l_tmpa_int
2495   \tl_clear:N \l_tmpa_tl
2496   \str_map_inline:Nn \l__stex_notation_args_str {
2497     \int_incr:N \l_tmpa_int
2498     \tl_put_right:Nx \l_tmpa_tl {
2499       \str_if_eq:nnTF {##1}{a}{ {} }{
2500         \str_if_eq:nnTF {##1}{B}{ {} }{
2501           {\_stex_term_arg:nn{##1}\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa
2502         }
2503       }
2504     }

```

```

2505 }
2506 \exp_after:wN\exp_after:wN\exp_after:wN \def
2507 \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2508 \exp_after:wN\exp_after:wN\exp_after:wN ##
2509 \exp_after:wN\exp_after:wN\exp_after:wN 1
2510 \exp_after:wN\exp_after:wN\exp_after:wN ##
2511 \exp_after:wN\exp_after:wN\exp_after:wN 2
2512 \exp_after:wN\exp_after:wN\exp_after:wN {
2513   \exp_after:wN \exp_after:wN \exp_after:wN
2514   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2515     \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2516   }
2517 }
2518
2519 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2520 \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
2521   \stex_term_math_assoc_arg:nnnn
2522   { #1\int_use:N \l__stex_notation_currarg_int }
2523   { \l_tmpa_str }
2524   { ####\int_use:N \l__stex_notation_currarg_int }
2525   { \l_tmpa_cs {####1} {####2} }
2526 } }
2527 \__stex_notation_arguments:
2528 }

```

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

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

```

2529 \cs_new_protected:Nn \__stex_notation_final: {
2530 % \exp_args:Nne \use:nn
2531 % {
2532 % \cs_generate_from_arg_count:cNnn {
2533 %   stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2534 %   \l__stex_notation_suffix_str
2535 %   _cs
2536 % }
2537 % \cs_set:Npn \l__stex_notation_arity_str } { {
2538 %   \exp_after:wN \exp_after:wN \exp_after:wN
2539 %   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2540 %   { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sym
2541 % } }
2542
2543 % \tl_if_empty:NF \l__stex_notation_op_tl {
2544 %   \cs_set:cpx {
2545 %     stex_op_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2546 %     \l__stex_notation_suffix_str
2547 %     _cs
2548 %   } { \exp_not:N \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2549 % }
2550
2551 \exp_args:Nx \stex_do_up_to_module:n {
2552   \cs_generate_from_arg_count:cNnn {
2553     stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2554     \l__stex_notation_suffix_str

```

```

2555     _cs
2556   } \cs_set:Npn {\l__stex_notation_arity_str} {
2557     \exp_after:wN \exp_after:wN \exp_after:wN
2558     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2559     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2560   }
2561   \tl_if_empty:NF \l__stex_notation_op_tl {
2562     \cs_set:cpn {
2563       stex_op_notation_\l_stex_get_symbol_uri_str \c_hash_str
2564       \l__stex_notation_suffix_str
2565       _cs
2566     } { \exp_not:N \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2567   }
2568 }
2569
2570 \exp_args:Ne
2571 \stex_add_to_current_module:n {
2572   \cs_generate_from_arg_count:cNnn {
2573     stex_notation_\l_stex_get_symbol_uri_str \c_hash_str
2574     \l__stex_notation_suffix_str
2575     _cs
2576   } \cs_set:Npn {\l__stex_notation_arity_str} {
2577     \exp_after:wN \exp_after:wN \exp_after:wN
2578     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2579     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2580   }
2581   \tl_if_empty:NF \l__stex_notation_op_tl {
2582     \cs_set:cpn {
2583       stex_op_notation_\l_stex_get_symbol_uri_str \c_hash_str
2584       \l__stex_notation_suffix_str
2585       _cs
2586     } { \exp_not:N \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2587   }
2588 }
2589
2590 \stex_debug:nn{symbols}{
2591   Notation~\l__stex_notation_suffix_str
2592   ~for~\l_stex_get_symbol_uri_str^^J
2593   Operator~precedence:~\l__stex_notation_opprec_tl^^J
2594   Argument~precedences:~
2595     \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2596   Notation: \cs_meaning:c {
2597     stex_notation_\l_stex_get_symbol_uri_str \c_hash_str
2598     \l__stex_notation_suffix_str
2599     _cs
2600   }
2601 }
2602
2603 \exp_args:Ne
2604 \stex_do_up_to_module:n {
2605   \exp_not:N \seq_if_exist:cT { l_stex_symdecl_\l_stex_get_symbol_uri_str _notations }{
2606     \seq_put_right:cn {
2607       l_stex_symdecl_\l_stex_get_symbol_uri_str
2608       _notations

```

```

2609     } {\l__stex_notation_suffix_str}
2610   }
2611 }
2612 \exp_args:Ne
2613 \stex_add_to_current_module:n {
2614   \seq_put_right:cn {
2615     l_stex_symdecl_l\l_stex_get_symbol_uri_str
2616     _notations
2617   } { \l__stex_notation_suffix_str }
2618 }
2619
2620 \stex_if_smsmode:F {
2621
2622   % HTML annotations
2623   \stex_if_do_html:T {
2624     \stex_annotate_invisible:nnn { notation }
2625     { \l_stex_get_symbol_uri_str } {
2626       \stex_annotate_invisible:nnn { notationfragment }
2627       { \l__stex_notation_suffix_str }{}
2628       \stex_annotate_invisible:nnn { precedence }
2629       { \l__stex_notation_prec_str }{}
2630
2631       \int_zero:N \l_tmpa_int
2632       \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2633       \tl_clear:N \l_tmpa_tl
2634       \int_step_inline:nn { \l__stex_notation_arity_str }{
2635         \int_incr:N \l_tmpa_int
2636         \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2637         \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_r
2638         \str_if_eq:VnTF \l_tmpb_str a {
2639           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2640             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2641             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2642           } }
2643         }{
2644           \str_if_eq:VnTF \l_tmpb_str B {
2645             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2646               \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2647               \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2648             } }
2649           }{
2650             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2651               \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
2652             } }
2653           }
2654         }
2655       }
2656       \stex_annotate_invisible:nnn { notationcomp }{}{
2657         \str_set:Nx \l_stex_current_symbol_str {\l_stex_get_symbol_uri_str }
2658         $ \exp_args:Nno \use:nn { \use:c {
2659           stex_notation_ \l_stex_current_symbol_str
2660           \c_hash_str \l__stex_notation_suffix_str _cs
2661         } } { \l_tmpa_tl } $
2662       }

```

```

2663     }
2664   }
2665 }
2666 }

```

(End definition for `\_stex_notation_final:`.)

## `\setnotation`

```

2667 \keys_define:nn { stex / setnotation } {
2668   lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2669   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2670   unknown  .code:n      = \str_set:Nx
2671             \l__stex_notation_variant_str \l_keys_key_str
2672 }
2673
2674 \cs_new_protected:Nn \stex_setnotation_args:n {
2675   \str_clear:N \l__stex_notation_lang_str
2676   \str_clear:N \l__stex_notation_variant_str
2677   \keys_set:nn { stex / setnotation } { #1 }
2678 }
2679
2680 \cs_new_protected:Nn \stex_setnotation:n {
2681   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1 _notations }
2682     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2683     \exp_args:Nx \stex_do_up_to_module:n {
2684       \exp_not:N \seq_if_exist:cT{l_stex_symdecl_#1_notations}{
2685         \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2686         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2687         \seq_put_left:cn { l_stex_symdecl_#1 _notations }
2688         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2689       }
2690     }
2691     \exp_args:Nx \stex_add_to_current_module:n {
2692       \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2693       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2694       \seq_put_left:cn { l_stex_symdecl_#1 _notations }
2695       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2696     }
2697     \stex_debug:nn {notations}{
2698       Setting~default~notation~
2699       {\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str}~for~
2700       #1 \\
2701       \expandafter\meaning\csname
2702       l_stex_symdecl_#1 _notations\endcsname
2703     }
2704   }{
2705     \msg_error:nxxx{stex}{unknownnotation}{\l__stex_notation_variant_str \c_hash_str \l__s
2706   }
2707 }
2708
2709 \NewDocumentCommand \setnotation {m m} {
2710   \stex_get_symbol:n { #1 }
2711   \stex_setnotation_args:n { #2 }
2712   \stex_setnotation:n{\l_stex_get_symbol_uri_str}

```

```

2713 \stex_smsmode_do:\ignorespacesandpars
2714 }
2715
2716 \cs_new_protected:Nn \stex_copy_notations:nn {
2717   \stex_debug:nn {notations}{
2718     Copying~notations~from~#2~to~#1\\
2719     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
2720   }
2721   \tl_clear:N \l_tmpa_tl
2722   \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2723     \tl_put_right:Nn \l_tmpa_tl { {##} ##1} }
2724   }
2725   \seq_map_inline:cn {l_stex_symdecl_#2_notations}{
2726     \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2727     \edef \l_tmpa_tl {
2728       \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2729       \exp_after:wN\exp_after:wN\exp_after:wN {
2730         \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2731       }
2732     }
2733     \exp_args:Nx
2734     \stex_add_to_current_module:n {
2735       \exp_not:N \seq_if_exist:cT{l_stex_symdecl_#1_notations}{
2736         \seq_put_right:cn{l_stex_symdecl_#1_notations}{##1}
2737         \cs_generate_from_arg_count:cNnn {
2738           stex_notation_ #1 \c_hash_str ##1 _cs
2739         } \cs_set:Npn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }{
2740           \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl}
2741         }
2742         \cs_if_exist:cT{stex_op_notation_ #2\c_hash_str ##1 _cs}{
2743           \tl_set:cn{stex_op_notation_ #1\c_hash_str ##1 _cs}
2744             {\exp_args:NNo\exp_args:No\exp_not:n{\cename stex_op_notation_ #2\c_hash_str ##1
2745             }
2746           }
2747         }
2748         \exp_args:Nx
2749         \stex_do_up_to_module:n {
2750           \exp_not:N \seq_if_exist:cT{l_stex_symdecl_#1_notations}{
2751             \seq_put_right:cn{l_stex_symdecl_#1_notations}{##1}
2752             \cs_generate_from_arg_count:cNnn {
2753               stex_notation_ #1 \c_hash_str ##1 _cs
2754             } \cs_set:Npn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }{
2755               \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl}
2756             }
2757             \cs_if_exist:cT{stex_op_notation_ #2\c_hash_str ##1 _cs}{
2758               \tl_set:cn{stex_op_notation_ #1\c_hash_str ##1 _cs}
2759                 {\exp_args:NNo\exp_args:No\exp_not:n{\cename stex_op_notation_ #2\c_hash_str ##1
2760                 }
2761               }
2762             }
2763           }
2764         }
2765       }
2766       \NewDocumentCommand \copynotation {m m} {

```

```

2767 \stex_get_symbol:n { #1 }
2768 \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2769 \stex_get_symbol:n { #2 }
2770 \exp_args:Noo
2771 \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2772 \exp_args:Nx \stex_add_to_current_module:n{
2773   \stex_copy_notations:nn {\l_tmpa_str} {\l_stex_get_symbol_uri_str}
2774 }
2775 \stex_smsmode_do:\ignorespacesandpars
2776 }
2777

```

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

## \symdef

```

2778 \keys_define:nn { stex / symdef } {
2779   name .str_set_x:N = \l_stex_symdecl_name_str ,
2780   local .bool_set:N = \l_stex_symdecl_local_bool ,
2781   args .str_set_x:N = \l_stex_symdecl_args_str ,
2782   type .tl_set:N = \l_stex_symdecl_type_tl ,
2783   def .tl_set:N = \l_stex_symdecl_definiens_tl ,
2784   op .tl_set:N = \l__stex_notation_op_tl ,
2785   lang .str_set_x:N = \l__stex_notation_lang_str ,
2786   variant .str_set_x:N = \l__stex_notation_variant_str ,
2787   prec .str_set_x:N = \l__stex_notation_prec_str ,
2788   assoc .choices:nn =
2789     {bin,binl,binr,pre,conj,pwconj}
2790     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
2791   unknown .code:n = \str_set:Nx
2792     \l__stex_notation_variant_str \l_keys_key_str
2793 }
2794
2795 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2796   \str_clear:N \l_stex_symdecl_name_str
2797   \str_clear:N \l_stex_symdecl_args_str
2798   \str_clear:N \l_stex_symdecl_assoctype_str
2799   \bool_set_false:N \l_stex_symdecl_local_bool
2800   \tl_clear:N \l_stex_symdecl_type_tl
2801   \tl_clear:N \l_stex_symdecl_definiens_tl
2802   \str_clear:N \l__stex_notation_lang_str
2803   \str_clear:N \l__stex_notation_variant_str
2804   \str_clear:N \l__stex_notation_prec_str
2805   \tl_clear:N \l__stex_notation_op_tl
2806
2807   \keys_set:nn { stex / symdef } { #1 }
2808 }
2809
2810 \NewDocumentCommand \symdef { m O{} } {
2811   \__stex_notation_symdef_args:n { #2 }
2812   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2813   \stex_symdecl_do:n { #1 }
2814   \tl_set:Nn \l_stex_notation_after_do_tl {
2815     \__stex_notation_final:
2816     \stex_smsmode_do:\ignorespacesandpars

```

```

2817 }
2818 \str_set:Nx \l_stex_get_symbol_uri_str {
2819   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2820 }
2821 \exp_args:Nx \stex_notation_do:nnnnn
2822   { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } }
2823   { \prop_item:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } }
2824   { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2825   { \l__stex_notation_prec_str }
2826 }
2827 \stex_deactivate_macro:Nn \symdef {module~environments}

```

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

## 29.3 Variables

```

2828 <@@=stex_variables>
2829
2830 \keys_define:nn { stex / vardef } {
2831   name .str_set_x:N = \l__stex_variables_name_str ,
2832   args .str_set_x:N = \l__stex_variables_args_str ,
2833   type .tl_set:N    = \l__stex_variables_type_tl ,
2834   def  .tl_set:N    = \l__stex_variables_def_tl ,
2835   op   .tl_set:N    = \l__stex_variables_op_tl ,
2836   prec .str_set_x:N = \l__stex_variables_prec_str ,
2837   assoc .choices:nn =
2838     {bin,binl,binr,pre,conj,pwconj}
2839     {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
2840   bind .choices:nn =
2841     {forall,exists}
2842     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2843 }
2844
2845 \cs_new_protected:Nn \__stex_variables_args:n {
2846   \str_clear:N \l__stex_variables_name_str
2847   \str_clear:N \l__stex_variables_args_str
2848   \str_clear:N \l__stex_variables_prec_str
2849   \str_clear:N \l__stex_variables_assoctype_str
2850   \str_clear:N \l__stex_variables_bind_str
2851   \tl_clear:N \l__stex_variables_type_tl
2852   \tl_clear:N \l__stex_variables_def_tl
2853   \tl_clear:N \l__stex_variables_op_tl
2854
2855   \keys_set:nn { stex / vardef } { #1 }
2856 }
2857
2858 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{}} {
2859   \__stex_variables_args:n {#2}
2860   \str_if_empty:NT \l__stex_variables_name_str {
2861     \str_set:Nx \l__stex_variables_name_str { #1 }
2862   }
2863   \prop_clear:N \l_tmpa_prop
2864   \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
2865

```



```

2866 \int_zero:N \l_tmpb_int
2867 \bool_set_true:N \l_tmpa_bool
2868 \str_map_inline:Nn \l__stex_variables_args_str {
2869   \token_case_meaning:NnF ##1 {
2870     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2871     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2872     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2873     {\tl_to_str:n a} {
2874       \bool_set_false:N \l_tmpa_bool
2875       \int_incr:N \l_tmpb_int
2876     }
2877     {\tl_to_str:n B} {
2878       \bool_set_false:N \l_tmpa_bool
2879       \int_incr:N \l_tmpb_int
2880     }
2881   }{
2882     \msg_error:nxxx{stex}{error/wrongargs}{
2883       variable~\l__stex_variables_name_str
2884     }{##1}
2885   }
2886 }
2887 \bool_if:NTF \l_tmpa_bool {
2888   % possibly numeric
2889   \str_if_empty:NTF \l__stex_variables_args_str {
2890     \prop_put:Nnn \l_tmpa_prop { args } {}
2891     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2892   }{
2893     \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
2894     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2895     \str_clear:N \l_tmpa_str
2896     \int_step_inline:nn \l_tmpa_int {
2897       \str_put_right:Nn \l_tmpa_str i
2898     }
2899     \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
2900     \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2901   }
2902 } {
2903   \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2904   \prop_put:Nnx \l_tmpa_prop { arity }
2905     { \str_count:N \l__stex_variables_args_str }
2906 }
2907 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2908 \tl_set:cx { #1 }{ \stex_invoke_variable:n { \l__stex_variables_name_str } }
2909
2910 \prop_set_eq:cN { l_stex_variable_\l__stex_variables_name_str _prop} \l_tmpa_prop
2911
2912 \tl_if_empty:NF \l__stex_variables_op_tl {
2913   \cs_set:cpx {
2914     stex_var_op_notation_ \l__stex_variables_name_str _cs
2915   } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
2916 }
2917
2918 \tl_set:Nn \l_stex_notation_after_do_tl {
2919   \exp_args:Nne \use:nn {

```

```

2920 \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str _cs }
2921 \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
2922 } {{
2923 \exp_after:wN \exp_after:wN \exp_after:wN
2924 \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2925 { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_symbol
2926 }}
2927 \stex_if_do_html:T {
2928 \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
2929 \stex_annotate_invisible:nnn { precedence }
2930 { \l__stex_variables_prec_str }}{
2931 \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{\l__stex_variables_type_str}{
2932 \stex_annotate_invisible:nnn{args}{\l__stex_variables_args_str }
2933 \stex_annotate_invisible:nnn{macroname}{#1}{
2934 \tl_if_empty:NF \l__stex_variables_def_tl {
2935 \stex_annotate_invisible:nnn{definiens}{
2936 {\l__stex_variables_def_tl$}
2937 }
2938 \str_if_empty:NF \l__stex_variables_assoctype_str {
2939 \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{
2940 }
2941 \str_if_empty:NF \l__stex_variables_bind_str {
2942 \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{
2943 }
2944 \int_zero:N \l_tmpa_int
2945 \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
2946 \tl_clear:N \l_tmpa_tl
2947 \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{{
2948 \int_incr:N \l_tmpa_int
2949 \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
2950 \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables_remaining_args_str }
2951 \str_if_eq:VnTF \l_tmpb_str a {
2952 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2953 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{
2954 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{
2955 } }
2956 }}{
2957 \str_if_eq:VnTF \l_tmpb_str B {
2958 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2959 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{
2960 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{
2961 } }
2962 }}{
2963 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2964 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{
2965 } }
2966 }
2967 }
2968 }
2969 \stex_annotate_invisible:nnn { notationcomp }{{
2970 \str_set:Nx \l_stex_current_symbol_str {var://\l__stex_variables_name_str }
2971 $ \exp_args:Nno \use:nn { \use:c {
2972 stex_var_notation_\l__stex_variables_name_str _cs
2973 } } { \l_tmpa_tl } $

```

```

2974     }
2975   }
2976   }\ignorespacesandpars
2977 }
2978
2979 \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
2980 }
2981
2982 \cs_new:Nn \_stex_reset:N {
2983   \tl_if_exist:NTF #1 {
2984     \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
2985   }{
2986     \let \exp_not:N #1 \exp_not:N \undefined
2987   }
2988 }
2989
2990 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
2991   \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
2992   \exp_args:Nnx \use:nn {
2993     % TODO
2994     \stex_annotate_invisible:nnn {vardecl}{\clist_use:Nn\l__stex_variables_names,}{
2995       #2
2996     }
2997   }{
2998     \_stex_reset:N \varnot
2999     \_stex_reset:N \vartype
3000     \_stex_reset:N \vardefi
3001   }
3002 }
3003
3004 \NewDocumentCommand \vardef { s } {
3005   \IfBooleanTF#1 {
3006     \__stex_variables_do_complex:nn
3007   }{
3008     \__stex_variables_do_simple:nnn
3009   }
3010 }
3011
3012 \NewDocumentCommand \svar { 0{} m }{
3013   \tl_if_empty:nTF {#1}{
3014     \str_set:Nn \l_tmpa_str { #2 }
3015   }{
3016     \str_set:Nn \l_tmpa_str { #1 }
3017   }
3018   \_stex_term_omv:nn {
3019     var://\l_tmpa_str
3020   }{
3021     \exp_args:Nnx \use:nn {
3022       \def\comp{\_varcomp}
3023       \str_set:Nx \l_stex_current_symbol_str { var://\l_tmpa_str }
3024       \comp{ #2 }
3025     }{
3026       \_stex_reset:N \comp
3027       \_stex_reset:N \l_stex_current_symbol_str

```

```

3028     }
3029   }
3030 }
3031
3032
3033
3034 \keys_define:nn { stex / varseq } {
3035   name      .str_set_x:N = \l__stex_variables_name_str ,
3036   args      .int_set:N   = \l__stex_variables_args_int ,
3037   type      .tl_set:N    = \l__stex_variables_type_tl ,
3038   mid       .tl_set:N    = \l__stex_variables_mid_tl ,
3039   bind      .choices:nn =
3040     {forall,exists}
3041     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
3042 }
3043
3044 \cs_new_protected:Nn \__stex_variables_seq_args:n {
3045   \str_clear:N \l__stex_variables_name_str
3046   \int_set:Nn \l__stex_variables_args_int 1
3047   \tl_clear:N \l__stex_variables_type_tl
3048   \str_clear:N \l__stex_variables_bind_str
3049
3050   \keys_set:nn { stex / varseq } { #1 }
3051 }
3052
3053 \NewDocumentCommand \varseq {m O{}} m m m){
3054   \__stex_variables_seq_args:n { #2 }
3055   \str_if_empty:NT \l__stex_variables_name_str {
3056     \str_set:Nx \l__stex_variables_name_str { #1 }
3057   }
3058   \prop_clear:N \l_tmpa_prop
3059   \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
3060
3061   \seq_set_from_clist:Nn \l_tmpa_seq {#3}
3062   \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
3063     \msg_error:nnxx{stex}{error/seqlength}
3064     {\int_use:N \l__stex_variables_args_int}
3065     {\seq_count:N \l_tmpa_seq}
3066   }
3067   \seq_set_from_clist:Nn \l_tmpb_seq {#4}
3068   \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
3069     \msg_error:nnxx{stex}{error/seqlength}
3070     {\int_use:N \l__stex_variables_args_int}
3071     {\seq_count:N \l_tmpb_seq}
3072   }
3073   \prop_put:Nnn \l_tmpa_prop {starts} {#3}
3074   \prop_put:Nnn \l_tmpa_prop {ends} {#4}
3075
3076   \cs_generate_from_arg_count:cNnn {stex_varseq\l__stex_variables_name_str _cs}
3077   \cs_set:Npn {\int_use:N \l__stex_variables_args_int} { #5 }
3078
3079   \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}
3080   \int_step_inline:nn \l__stex_variables_args_int {
3081     \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }

```

```

3082 }
3083 \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
3084 \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
3085 \tl_if_empty:NF \l__stex_variables_mid_tl {
3086   \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
3087   \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
3088 }
3089 \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq\l__stex_variables_name_str_cs}}
3090 \int_step_inline:nn \l__stex_variables_args_int {
3091   \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
3092 }
3093 \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
3094 \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
3095
3096
3097 \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl
3098
3099 \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}
3100
3101 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq\l__stex_variables_name_str_cs}}
3102
3103 \int_step_inline:nn \l__stex_variables_args_int {
3104   \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
3105     \stex_term_math_arg:nnn{i##1}{0}{\exp_not:n{####}##1}
3106   }}
3107 }
3108
3109 \tl_set:Nx \l_tmpa_tl {
3110   \stex_term_math_oma:nnnn { varseq://\l__stex_variables_name_str}{0}{
3111     \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
3112   }
3113 }
3114
3115 \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \stex_symbol_after_invokation_tl} }
3116
3117 \exp_args:Nno \use:nn {
3118   \cs_generate_from_arg_count:cNnn {stex_varseq\l__stex_variables_name_str_cs}
3119   \cs_set:Npn {\int_use:N \l__stex_variables_args_int}{\l_tmpa_tl}
3120
3121   \stex_debug:nn{sequences}{New~Sequence:~
3122     \expandafter\meaning\csname stex_varseq\l__stex_variables_name_str_cs\endcsname\\~\\
3123     \prop_to_keyval:N \l_tmpa_prop
3124   }
3125   \stex_if_do_html:T{\stex_annotate_invisible:nnn{varseq}{\l__stex_variables_name_str}{
3126     \tl_if_empty:NF \l__stex_variables_type_tl {
3127       \stex_annotate:nnn {type}{}{\seqtype\l__stex_variables_type_tl$}
3128     }
3129     \stex_annotate:nnn {args}{\int_use:N \l__stex_variables_args_int}{}
3130     \str_if_empty:NF \l__stex_variables_bind_str {
3131       \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{}
3132     }
3133   }}
3134
3135   \prop_set_eq:cN {stex_varseq\l__stex_variables_name_str_prop}\l_tmpa_prop

```

```
3136 \ignorespacesandpars
3137 }
3138
3139 </package>
```

## Chapter 30

# STEX -Terms Implementation

```
3140 <*package>
3141
3142 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
3143
3144 <@@=stex_terms>
3145
3146   Warnings and error messages
3147 \msg_new:nnn{stex}{error/nonotation}{
3148   Symbol~#1~invoked,~but~has~no~notation~#2!
3149 }
3150 \msg_new:nnn{stex}{error/notationarg}{
3151   Error~in~parsing~notation~#1
3152 }
3153 \msg_new:nnn{stex}{error/noop}{
3154   Symbol~#1~has~no~operator~notation~for~notation~#2
3155 }
3156 \msg_new:nnn{stex}{error/notallowed}{
3157   Symbol~invocation~#1~not~allowed~in~notation~component~of~#2
3158 }
3159 \msg_new:nnn{stex}{error/doubleargument}{
3160   Argument~#1~of~symbol~#2~already~assigned
3161 }
3162 \msg_new:nnn{stex}{error/overarity}{
3163   Argument~#1~invalid~for~symbol~#2~with~arity~#3
3164 }
```

### 30.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
3164
3165
3166 \bool_new:N \l_stex_allow_semantic_bool
3167 \bool_set_true:N \l_stex_allow_semantic_bool
3168
```

```

3169 \cs_new_protected:Nn \stex_invoke_symbol:n {
3170   \bool_if:NTF \l_stex_allow_semantic_bool {
3171     \str_if_eq:eeF {
3172       \prop_item:cn {
3173         l_stex_symdecl_#1_prop
3174       }{ deprecate }
3175     }{}{
3176       \msg_warning:nxxx{stex}{warning/deprecated}{
3177         Symbol~#1
3178       }{
3179         \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
3180       }
3181     }
3182     \if_mode_math:
3183       \exp_after:wN \__stex_terms_invoke_math:n
3184     \else:
3185       \exp_after:wN \__stex_terms_invoke_text:n
3186     \fi: { #1 }
3187   }{
3188     \msg_error:nxxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
3189   }
3190 }
3191
3192 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
3193   \peek_charcode_remove:NTF ! {
3194     \__stex_terms_invoke_op_custom:nn {#1}
3195   }{
3196     \__stex_terms_invoke_custom:nn {#1}
3197   }
3198 }
3199
3200 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
3201   \peek_charcode_remove:NTF ! {
3202     % operator
3203     \peek_charcode_remove:NTF * {
3204       % custom op
3205       \__stex_terms_invoke_op_custom:nn {#1}
3206     }{
3207       % op notation
3208       \peek_charcode:NTF [ {
3209         \__stex_terms_invoke_op_notation:nw {#1}
3210       }{
3211         \__stex_terms_invoke_op_notation:nw {#1}[]
3212       }
3213     }
3214   }{
3215     \peek_charcode_remove:NTF * {
3216       \__stex_terms_invoke_custom:nn {#1}
3217       % custom
3218     }{
3219       % normal
3220       \peek_charcode:NTF [ {
3221         \__stex_terms_invoke_notation:nw {#1}
3222       }{

```



```

3223     \__stex_terms_invoke_notation:nw {#1}[]
3224   }
3225 }
3226 }
3227 }
3228
3229
3230 \cs_new_protected:Nn \__stex_terms_invoke_op_custom:nn {
3231   \exp_args:Nnx \use:nn {
3232     \def\comp{\_comp}
3233     \str_set:Nn \l_stex_current_symbol_str { #1 }
3234     \bool_set_false:N \l_stex_allow_semantic_bool
3235     \stex_term_oms:nnn {#1}{#1 \c_hash_str CUSTOM-}{
3236       \comp{ #2 }
3237     }
3238   }{
3239     \stex_reset:N \comp
3240     \stex_reset:N \l_stex_current_symbol_str
3241     \bool_set_true:N \l_stex_allow_semantic_bool
3242   }
3243 }
3244
3245 \keys_define:nn { stex / terms } {
3246   lang .tl_set_x:N = \l_stex_notation_lang_str ,
3247   variant .tl_set_x:N = \l_stex_notation_variant_str ,
3248   unknown .code:n = \str_set:Nx
3249     \l_stex_notation_variant_str \l_keys_key_str
3250 }
3251
3252 \cs_new_protected:Nn \__stex_terms_args:n {
3253   \str_clear:N \l_stex_notation_lang_str
3254   \str_clear:N \l_stex_notation_variant_str
3255
3256   \keys_set:nn { stex / terms } { #1 }
3257 }
3258
3259 \cs_new_protected:Nn \stex_find_notation:nn {
3260   \__stex_terms_args:n { #2 }
3261   \seq_if_empty:cTF {
3262     l_stex_symdecl_ #1 _notations
3263   } {
3264     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
3265   } {
3266     \bool_lazy_all:nTF {
3267       {\str_if_empty_p:N \l_stex_notation_variant_str}
3268       {\str_if_empty_p:N \l_stex_notation_lang_str}
3269     }{
3270       \seq_get_left:cN {l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
3271     }{
3272       \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
3273         \l_stex_notation_variant_str \c_hash_str \l_stex_notation_lang_str
3274       }{
3275         \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3276       }{

```

```

3277         \msg_error:nxxx{stex}{error/nonotation}{#1}{
3278             ~\l_stex_notation_variant_str \c_hash_str \l_stex_notation_lang_str
3279         }
3280     }
3281 }
3282 }
3283 }
3284
3285 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3286     \exp_args:Nnx \use:nn {
3287         \def\comp{\_comp}
3288         \str_set:Nn \l_stex_current_symbol_str { #1 }
3289         \stex_find_notation:nn { #1 }{ #2 }
3290         \bool_set_false:N \l_stex_allow_semantic_bool
3291         \cs_if_exist:cTF {
3292             stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3293         }{
3294             \_stex_term_oms:nnn { #1 }{
3295                 #1 \c_hash_str \l_stex_notation_variant_str
3296             }{
3297                 \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3298             }
3299         }{
3300             \int_compare:nNnTF {\prop_item:cn {\l_stex_symdecl_#1_prop}{arity}} = 0{
3301                 \cs_if_exist:cTF {
3302                     stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3303                 }{
3304                     \tl_set:Nx \stex_symbol_after_invokation_tl {
3305                         \_stex_reset:N \comp
3306                         \_stex_reset:N \stex_symbol_after_invokation_tl
3307                         \_stex_reset:N \l_stex_current_symbol_str
3308                         \bool_set_true:N \l_stex_allow_semantic_bool
3309                     }
3310                     \def\comp{\_comp}
3311                     \str_set:Nn \l_stex_current_symbol_str { #1 }
3312                     \bool_set_false:N \l_stex_allow_semantic_bool
3313                     \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3314                 }{
3315                     \msg_error:nxxx{stex}{error/nonotation}{#1}{
3316                         ~\l_stex_notation_variant_str
3317                     }
3318                 }
3319             }{
3320                 \msg_error:nxxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3321             }
3322         }
3323     }{
3324         \_stex_reset:N \comp
3325         \_stex_reset:N \l_stex_current_symbol_str
3326         \bool_set_true:N \l_stex_allow_semantic_bool
3327     }
3328 }
3329
3330 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {

```

```

3331 \stex_find_notation:nn { #1 }{ #2 }
3332 \cs_if_exist:cTF {
3333   stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3334 }{
3335   \tl_set:Nx \stex_symbol_after_invokation_tl {
3336     \_stex_reset:N \comp
3337     \_stex_reset:N \stex_symbol_after_invokation_tl
3338     \_stex_reset:N \l_stex_current_symbol_str
3339     \bool_set_true:N \l_stex_allow_semantic_bool
3340   }
3341   \def\comp{\_comp}
3342   \str_set:Nn \l_stex_current_symbol_str { #1 }
3343   \bool_set_false:N \l_stex_allow_semantic_bool
3344   \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3345 }{
3346   \msg_error:nnxx{stex}{error/nonotation}{#1}{
3347     ~\l_stex_notation_variant_str
3348   }
3349 }
3350 }
3351
3352 \prop_new:N \l__stex_terms_custom_args_prop
3353
3354 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3355   \exp_args:Nnx \use:nn {
3356     \bool_set_false:N \l_stex_allow_semantic_bool
3357     \def\comp{\_comp}
3358     \str_set:Nn \l_stex_current_symbol_str { #1 }
3359     \prop_clear:N \l__stex_terms_custom_args_prop
3360     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3361     \prop_get:cnN {
3362       l_stex_symdecl_#1 _prop
3363     }{ args } \l_tmpa_str
3364     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3365     \tl_set:Nn \arg { \__stex_terms_arg: }
3366     \str_if_empty:NTF \l_tmpa_str {
3367       \_stex_term_oms:nnn {#1}{#1\c_hash_str CUSTOM-}{#2}
3368     }{
3369       \str_if_in:NnTF \l_tmpa_str b {
3370         \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3371       }{
3372         \str_if_in:NnTF \l_tmpa_str B {
3373           \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3374         }{
3375           \_stex_term_oma:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3376         }
3377       }
3378     }
3379     % TODO check that all arguments exist
3380   }{
3381     \_stex_reset:N \l_stex_current_symbol_str
3382     \_stex_reset:N \arg
3383     \_stex_reset:N \comp
3384     \_stex_reset:N \l__stex_terms_custom_args_prop

```

```

3385     \bool_set_true:N \l_stex_allow_semantic_bool
3386   }
3387 }
3388
3389 \NewDocumentCommand \__stex_terms_arg: { s O{} m}{
3390   \tl_if_empty:nTF {#2}{
3391     \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
3392     \bool_set_true:N \l_tmpa_bool
3393     \bool_do_while:Nn \l_tmpa_bool {
3394       \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int}
3395       \int_incr:N \l_tmpa_int
3396     }{
3397       \bool_set_false:N \l_tmpa_bool
3398     }
3399   }
3400   ){
3401     \int_set:Nn \l_tmpa_int { #2 }
3402   }
3403   \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
3404   \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3405     \msg_error:nnxxx{stex}{error/overarity}
3406     {\int_use:N \l_tmpa_int}
3407     {\l_stex_current_symbol_str}
3408     {\str_count:N \l_tmpa_str}
3409   }
3410   \str_set:Nx \l_tmpa_str {\str_item:Nn \l_tmpa_str \l_tmpa_int}
3411   \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
3412     \bool_lazy_any:nF {
3413       {\str_if_eq_p:Vn \l_tmpa_str {a}}
3414       {\str_if_eq_p:Vn \l_tmpa_str {B}}
3415     }{
3416       \msg_error:nnxx{stex}{error/doubleargument}
3417       {\int_use:N \l_tmpa_int}
3418       {\l_stex_current_symbol_str}
3419     }
3420   }
3421   \exp_args:NNx \prop_put:Nnn \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {#3}
3422   \bool_set_true:N \l_stex_allow_semantic_bool
3423   \IfBooleanTF#1{
3424     \stex_annotate_invisible:n { %TODO
3425       \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{#3}
3426     }
3427   }{ %TODO
3428     \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{#3}
3429   }
3430   \bool_set_false:N \l_stex_allow_semantic_bool
3431 }
3432
3433
3434 \cs_new_protected:Nn \_stex_term_arg:nn {
3435   \bool_set_true:N \l_stex_allow_semantic_bool
3436   \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3437   \bool_set_false:N \l_stex_allow_semantic_bool
3438 }

```

```

3439
3440 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
3441   \exp_args:Nnx \use:nn
3442   { \int_set:Nn \l__stex_terms_downprec { #2 }
3443     \_stex_term_arg:nn { #1 }{ #3 }
3444   }
3445   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3446 }

```

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

`\_stex_term_math_assoc_arg:nnnn`

```

3447 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
3448   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3449   \tl_set:Nn \l_tmpb_tl {\_stex_term_math_arg:nnn{#1}{#2}}
3450   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 }}{
3451     \expandafter\if\expandafter\relax\noexpand#3
3452     \expandafter\_stex_terms_math_assoc_arg_maybe_sequence:N\expandafter#3
3453     \else\expandafter\_stex_terms_math_assoc_arg_simple:nn
3454     \expandafter{\expandafter}\expandafter#3\fi
3455   }{
3456     \_stex_terms_math_assoc_arg_simple:nn{#1}{#3}
3457   }
3458 }
3459
3460 \cs_new_protected:Nn \_stex_terms_math_assoc_arg_maybe_sequence:N {
3461   \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
3462   \str_if_empty:NTF \l_tmpa_str {
3463     \exp_args:Nx \cs_if_eq:NNTF {
3464       \tl_head:N #1
3465     } \stex_invoke_sequence:n {
3466       \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
3467       \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
3468       \tl_set:Nx \l_tmpa_tl {\prop_item:cn {stex_varseq\_l_tmpa_str\_prop}{notation}}
3469       \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
3470       \tl_set:Nx \l_tmpa_tl {\exp_not:N \exp_not:n{
3471         \exp_not:n{\exp_args:Nnx \use:nn} {
3472           \exp_not:n {
3473             \def\comp{\_varcomp}
3474             \str_set:Nn \l_stex_current_symbol_str
3475             } {varseq://\l_tmpa_str}
3476             \exp_not:n{ ##1 }
3477           }{
3478             \exp_not:n {
3479               \_stex_reset:N \comp
3480               \_stex_reset:N \l_stex_current_symbol_str
3481             }
3482           }
3483         }}}
3484       \exp_args:Nno \use:nn {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
3485       \seq_reverse:N \l_tmpa_seq
3486       \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
3487       \seq_map_inline:Nn \l_tmpa_seq {
3488         \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {

```

```

3489         \exp_args:Nno
3490         \l_tmpa_cs { ##1 } \l_tmpa_tl
3491     }
3492 }
3493 \tl_set:Nx \l_tmpa_tl {
3494     \stex_term_omv:nn {varseq://\l_tmpa_str}{
3495         \exp_args:No \exp_not:n \l_tmpa_tl
3496     }
3497 }
3498 \exp_args:No\l_tmpb_tl\l_tmpa_tl
3499 }{
3500     \stex_terms_math_assoc_arg_simple:nn{} { #1 }
3501 }
3502 } {
3503     \stex_terms_math_assoc_arg_simple:nn{} { #1 }
3504 }
3505 }
3506 }
3507
3508 \cs_new_protected:Nn \stex_terms_math_assoc_arg_simple:nn {
3509     \clist_set:Nn \l_tmpa_clist{ #2 }
3510     \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3511         \tl_set:Nn \l_tmpa_tl { #2 }
3512     }{
3513         \clist_reverse:N \l_tmpa_clist
3514         \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3515         \tl_set:Nx \l_tmpa_tl { \stex_term_arg:nn{A#1}{
3516             \exp_args:No \exp_not:n \l_tmpa_tl
3517         }}
3518         \clist_map_inline:Nn \l_tmpa_clist {
3519             \exp_args:NNNo \exp_args:NNNo \tl_set:No \l_tmpa_tl {
3520                 \exp_args:Nno
3521                 \l_tmpa_cs { \stex_term_arg:nn{A#1}{##1} } \l_tmpa_tl
3522             }
3523         }
3524     }
3525     \exp_args:No\l_tmpb_tl\l_tmpa_tl
3526 }

```

(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
3527 \tl_const:Nx \infprec {\int_use:N \c_max_int}
3528 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
3529 \int_new:N \l__stex_terms_downprec
3530 \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`

```
3531 \tl_set:Nn \l__stex_terms_left_bracket_str (
3532 \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

```
3533 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
3534   \bool_if:NTF \l__stex_terms_brackets_done_bool {
3535     \bool_set_false:N \l__stex_terms_brackets_done_bool
3536     #2
3537   } {
3538     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
3539       \bool_if:NTF \l_stex_inarray_bool { #2 }{
3540         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
3541         \dobrackets { #2 }
3542       }
3543     }{ #2 }
3544   }
3545 }
```

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

**`\dobrackets`**

```
3546 \bool_new:N \l__stex_terms_brackets_done_bool
3547 %\RequirePackage{scalerel}
3548 \cs_new_protected:Npn \dobrackets #1 {
3549   %\ThisStyle{\if D\m@switch
3550   %   \exp_args:Nnx \use:nn
3551   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
3552   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
3553   % \else
3554   \exp_args:Nnx \use:nn
3555   {
3556     \bool_set_true:N \l__stex_terms_brackets_done_bool
3557     \int_set:Nn \l__stex_terms_downprec \infprec
3558     \l__stex_terms_left_bracket_str
3559     #1
3560   }
3561   {
3562     \bool_set_false:N \l__stex_terms_brackets_done_bool
3563     \l__stex_terms_right_bracket_str
3564     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3565   }
3566   %\fi}
3567 }
```

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

**`\withbrackets`**

```
3568 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
3569   \exp_args:Nnx \use:nn
3570   {
3571     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
```

```

3572 \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
3573 #3
3574 }
3575 {
3576 \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
3577 {\l__stex_terms_left_bracket_str}
3578 \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
3579 {\l__stex_terms_right_bracket_str}
3580 }
3581 }

```

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

### `\STEXinvisible`

```

3582 \cs_new_protected:Npn \STEXinvisible #1 {
3583 \stex_annotate_invisible:n { #1 }
3584 }

```

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

OMDoc terms:

### `\_stex_term_math_oms:nnnn`

```

3585 \cs_new_protected:Nn \_stex_term_oms:nnn {
3586 \stex_annotate:nnn{ OMID }{ #2 }{
3587 \stex_highlight_term:nn { #1 } { #3 }
3588 }
3589 }
3590
3591 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
3592 \_stex_terms_maybe_brackets:nn { #3 }{
3593 \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3594 }
3595 }

```

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

### `\_stex_term_math_omv:nn`

```

3596 \cs_new_protected:Nn \_stex_term_omv:nn {
3597 \stex_annotate:nnn{ OMV }{ #1 }{
3598 \stex_highlight_term:nn { #1 } { #2 }
3599 }
3600 }

```

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

### `\_stex_term_math_oma:nnnn`

```

3601 \cs_new_protected:Nn \_stex_term_oma:nnn {
3602 \stex_annotate:nnn{ OMA }{ #2 }{
3603 \stex_highlight_term:nn { #1 } { #3 }
3604 }
3605 }
3606
3607 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
3608 \_stex_terms_maybe_brackets:nn { #3 }{
3609 \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }

```



```

3610 }
3611 }

```

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

`\_stex_term_math_omb:nnnn`

```

3612 \cs_new_protected:Nn \_stex_term_ombind:nnn {
3613   \stex_annotate:nnn{ OMBIND }{ #2 }{
3614     \stex_highlight_term:nn { #1 } { #3 }
3615   }
3616 }
3617
3618 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
3619   \__stex_terms_maybe_brackets:nn { #3 }{
3620     \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3621   }
3622 }

```

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

`\symref`  
`\symname`

```

3623 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3624
3625 \keys_define:nn { stex / symname } {
3626   pre      .tl_set_x:N = \l__stex_terms_pre_tl ,
3627   post     .tl_set_x:N = \l__stex_terms_post_tl ,
3628   root     .tl_set_x:N = \l__stex_terms_root_tl
3629 }
3630
3631 \cs_new_protected:Nn \stex_symname_args:n {
3632   \tl_clear:N \l__stex_terms_post_tl
3633   \tl_clear:N \l__stex_terms_pre_tl
3634   \tl_clear:N \l__stex_terms_root_str
3635   \keys_set:nn { stex / symname } { #1 }
3636 }
3637
3638 \NewDocumentCommand \symref { m m }{
3639   \let\compemph_uri_prev:\compemph@uri
3640   \let\compemph@uri\symrefemph@uri
3641   \STEXsymbol{#1}!{ #2 }
3642   \let\compemph@uri\compemph_uri_prev:
3643 }
3644
3645 \NewDocumentCommand \synonym { 0{} m m }{
3646   \stex_symname_args:n { #1 }
3647   \let\compemph_uri_prev:\compemph@uri
3648   \let\compemph@uri\symrefemph@uri
3649   % TODO
3650   \STEXsymbol{#2}!{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
3651   \let\compemph@uri\compemph_uri_prev:
3652 }
3653
3654 \NewDocumentCommand \symname { 0{} m }{
3655   \stex_symname_args:n { #1 }
3656   \stex_get_symbol:n { #2 }

```

```

3657 \str_set:Nx \l_tmpa_str {
3658   \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3659 }
3660 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3661
3662 \let\compemph_uri_prev:\compemph@uri
3663 \let\compemph@uri\symrefemph@uri
3664 \exp_args:NNx \use:nn
3665 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!{
3666   \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
3667 } }
3668 \let\compemph@uri\compemph_uri_prev:
3669 }
3670
3671 \NewDocumentCommand \Symname { 0{ } m }{
3672   \stex_symname_args:n { #1 }
3673   \stex_get_symbol:n { #2 }
3674   \str_set:Nx \l_tmpa_str {
3675     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3676   }
3677   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3678   \let\compemph_uri_prev:\compemph@uri
3679   \let\compemph@uri\symrefemph@uri
3680   \exp_args:NNx \use:nn
3681   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!{
3682     \exp_after:wN \stex_capitalize:n \l_tmpa_str
3683     \l__stex_terms_post_tl
3684   } }
3685   \let\compemph@uri\compemph_uri_prev:
3686 }

```

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

## 30.3 Notation Components

```

3687 <@@=stex_notationcomps>
3688 \cs_new_protected:Nn \stex_highlight_term:nn {
3689   #2
3690 }
3691
3692 \cs_new_protected:Nn \stex_unhighlight_term:n {
3693   % \latexml_if:TF {
3694   %   #1
3695   % } {
3696   %   \rustex_if:TF {
3697   %     #1
3698   %   } {
3699     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
3700   %   }
3701   % }
3702 }

```

`\stex_highlight_term:nn`

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

```

\comp
\compemph@uri 3703 \cs_new_protected:Npn \_comp #1 {
\compemph 3704 \str_if_empty:NF \l_stex_current_symbol_str {
\defemph 3705 \rustex_if:TF {
\defemph@uri 3706 \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
\symrefemph 3707 }{
\symrefemph@uri 3708 \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
\varernph 3709 }
\varernph@uri 3710 }
3711 }
3712
3713 \cs_new_protected:Npn \_varcomp #1 {
3714 \str_if_empty:NF \l_stex_current_symbol_str {
3715 \rustex_if:TF {
3716 \stex_annotate:nnn { varcomp }{ \l_stex_current_symbol_str }{ #1 }
3717 }{
3718 \exp_args:Nnx \varernph@uri { #1 } { \l_stex_current_symbol_str }
3719 }
3720 }
3721 }
3722
3723 \def\comp{\_comp}
3724
3725 \cs_new_protected:Npn \compemph@uri #1 #2 {
3726 \compemph{ #1 }
3727 }
3728
3729
3730 \cs_new_protected:Npn \compemph #1 {
3731 #1
3732 }
3733
3734 \cs_new_protected:Npn \defemph@uri #1 #2 {
3735 \defemph{#1}
3736 }
3737
3738 \cs_new_protected:Npn \defemph #1 {
3739 \textbf{#1}
3740 }
3741
3742 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3743 \symrefemph{#1}
3744 }
3745
3746 \cs_new_protected:Npn \symrefemph #1 {
3747 \textbf{#1}
3748 }
3749
3750 \cs_new_protected:Npn \varernph@uri #1 #2 {
3751 \varernph{#1}
3752 }
3753

```

```

3754 \cs_new_protected:Npn \varemp #1 {
3755     #1
3756 }

```

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

## `\ellipses`

```

3757 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix
\parrayline
\parraylineh
\parraycell
3758 \bool_new:N \l_stex_inarray_bool
3759 \bool_set_false:N \l_stex_inarray_bool
3760 \NewDocumentCommand \parray { m m } {
3761     \begingroup
3762     \bool_set_true:N \l_stex_inarray_bool
3763     \begin{array}{#1}
3764         #2
3765     \end{array}
3766     \endgroup
3767 }
3768
3769 \NewDocumentCommand \prmatrix { m } {
3770     \begingroup
3771     \bool_set_true:N \l_stex_inarray_bool
3772     \begin{matrix}
3773         #1
3774     \end{matrix}
3775     \endgroup
3776 }
3777
3778 \def \maybepline {
3779     \bool_if:NT \l_stex_inarray_bool {\hline}
3780 }
3781
3782 \def \parrayline #1 #2 {
3783     #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3784 }
3785
3786 \def \pmrow #1 { \parrayline{}{ #1 } }
3787
3788 \def \parraylineh #1 #2 {
3789     #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3790 }
3791
3792 \def \parraycell #1 {
3793     #1 \bool_if:NT \l_stex_inarray_bool {\&}
3794 }

```

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

## 30.4 Variables

3795 <@@=stex\_variables>

\stex\_invoke\_variable:n Invokes a variable

```

3796 \cs_new_protected:Nn \stex_invoke_variable:n {
3797   \if_mode_math:
3798     \exp_after:wN \__stex_variables_invoke_math:n
3799   \else:
3800     \exp_after:wN \__stex_variables_invoke_text:n
3801   \fi: {#1}
3802 }
3803
3804 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
3805   %TODO
3806 }
3807
3808
3809 \cs_new_protected:Nn \__stex_variables_invoke_math:n {
3810   \peek_charcode_remove:NTF ! {
3811     \peek_charcode_remove:NTF ! {
3812       \peek_charcode:NTF [ {
3813         \__stex_variables_invoke_op_custom:nw
3814       }{
3815         % TODO throw error
3816       }
3817     }{
3818       \__stex_variables_invoke_op:n { #1 }
3819     }
3820   }{
3821     \peek_charcode_remove:NTF * {
3822       \__stex_variables_invoke_text:n { #1 }
3823     }{
3824       \__stex_variables_invoke_math_ii:n { #1 }
3825     }
3826   }
3827 }
3828
3829 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
3830   \cs_if_exist:cTF {
3831     stex_var_op_notation_ #1 _cs
3832   }{
3833     \exp_args:Nnx \use:nn {
3834       \def\comp{\_varcomp}
3835       \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3836       \stex_term_omv:nn { var://#1 }{
3837         \use:c{stex_var_op_notation_ #1 _cs }
3838       }
3839     }{
3840       \stex_reset:N \comp
3841       \stex_reset:N \l_stex_current_symbol_str
3842     }
3843   }{
3844     \int_compare:nNnTF {\prop_item:cn {\l_stex_variable_#1_prop}{arity}} = 0{

```

```

3845     \stex_variables_invoke_math_ii:n {#1}
3846   }{
3847     \msg_error:nnxx{stex}{error/noop}{variable~#1}{ }
3848   }
3849 }
3850 }
3851
3852 \cs_new_protected:Npn \stex_variables_invoke_math_ii:n #1 {
3853   \cs_if_exist:cTF {
3854     stex_var_notation_#1_cs
3855   }{
3856     \tl_set:Nx \stex_symbol_after_invokation_tl {
3857       \stex_reset:N \comp
3858       \stex_reset:N \stex_symbol_after_invokation_tl
3859       \stex_reset:N \l_stex_current_symbol_str
3860       \bool_set_true:N \l_stex_allow_semantic_bool
3861     }
3862     \def\comp{\_varcomp}
3863     \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3864     \bool_set_false:N \l_stex_allow_semantic_bool
3865     \use:c{stex_var_notation_#1_cs}
3866   }{
3867     \msg_error:nnxx{stex}{error/nonotation}{variable~#1}{s}
3868   }
3869 }

```

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

## 30.5 Sequences

```

3870 <@@=stex_sequences>
3871
3872 \cs_new_protected:Nn \stex_invoke_sequence:n {
3873   \peek_charcode_remove:NTF ! {
3874     \stex_term_omv:nn {varseq://#1}{
3875       \exp_args:Nnx \use:nn {
3876         \def\comp{\_varcomp}
3877         \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3878         \prop_item:cn{stex_varseq_#1_prop}{notation}
3879       }{
3880         \stex_reset:N \comp
3881         \stex_reset:N \l_stex_current_symbol_str
3882       }
3883     }
3884   }{
3885     \bool_set_false:N \l_stex_allow_semantic_bool
3886     \def\comp{\_varcomp}
3887     \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3888     \tl_set:Nx \stex_symbol_after_invokation_tl {
3889       \stex_reset:N \comp
3890       \stex_reset:N \stex_symbol_after_invokation_tl
3891       \stex_reset:N \l_stex_current_symbol_str
3892       \bool_set_true:N \l_stex_allow_semantic_bool
3893     }

```

```
3894     \use:c { stex_varseq_#1_cs }
3895   }
3896 }
3897 </package>
```

## Chapter 31

# STEX -Structural Features Implementation

```
3898 ⟨*package⟩
3899
3900 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3901
3902     Warnings and error messages
3903 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3904     Symbol~#1~can~not~be~assigned~in~copymodule~#2
3905 }
3906 \msg_new:nnn{stex}{error/interpretmodule/nodfiniens}{
3907     Symbol~#1~not~assigned~in~interpretmodule~#2
3908 }
3909 \msg_new:nnn{stex}{error/unknownstructure}{
3910     No~structure~#1~found!
3911 }
3912
3913 \msg_new:nnn{stex}{error/unknownfield}{
3914     No~field~#1~in~instance~#2~found!\#3
3915 }
3916
3917 \msg_new:nnn{stex}{error/keyval}{
3918     Invalid~key=value~pair~#1
3919 }
3920 \msg_new:nnn{stex}{error/instantiate/missing}{
3921     Assignments~missing~in~instantiate:~#1
3922 }
3923 \msg_new:nnn{stex}{error/incompatible}{
3924     Incompatible~signature:~#1~(#2)~and~#3~(#4)
3925 }
3926
```



## 31.1 Imports with modification

```

3927 <@@=stex_copymodule>
3928 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
3929   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3930     \tl_set:Nn \l_tmpa_tl { #1 }
3931     \__stex_copymodule_get_symbol_from_cs:
3932   }{
3933     % argument is a string
3934     % is it a command name?
3935     \cs_if_exist:cTF { #1 }{
3936       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3937       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3938       \str_if_empty:NNTF \l_tmpa_str {
3939         \exp_args:Nx \cs_if_eq:NNTF {
3940           \tl_head:N \l_tmpa_tl
3941         } \stex_invoke_symbol:n {
3942           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
3943         }{
3944           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3945         }
3946       } {
3947         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3948       }
3949     }{
3950       % argument is not a command name
3951       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3952       % \l_stex_all_symbols_seq
3953     }
3954   }
3955 }
3956
3957 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
3958   \str_set:Nn \l_tmpa_str { #1 }
3959   \bool_set_false:N \l_tmpa_bool
3960   \bool_if:NF \l_tmpa_bool {
3961     \tl_set:Nn \l_tmpa_tl {
3962       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
3963     }
3964     \str_set:Nn \l_tmpa_str { #1 }
3965     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3966     \seq_map_inline:Nn #2 {
3967       \str_set:Nn \l_tmpb_str { ##1 }
3968       \str_if_eq:eeT { \l_tmpa_str } {
3969         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3970       } {
3971         \seq_map_break:n {
3972           \tl_set:Nn \l_tmpa_tl {
3973             \str_set:Nn \l_stex_get_symbol_uri_str {
3974               ##1
3975             }
3976           }
3977         }
3978       }

```

```

3979     }
3980     \l_tmpa_tl
3981   }
3982 }
3983
3984 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
3985   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3986     { \tl_tail:N \l_tmpa_tl }
3987   \tl_if_single:NTF \l_tmpa_tl {
3988     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3989       \exp_after:wN \str_set:Nn \exp_after:wN
3990         \l_stex_get_symbol_uri_str \l_tmpa_tl
3991       \__stex_copymodule_get_symbol_check:n { #1 }
3992     }{
3993       % TODO
3994       % tail is not a single group
3995     }
3996   }{
3997     % TODO
3998     % tail is not a single group
3999   }
4000 }
4001
4002 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
4003   \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
4004     \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
4005       :~\seq_use:Nn #1 {,~}
4006     }
4007   }
4008 }
4009
4010 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
4011   \stex_import_module_uri:nn { #1 } { #2 }
4012   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
4013   \stex_import_require_module:nnnn
4014     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4015     { \l_stex_import_path_str } { \l_stex_import_name_str }
4016   \stex_collect_imports:n { \l_stex_import_ns_str ? \l_stex_import_name_str }
4017   \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
4018   \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
4019   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4020     \seq_map_inline:cn {c_stex_module_###_constants}{
4021       \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
4022         ##1 ? ####1
4023       }
4024     }
4025   }
4026   \seq_clear:N \l_tmpa_seq
4027   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
4028     name      = \l_stex_current_copymodule_name_str ,
4029     module    = \l_stex_current_module_str ,
4030     from      = \l_stex_import_ns_str ? \l_stex_import_name_str ,
4031     includes  = \l_tmpa_seq ,
4032     fields    = \l_tmpa_seq

```

```

4033 }
4034 \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
4035   as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
4036   \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_seq {,
4037 \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
4038 \stex_if_smsmode:F {
4039   \begin{stex_annotate_env} {#4} {
4040     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4041   }
4042   \stex_annotate_invisible:nnn{domain}{\l_stex_import_ns_str ?\l_stex_import_name_str}{}}
4043 }
4044 %\bool_set_eq:NN \l__stex_copymodule_oldhtml_bool \_stex_html_do_output_bool
4045 %\bool_set_false:N \_stex_html_do_output_bool
4046 }
4047 \cs_new_protected:Nn \stex_copymodule_end:n {
4048   \def \l_tmpa_cs ##1 ##2 {#1}
4049   %\bool_set_eq:NN \_stex_html_do_output_bool \l__stex_copymodule_oldhtml_bool
4050   \tl_clear:N \l_tmpa_tl
4051   \tl_clear:N \l_tmpb_tl
4052   \prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
4053   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4054     \seq_map_inline:cn {c_stex_module_##1_constants}{
4055       \tl_clear:N \l_tmpc_tl
4056       \l_tmpa_cs{##1}{####1}
4057       \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
4058         \stex_add_constant_to_current_module:n {\use:c{l__stex_copymodule_copymodule_##1?####1_
4059         \tl_put_right:Nx \l_tmpa_tl {
4060           \prop_set_from_keyval:cn {
4061             l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_
4062           }}{
4063             \exp_after:wN \prop_to_keyval:N \csname
4064               l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_
4065             \endcsname
4066           }
4067           \seq_clear:c {
4068             l_stex_symdecl_
4069             \l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_name
4070             _notations
4071           }
4072         }
4073         \tl_put_right:Nx \l_tmpc_tl {
4074           \stex_copy_notations:nn {\l_stex_current_module_str ? \use:c{l__stex_copymodule_co
4075           \stex_if_smsmode:F{\stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_c
4076         }
4077         \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{l__stex_copymodul
4078         \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
4079           \tl_put_right:Nx \l_tmpc_tl {
4080             \stex_if_smsmode:F{\stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymo
4081           }
4082           \tl_put_right:Nx \l_tmpa_tl {
4083             \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
4084             \stex_invoke_symbol:n {
4085               \l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_
4086             }

```

```

4087     }
4088   }
4089 }
4090 }{
4091   \tl_put_right:Nx \l_tmpc_tl {
4092     \stex_copy_notations:nn {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str}
4093   }
4094   \stex_add_constant_to_current_module:n { \l_stex_current_copymodule_name_str / #####1
4095   \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1_prop}
4096   \prop_put:Nnx \l_tmpa_prop { name }{ \l_stex_current_copymodule_name_str / #####1 }
4097   \prop_put:Nnx \l_tmpa_prop { module }{ \l_stex_current_module_str }
4098   \tl_put_right:Nx \l_tmpa_tl {
4099     \prop_set_from_keyval:cn {
4100       l_stex_symdecl_ \l_stex_current_module_str ? \l_stex_current_copymodule_name_str
4101     }{
4102       \prop_to_keyval:N \l_tmpa_prop
4103     }
4104     \seq_clear:c {
4105       l_stex_symdecl_
4106       \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / #####1
4107       _notations
4108     }
4109   }
4110   \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str}
4111   \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
4112     \tl_put_right:Nx \l_tmpc_tl {
4113       \stex_if_smsmode:F{\stex_annotate_invisible:nnn{macroname}}{\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}
4114     }
4115     \tl_put_right:Nx \l_tmpa_tl {
4116       \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
4117         \stex_invoke_symbol:n {
4118           \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / #####1
4119         }
4120       }
4121     }
4122   }
4123 }
4124 \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4125   \tl_put_right:Nx \l_tmpc_tl {
4126     \stex_if_smsmode:F{
4127       $\stex_annotate_invisible:nnn{definiens}}{\exp_after:wN \exp_not:N\csname l__stex_copymodule_copymodule_##1?####1_def_tl\endcsname}
4128     }
4129   }
4130 }
4131 \tl_put_right:Nx \l_tmpb_tl {
4132   \stex_if_smsmode:TF{
4133     \exp_after:wN \exp_not:n \exp_after:wN {\l_tmpc_tl}
4134   }{
4135     \stex_annotate:nnn{assignment} {##1?####1} { \exp_after:wN \exp_not:n \exp_after:wN {\l_tmpc_tl}
4136   }
4137 }
4138 }
4139 }
4140 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \l_tmpa_seq

```

```

4141 \tl_put_left:Nx \l_tmpa_tl {
4142   \prop_set_from_keyval:cn {
4143     l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _pro
4144   }{
4145     \prop_to_keyval:N \l_stex_current_copymodule_prop
4146   }
4147 }
4148 \seq_gput_right:cx{c_stex_module_ \l_stex_current_module_str _copymodules}{
4149   \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4150 }
4151 \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl
4152 \stex_debug:nn{copymodule}{result:\meaning \l_tmpa_tl}
4153 \exp_args:Nx \stex_do_up_to_module:n {
4154   \exp_args:No \exp_not:n \l_tmpa_tl
4155 }
4156 \stex_debug:nn{copymodule}{output:\meaning \l_tmpb_tl}
4157 \l_tmpb_tl
4158 \stex_if_smsmode:F {
4159   \end{stex_annotate_env}
4160 }
4161 }
4162
4163 \NewDocumentEnvironment {copymodule} { 0{} m m}{
4164   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ copymodule }
4165   \stex_deactivate_macro:Nn \symdecl {module~environments}
4166   \stex_deactivate_macro:Nn \symdef {module~environments}
4167   \stex_deactivate_macro:Nn \notation {module~environments}
4168   \stex_reactivate_macro:N \assign
4169   \stex_reactivate_macro:N \renamedekl
4170   \stex_reactivate_macro:N \donotcopy
4171   \stex_smsmode_do:
4172 }{
4173   \stex_copymodule_end:n {}
4174 }
4175
4176 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
4177   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ interpretmodule }
4178   \stex_deactivate_macro:Nn \symdecl {module~environments}
4179   \stex_deactivate_macro:Nn \symdef {module~environments}
4180   \stex_deactivate_macro:Nn \notation {module~environments}
4181   \stex_reactivate_macro:N \assign
4182   \stex_reactivate_macro:N \renamedekl
4183   \stex_reactivate_macro:N \donotcopy
4184   \stex_smsmode_do:
4185 }{
4186   \stex_copymodule_end:n {
4187     \tl_if_exist:cF {
4188       l__stex_copymodule_copymodule_##1?##2_def_tl
4189     }{
4190       \str_if_eq:eeF {
4191         \prop_item:cn{
4192           l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
4193       }{ true }{
4194         \msg_error:nxxx{stex}{error/interpretmodule/noddefinens}{

```

```

4195         ##1?##2
4196     }\l_stex_current_copymodule_name_str}
4197 }
4198 }
4199 }
4200 }
4201
4202 \NewDocumentCommand \donotcopy { m }{
4203     \str_clear:N \l_stex_import_name_str
4204     \str_set:Nn \l_tmpa_str { #1 }
4205     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4206     \seq_map_inline:Nn \l_stex_all_modules_seq {
4207         \str_set:Nn \l_tmpb_str { ##1 }
4208         \str_if_eq:eeT { \l_tmpa_str } {
4209             \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4210         } {
4211             \seq_map_break:n {
4212                 \stex_if_do_html:T {
4213                     \stex_if_smsmode:F {
4214                         \stex_annotate_invisible:nnn{donotcopy}{##1}{
4215                             \stex_annotate:nnn{domain}{##1}{}}
4216                     }
4217                 }
4218             }
4219             \str_set_eq:NN \l_stex_import_name_str \l_tmpb_str
4220         }
4221     }
4222     \seq_map_inline:cn {c_stex_module_##1_copymodules}{
4223         \str_set:Nn \l_tmpb_str { #####1 }
4224         \str_if_eq:eeT { \l_tmpa_str } {
4225             \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4226         } {
4227             \seq_map_break:n {\seq_map_break:n {
4228                 \stex_if_do_html:T {
4229                     \stex_if_smsmode:F {
4230                         \stex_annotate_invisible:nnn{donotcopy}{#####1}{
4231                             \stex_annotate:nnn{domain}{
4232                                 \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4233                             }{}}
4234                     }
4235                 }
4236             }
4237             \str_set:Nx \l_stex_import_name_str {
4238                 \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4239             }
4240         }}
4241     }
4242 }
4243 }
4244 \str_if_empty:NTF \l_stex_import_name_str {
4245     % TODO throw error
4246 }{
4247     \stex_collect_imports:n {\l_stex_import_name_str }
4248     \seq_map_inline:Nn \l_stex_collect_imports_seq {

```

```

4249 \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
4250 \seq_map_inline:cn {c_stex_module_##1_constants}{
4251   \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? #####1 }
4252   \bool_lazy_any:nT {
4253     { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_name_str}}
4254     { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_macroname_str}}
4255     { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_def_tl}}
4256   }{
4257     % TODO throw error
4258   }
4259 }
4260 }
4261 \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
4262 \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_name_str }
4263 \prop_put:Nno \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
4264 }
4265 \stex_smsmode_do:
4266 }
4267
4268 \NewDocumentCommand \assign { m m }{
4269   \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
4270   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
4271   \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _def_tl}{#2}
4272   \stex_smsmode_do:
4273 }
4274
4275 \keys_define:nn { stex / renamedec1 } {
4276   name .str_set_x:N = \l_stex_renamedec1_name_str
4277 }
4278 \cs_new_protected:Nn \__stex_copymodule_renamedec1_args:n {
4279   \str_clear:N \l_stex_renamedec1_name_str
4280   \keys_set:nn { stex / renamedec1 } { #1 }
4281 }
4282
4283 \NewDocumentCommand \renamedec1 { 0{} m m }{
4284   \__stex_copymodule_renamedec1_args:n { #1 }
4285   \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
4286   \stex_debug:nn{renamedec1}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
4287   \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
4288   \str_if_empty:NTF \l_stex_renamedec1_name_str {
4289     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4290       \l_stex_get_symbol_uri_str
4291     } }
4292   } {
4293     \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex
4294     \stex_debug:nn{renamedec1}{@~\l_stex_current_module_str ? \l_stex_renamedec1_name_str}
4295     \prop_set_eq:cc {l_stex_symdecl_
4296       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
4297     }_prop
4298     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
4299     \seq_set_eq:cc {l_stex_symdecl_
4300       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
4301     }_notations
4302     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}

```

```

4303 \prop_put:cnx {l_stex_symdecl_
4304 \l_stex_current_module_str ? \l_stex_renameddecl_name_str
4305 _prop
4306 }{ name }{ \l_stex_renameddecl_name_str }
4307 \prop_put:cnx {l_stex_symdecl_
4308 \l_stex_current_module_str ? \l_stex_renameddecl_name_str
4309 _prop
4310 }{ module }{ \l_stex_current_module_str }
4311 \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
4312 \l_stex_current_module_str ? \l_stex_renameddecl_name_str
4313 }
4314 \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4315 \l_stex_current_module_str ? \l_stex_renameddecl_name_str
4316 } }
4317 }
4318 \stex_smsmode_do:
4319 }
4320
4321 \stex_deactivate_macro:Nn \assign {copymodules}
4322 \stex_deactivate_macro:Nn \renameddecl {copymodules}
4323 \stex_deactivate_macro:Nn \donotcopy {copymodules}
4324
4325
4326 \seq_new:N \l_stex_implicit_morphisms_seq
4327 \NewDocumentCommand \implicitmorphism { 0{ } m m }{
4328 \stex_import_module_uri:nn { #1 } { #2 }
4329 \stex_debug:nn{implicits}{
4330 Implicit~morphism:~
4331 \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4332 }
4333 \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
4334 \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4335 }{
4336 \msg_error:nnn{stex}{error/conflictingmodules}{
4337 \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4338 }
4339 }
4340
4341 % TODO
4342
4343
4344
4345 \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
4346 \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4347 }
4348 }
4349

```

## 31.2 The feature environment

structural@feature

```

4350 <@@=stex_features>
4351

```



```

4352 \NewDocumentEnvironment{structural_feature_module}{m m m}{
4353   \stex_if_in_module:F {
4354     \msg_set:nnn{stex}{error/nomodule}{
4355       Structural~Feature~has~to~occur~in~a~module:\\
4356       Feature~#2~of~type~#1\\
4357       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
4358     }
4359     \msg_error:nn{stex}{error/nomodule}
4360   }
4361
4362   \str_set_eq:NN \l_tmpa_str \l_stex_current_module_str
4363
4364   \stex_module_setup:nn{meta=NONE}{#2 - #1}
4365
4366   \stex_if_smsmode:F {
4367     \begin{stex_annotate_env}{feature:#1 }{\l_tmpa_str ? #2 - #1}
4368     \stex_annotate_invisible:nnn{header}{}{ #3 }
4369   }
4370 }{
4371   \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
4372   \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
4373   \stex_debug:nn{features}{
4374     Feature: \l_stex_last_feature_str
4375   }
4376   \stex_if_smsmode:F {
4377     \end{stex_annotate_env}
4378   }
4379 }

```

### 31.3 Structure

structure

```

4380 <@@=stex_structures>
4381 \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
4382   \prop_if_exist:cF {c_stex_module_ \l_stex_current_module_str _structures}{
4383     \prop_new:c {c_stex_module_ \l_stex_current_module_str _structures}
4384   }
4385   \prop_gput:cxn{c_stex_module_ \l_stex_current_module_str _structures}
4386   {#1}{#2}
4387 }
4388
4389 \keys_define:nn { stex / features / structure } {
4390   name .str_set_x:N = \l__stex_structures_name_str ,
4391 }
4392
4393 \cs_new_protected:Nn \__stex_structures_structure_args:n {
4394   \str_clear:N \l__stex_structures_name_str
4395   \keys_set:nn { stex / features / structure } { #1 }
4396 }
4397
4398 \NewDocumentEnvironment{mathstructure}{m O{}}{
4399   \__stex_structures_structure_args:n { #2 }
4400   \str_if_empty:NT \l__stex_structures_name_str {

```

```

4401 \str_set:Nx \l__stex_structures_name_str { #1 }
4402 }
4403 \stex_suppress_html:n {
4404 \exp_args:Nx \stex_symdecl_do:nn {
4405 name = \l__stex_structures_name_str ,
4406 def = {\STEXsymbol{module-type}}{
4407 \_stex_term_math_oms:nnnn {
4408 \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4409 { ns } ?
4410 \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4411 { name } / \l__stex_structures_name_str - structure
4412 }{}{}{}
4413 }}
4414 }{ #1 }
4415 }
4416 \exp_args:Nnnx
4417 \begin{structural_feature_module}{ structure }
4418 { \l__stex_structures_name_str }{}
4419 \stex_smsmode_do:
4420 }{
4421 \end{structural_feature_module}
4422 \stex_reset_up_to_module:n \l_stex_last_feature_str
4423 \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
4424 \seq_clear:N \l_tmpa_seq
4425 \seq_map_inline:Nn \l_stex_collect_imports_seq {
4426 \seq_map_inline:cn{c_stex_module_##1_constants}{
4427 \seq_put_right:Nn \l_tmpa_seq { ##1 ? ###1 }
4428 }
4429 }
4430 \exp_args:Nnno
4431 \prop_gput:cn {c_stex_module_\l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
4432 \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
4433 \stex_add_structure_to_current_module:nn
4434 \l__stex_structures_name_str
4435 \l_stex_last_feature_str
4436 \exp_args:Nx
4437 \stex_add_to_current_module:n {
4438 \tl_set:cn { #1 }{
4439 \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures_name_str }
4440 }
4441 }
4442 \exp_args:Nx
4443 \stex_do_up_to_module:n {
4444 \tl_set:cn { #1 }{
4445 \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures_name_str }
4446 }
4447 }
4448 }
4449
4450 \cs_new:Nn \stex_invoke_structure:nn {
4451 \stex_invoke_symbol:n { #1?#2 }
4452 }
4453
4454 \cs_new_protected:Nn \stex_get_structure:n {

```

```

4455 \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4456   \tl_set:Nn \l_tmpa_tl { #1 }
4457   \__stex_structures_get_from_cs:
4458 }{
4459   \cs_if_exist:cTF { #1 }{
4460     \cs_set_eq:Nc \l_tmpa_cs { #1 }
4461     \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
4462     \str_if_empty:NTF \l_tmpa_str {
4463       \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs} \stex_invoke_structure:nn {
4464         \__stex_structures_get_from_cs:
4465       }{
4466         \__stex_structures_get_from_string:n { #1 }
4467       }
4468     }{
4469       \__stex_structures_get_from_string:n { #1 }
4470     }
4471   }{
4472     \__stex_structures_get_from_string:n { #1 }
4473   }
4474 }
4475 }
4476
4477 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
4478   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4479   { \tl_tail:N \l_tmpa_tl }
4480   \str_set:Nx \l_tmpa_str {
4481     \exp_after:wN \use_i:nn \l_tmpa_tl
4482   }
4483   \str_set:Nx \l_tmpb_str {
4484     \exp_after:wN \use_ii:nn \l_tmpa_tl
4485   }
4486   \str_set:Nx \l_stex_get_structure_str {
4487     \l_tmpa_str ? \l_tmpb_str
4488   }
4489   \str_set:Nx \l_stex_get_structure_module_str {
4490     \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
4491   }
4492 }
4493
4494 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
4495   \tl_set:Nn \l_tmpa_tl {
4496     \msg_error:nnn{stex}{error/unknownstructure}{#1}
4497   }
4498   \str_set:Nn \l_tmpa_str { #1 }
4499   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4500
4501   \seq_map_inline:Nn \l_stex_all_modules_seq {
4502     \prop_if_exist:cT {c_stex_module_##1_structures} {
4503       \prop_map_inline:cn {c_stex_module_##1_structures} {
4504         \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
4505           \prop_map_break:n{\seq_map_break:n{
4506             \tl_set:Nn \l_tmpa_tl {
4507               \str_set:Nn \l_stex_get_structure_str {##1?####1}
4508               \str_set:Nn \l_stex_get_structure_module_str {####2}

```

```

4509     }
4510   }}
4511 }
4512 }
4513 }
4514 }
4515 \l_tmpa_tl
4516 }

```

**\instantiate**

```

4517
4518 \keys_define:nn { stex / instantiate } {
4519   name          .str_set_x:N = \l__stex_structures_name_str
4520 }
4521 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
4522   \str_clear:N \l__stex_structures_name_str
4523   \keys_set:nn { stex / instantiate } { #1 }
4524 }
4525
4526 \NewDocumentCommand \instantiate {m O{} m m m}{
4527   \begin{group}
4528     \stex_get_structure:n {#4}
4529     \__stex_structures_instantiate_args:n { #2 }
4530     \str_if_empty:NT \l__stex_structures_name_str {
4531       \str_set:Nn \l__stex_structures_name_str { #1 }
4532     }
4533     \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4534     \seq_clear:N \l__stex_structures_fields_seq
4535     \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4536     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4537       \seq_map_inline:cn {c_stex_module_##1_constants}{
4538         \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? #####1 }
4539       }
4540     }
4541
4542     \tl_if_empty:nF{#3}{
4543       \seq_set_split:Nnn \l_tmpa_seq , {#3}
4544       \prop_clear:N \l_tmpa_prop
4545       \seq_map_inline:Nn \l_tmpa_seq {
4546         \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4547         \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4548           \msg_error:nnn{stex}{error/keyval}{##1}
4549         }
4550         \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
4551         \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4552         \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
4553         \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}
4554         \exp_args:Nxx \str_if_eq:nnF
4555           {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4556           {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}{
4557           \msg_error:nnxxx{stex}{error/incompatible}
4558           {\l__stex_structures_dom_str}
4559           {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4560           {\l_stex_get_symbol_uri_str}

```

```

4561         {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
4562     }
4563     \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
4564 }
4565 }
4566
4567
4568
4569
4570 \seq_map_inline:Nn \l__stex_structures_fields_seq {
4571     \str_set:Nx \l_tmpa_str {field:\l__stex_structures_name_str . \prop_item:cn {l_stex_sy
4572     \stex_debug:nn{instantiate}{Field~\l_tmpa_str :~##1}
4573
4574     \exp_args:Nx \stex_do_up_to_module:n {
4575         \prop_set_from_keyval:cn { l_stex_symdecl_ \l_stex_current_module_str?\l_tmpa_str _p
4576             name = \l_tmpa_str ,
4577             args = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4578             arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4579             assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4580         }
4581         \seq_clear:c {l_stex_symdecl_ \l_stex_current_module_str?\l_tmpa_str _notations}
4582         \stex_add_constant_to_current_module:n {\l_tmpa_str}
4583     }
4584     \exp_args:Nx \stex_add_to_current_module:n {
4585         \prop_set_from_keyval:cn { l_stex_symdecl_ \l_stex_current_module_str?\l_tmpa_str _p
4586             name = \l_tmpa_str ,
4587             args = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4588             arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4589             assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4590         }
4591         \seq_clear:c {l_stex_symdecl_ \l_stex_current_module_str?\l_tmpa_str _notations}
4592         \stex_add_constant_to_current_module:n {\l_tmpa_str}
4593     }
4594
4595
4596
4597 \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4598     \stex_find_notation:nn{##1}{}
4599     \exp_args:Nx\stex_do_up_to_module:n {
4600         \seq_put_right:cn {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notation
4601     }
4602     \exp_args:Nx\stex_add_to_current_module:n {
4603         \seq_put_right:cn {l_stex_symdecl_f\l_stex_current_module_str?\l_tmpa_str _notatio
4604     }
4605
4606     \stex_copy_control_sequence:ccN
4607         {stex_notation_\l_stex_current_module_str?\l_tmpa_str _cs}
4608         {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4609         \l_tmpa_tl
4610     \exp_args:No \stex_do_up_to_module:n \l_tmpa_tl
4611     \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl
4612
4613
4614     \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{

```

```

4615         \tl_set_eq:Nc \l_tmpa_cs {stex_op_notation_##1\c_hash_str \l_stex_notation_variant
4616         \tl_set:Nx \l_tmpa_tl {
4617             \tl_set:cn
4618                 {stex_op_notation_\l_stex_current_module_str?\l_tmpa_str_cs}
4619                 { \exp_args:No \exp_not:n \l_tmpa_cs}
4620         }
4621     }
4622 }
4623 }
4624
4625 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\l_stex_cur
4626 }
4627
4628
4629 %\seq_if_empty:NF \l__stex_structures_fields_seq {
4630 % \msg_error:nnx{stex}{error/instantiate/missing}{\seq_use:Nn\l__stex_structures_fields
4631 %}
4632 \exp_args:Nx
4633 \stex_add_to_current_module:n {
4634     \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
4635     domain = \l_stex_get_structure_module_str ,
4636     \prop_to_keyval:N \l_tmpa_prop
4637 }
4638 \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
4639 }
4640 \exp_args:Nx
4641 \stex_do_up_to_module:n {
4642     \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
4643     domain = \l_stex_get_structure_module_str ,
4644     \prop_to_keyval:N \l_tmpa_prop
4645 }
4646 \tl_set:cn{ #1 }{\stex_invoke_instance:n{\l_stex_current_module_str?\l__stex_structur
4647 }
4648 \stex_debug:nn{instantiate}{
4649     Instance~\l_stex_current_module_str?\l__stex_structures_name_str \
4650     \prop_to_keyval:N \l_tmpa_prop
4651 }
4652 \exp_args:Nxx \stex_symdecl_do:nn {
4653     type={\STEXsymbol{module-type}}{
4654         \_stex_term_math_oms:nnnn {
4655             \l_stex_get_structure_module_str
4656         }{}{0}{}
4657     }}
4658 }{\l__stex_structures_name_str}
4659 \exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}
4660 \endgroup
4661 \stex_smsmode_do:\ignorespacesandpars
4662 }
4663
4664 \cs_new_protected:Nn \stex_symbol_or_var:n {
4665     \cs_if_exist:cTF{#1}{
4666         \cs_set_eq:Nc \l_tmpa_tl { #1 }
4667         \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4668         \str_if_empty:NTF \l_tmpa_str {

```

```

4669 \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
4670 \stex_invoke_variable:n {
4671   \bool_set_true:N \l_stex_symbol_or_var_bool
4672   \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4673   \str_set:Nx \l_stex_get_symbol_uri_str {
4674     \exp_after:wN \use:n \l_tmpa_tl
4675   }
4676   }{
4677     \bool_set_false:N \l_stex_symbol_or_var_bool
4678     \stex_get_symbol:n{#1}
4679   }
4680   }{
4681     \__stex_structures_symbolorvar_from_string:n{ #1 }
4682   }
4683   }{
4684     \__stex_structures_symbolorvar_from_string:n{ #1 }
4685   }
4686 }
4687
4688 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
4689   \prop_if_exist:cTF {l_stex_variable_#1 _prop}{
4690     \bool_set_true:N \l_stex_symbol_or_var_bool
4691     \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
4692   }{
4693     \bool_set_false:N \l_stex_symbol_or_var_bool
4694     \stex_get_symbol:n{#1}
4695   }
4696 }
4697
4698 \keys_define:nn { stex / varinstantiate } {
4699   name      .str_set_x:N = \l__stex_structures_name_str,
4700   bind      .choices:nn =
4701     {forall,exists}
4702     {\str_set:Nx \l__stex_structures_bind_str {\l_keys_choice_tl}}
4703 }
4704
4705 \cs_new_protected:Nn \__stex_structures_varinstantiate_args:n {
4706   \str_clear:N \l__stex_structures_name_str
4707   \str_clear:N \l__stex_structures_bind_str
4708   \keys_set:nn { stex / varinstantiate } { #1 }
4709 }
4710
4711 \NewDocumentCommand \varinstantiate {m O{}} m m m m){
4712   \begin{group}
4713     \stex_get_structure:n {#4}
4714     \__stex_structures_varinstantiate_args:n { #2 }
4715     \str_if_empty:NT \l__stex_structures_name_str {
4716       \str_set:Nn \l__stex_structures_name_str { #1 }
4717     }
4718     \stex_if_do_html:TF{
4719       \stex_annotate:nnn{varinstance}{\l__stex_structures_name_str}
4720     }{\use:n}
4721     {
4722       \stex_if_do_html:T{

```

```

4723     \stex_annotate:nnn{domain}{\l_stex_get_structure_module_str}{ }
4724 }
4725 \seq_clear:N \l__stex_structures_fields_seq
4726 \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4727 \seq_map_inline:Nn \l_stex_collect_imports_seq {
4728     \seq_map_inline:cn {c_stex_module_##1_constants}{
4729         \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4730     }
4731 }
4732 \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4733 \prop_clear:N \l_tmpa_prop
4734 \tl_if_empty:nF {#3} {
4735     \seq_set_split:Nnn \l_tmpa_seq , {#3}
4736     \seq_map_inline:Nn \l_tmpa_seq {
4737         \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4738         \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4739             \msg_error:nnn{stex}{error/keyval}{##1}
4740         }
4741         \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_structures_fields_seq
4742         \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4743         \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_in_seq:nn
4744         \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
4745         \stex_if_do_html:T{
4746             \stex_annotate:nnn{assign}{\l__stex_structures_dom_str,\l_stex_get_symbol_uri_str}{ }
4747         }
4748         \bool_if:NTF \l_stex_symbol_or_var_bool {
4749             \exp_args:Nxx \str_if_eq:nnF
4750                 {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4751                 {\prop_item:cn{l_stex_variable\l_stex_get_symbol_uri_str _prop}{args}}{
4752                 \msg_error:nnxxx{stex}{error/incompatible}
4753                 {\l__stex_structures_dom_str}
4754                 {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4755                 {\l_stex_get_symbol_uri_str}
4756                 {\prop_item:cn{l_stex_variable\l_stex_get_symbol_uri_str _prop}{args}}
4757             }
4758             \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n {
4759         }}{
4760             \exp_args:Nxx \str_if_eq:nnF
4761                 {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4762                 {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}{
4763                 \msg_error:nnxxx{stex}{error/incompatible}
4764                 {\l__stex_structures_dom_str}
4765                 {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4766                 {\l_stex_get_symbol_uri_str}
4767                 {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}
4768             }
4769             \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n {
4770         }}{
4771     }
4772 }
4773 \tl_gclear:N \g__stex_structures_aftergroup_tl
4774 \seq_map_inline:Nn \l__stex_structures_fields_seq {
4775     \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdecl
4776     \stex_debug:nn{varinstantiate}{Field~\l_tmpa_str :~##1}

```



```

4777 \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4778 \stex_find_notation:nn{##1}{}
4779 \cs_gset_eq:cc{g__stex_structures_tmpa_\l_tmpa_str_cs}
4780 {stex_notation_##1\c_hash_str \l_stex_notation_variant_str_cs}
4781 \stex_debug:nn{varinstantiate}{Notation:~\cs_meaning:c{g__stex_structures_tmpa_\l_
4782 \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str_cs}{
4783 \cs_gset_eq:cc {g__stex_structures_tmpa_op_\l_tmpa_str_cs}
4784 {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str_cs}
4785 \stex_debug:nn{varinstantiate}{Operator~Notation:~\cs_meaning:c{g__stex_struct
4786 }
4787 }
4788
4789 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4790 \prop_set_from_keyval:cn { l_stex_variable_ \l_tmpa_str_prop}{
4791 name = \l_tmpa_str ,
4792 args = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4793 arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4794 assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4795 }
4796 \cs_set_eq:cc {stex_var_notation_\l_tmpa_str_cs}
4797 {g__stex_structures_tmpa_\l_tmpa_str_cs}
4798 \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str_cs}
4799 {g__stex_structures_tmpa_op_\l_tmpa_str_cs}
4800 }
4801 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_inv
4802 }
4803 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4804 \prop_set_from_keyval:cn {l_stex_varinstance_\l_stex_structures_name_str_prop }{
4805 domain = \l_stex_get_structure_module_str ,
4806 \prop_to_keyval:N \l_tmpa_prop
4807 }
4808 \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
4809 \tl_set:cn {l_stex_varinstance_\l_stex_structures_name_str_op_tl}{
4810 \exp_args:Nnx \exp_not:N \use:nn {
4811 \str_set:Nn \exp_not:N \l_stex_current_symbol_str {var://\l__stex_structures_nam
4812 \stex_term_omv:nn {var://\l__stex_structures_name_str}{
4813 \exp_not:n{
4814 \varcomp{#5}
4815 }
4816 }
4817 }{
4818 \exp_not:n{\stex_reset:N \l_stex_current_symbol_str}
4819 }
4820 }
4821 }
4822 }
4823 \stex_debug:nn{varinstantiate}{\expandafter\detokenize\expandafter{\g__stex_structures_a
4824 \aftergroup\g__stex_structures_aftergroup_tl
4825 \endgroup
4826 \stex_smsmode_do:\ignorespacesandpars
4827 }
4828
4829 \cs_new_protected:Nn \stex_invoke_instance:n {
4830 \peek_charcode_remove:NTF ! {

```

```

4831     \stex_invoke_symbol:n{#1}
4832   }{
4833     \_stex_invoke_instance:nn {#1}
4834   }
4835 }
4836
4837
4838 \cs_new_protected:Nn \stex_invoke_varinstance:n {
4839   \peek_charcode_remove:NTF ! {
4840     \exp_args:Nnx \use:nn {
4841       \def\comp{\_varcomp}
4842       \use:c{l_stex_varinstance_#1_op_tl}
4843     }{
4844       \_stex_reset:N \comp
4845     }
4846   }{
4847     \_stex_invoke_varinstance:nn {#1}
4848   }
4849 }
4850
4851 \cs_new_protected:Nn \_stex_invoke_instance:nn {
4852   \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
4853     \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
4854   }{
4855     \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
4856     \msg_error:nnxxx{stex}{error/unknownfield}{#2}{#1}{
4857       \prop_to_keyval:N \l_tmpa_prop
4858     }
4859   }
4860 }
4861
4862 \cs_new_protected:Nn \_stex_invoke_varinstance:nn {
4863   \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
4864     \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
4865     \l_tmpa_tl
4866   }{
4867     \msg_error:nnnnn{stex}{error/unknownfield}{#2}{#1}{
4868     }
4869   }

```

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

\stex\_invoke\_structure:nnn

```

4870 % #1: URI of the instance
4871 % #2: URI of the instantiated module
4872 \cs_new_protected:Nn \stex_invoke_structure:nnn {
4873   \tl_if_empty:nTF{ #3 }{
4874     \prop_set_eq:Nc \l__stex_structures_structure_prop {
4875       c_stex_feature_ #2 _prop
4876     }
4877     \tl_clear:N \l_tmpa_tl
4878     \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
4879     \seq_map_inline:Nn \l_tmpa_seq {
4880       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }

```

```

4881 \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
4882 \cs_if_exist:cT {
4883   stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
4884 }{
4885   \tl_if_empty:NF \l_tmpa_tl {
4886     \tl_put_right:Nn \l_tmpa_tl {,}
4887   }
4888   \tl_put_right:Nx \l_tmpa_tl {
4889     \stex_invoke_symbol:n {#1/\l_tmpa_str}!
4890   }
4891 }
4892 }
4893 \exp_args:No \mathstrut \l_tmpa_tl
4894 }{
4895   \stex_invoke_symbol:n{#1/#3}
4896 }
4897 }

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

4898 </package>

```

## Chapter 32

# STEX -Statements Implementation

```
4899 <*package>
4900
4901 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4902
4903 <@@=stex_statements>
4904
4905 Warnings and error messages
4906
4907
4908 \titleemph
4909 \def\titleemph#1{\textbf{#1}}
4910
4911 (End definition for \titleemph. This function is documented on page ??.)
```

### 32.1 Definitions

#### definiendum

```
4906 \keys_define:nn {stex / definiendum }{
4907   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
4908   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
4909   root     .str_set_x:N    = \l__stex_statements_definiendum_root_str,
4910   gfa      .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
4911 }
4912 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
4913   \str_clear:N \l__stex_statements_definiendum_root_str
4914   \tl_clear:N \l__stex_statements_definiendum_post_tl
4915   \str_clear:N \l__stex_statements_definiendum_gfa_str
4916   \keys_set:nn { stex / definiendum }{ #1 }
4917 }
4918 \NewDocumentCommand \definiendum { O{} m m } {
4919   \__stex_statements_definiendum_args:n { #1 }
4920   \stex_get_symbol:n { #2 }
4921   \stex_ref_new_sym_target:n \l__stex_get_symbol_uri_str
4922   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
4923     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```

4924     \tl_set:Nn \l_tmpa_tl { #3 }
4925   } {
4926     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
4927     \tl_set:Nn \l_tmpa_tl {
4928       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
4929     }
4930   }
4931 } {
4932   \tl_set:Nn \l_tmpa_tl { #3 }
4933 }
4934
4935 % TODO root
4936 \rustex_if:TF {
4937   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
4938 } {
4939   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
4940 }
4941 }
4942 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

## definame

```

4943
4944 \NewDocumentCommand \definame { 0{ } m } {
4945   \__stex_statements_definiendum_args:n { #1 }
4946   % TODO: root
4947   \stex_get_symbol:n { #2 }
4948   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4949   \str_set:Nx \l_tmpa_str {
4950     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4951   }
4952   \str_replace_all:Nnn \l_tmpa_str {-} {~}
4953   \rustex_if:TF {
4954     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4955       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4956     }
4957   } {
4958     \exp_args:Nnx \defemph@uri {
4959       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4960     } { \l_stex_get_symbol_uri_str }
4961   }
4962 }
4963 \stex_deactivate_macro:Nn \definame {definition~environments}
4964
4965 \NewDocumentCommand \Definame { 0{ } m } {
4966   \__stex_statements_definiendum_args:n { #1 }
4967   \stex_get_symbol:n { #2 }
4968   \str_set:Nx \l_tmpa_str {
4969     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4970   }
4971   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4972   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4973   \rustex_if:TF {

```

```

4974 \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4975 \l_tmpa_str\l__stex_statements_definiendum_post_tl
4976 }
4977 } {
4978 \exp_args:Nnx \defemph@uri {
4979 \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
4980 } { \l_stex_get_symbol_uri_str }
4981 }
4982 }
4983 \stex_deactivate_macro:Nn \Definame {definition~environments}
4984
4985 \NewDocumentCommand \premise { m }{
4986 \stex_annotate:nnn{ premise }{}{ #1 }
4987 }
4988 \NewDocumentCommand \conclusion { m }{
4989 \stex_annotate:nnn{ conclusion }{}{ #1 }
4990 }
4991 \NewDocumentCommand \definiens { 0{} m }{
4992 \str_clear:N \l_stex_get_symbol_uri_str
4993 \tl_if_empty:nF {#1} {
4994 \stex_get_symbol:n { #1 }
4995 }
4996 \str_if_empty:NT \l_stex_get_symbol_uri_str {
4997 \int_compare:nNnTF {\clist_count:N \l__stex_statements_sdefinition_for_clist} = 1 {
4998 \str_set:Nx \l_stex_get_symbol_uri_str {\clist_item:Nn \l__stex_statements_sdefinition
4999 }{
5000 % TODO throw error
5001 }
5002 }
5003 \str_if_eq:eeT {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{module}}{
5004 {\l_stex_current_module_str}{
5005 \str_if_eq:eeF {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defin
5006 }{true}}{
5007 \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5008 \exp_args:Nx \stex_add_to_current_module:n {
5009 \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5010 }
5011 }
5012 }
5013 \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
5014 }
5015
5016 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
5017 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
5018 \stex_deactivate_macro:Nn \definiens {definition~environments}
5019

```

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

## sdefinition

```

5020
5021 \keys_define:nn {stex / sdefinition }{
5022 type .str_set_x:N = \sdefinitiontype,
5023 id .str_set_x:N = \sdefinitionid,

```

```

5024   name      .str_set_x:N = \sdefinitionname,
5025   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5026   title     .tl_set:N    = \sdefinitiontitle
5027 }
5028 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
5029   \str_clear:N \sdefinitiontype
5030   \str_clear:N \sdefinitionid
5031   \str_clear:N \sdefinitionname
5032   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5033   \tl_clear:N \sdefinitiontitle
5034   \keys_set:nn { stex / sdefinition }{ #1 }
5035 }
5036
5037 \NewDocumentEnvironment{sdefinition}{0{}}{
5038   \__stex_statements_sdefinition_args:n{ #1 }
5039   \stex_reactivate_macro:N \definiendum
5040   \stex_reactivate_macro:N \definame
5041   \stex_reactivate_macro:N \Definame
5042   \stex_reactivate_macro:N \premise
5043   \stex_reactivate_macro:N \definiens
5044   \stex_if_smsmode:F{
5045     \seq_clear:N \l_tmpa_seq
5046     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5047       \tl_if_empty:NF{ ##1 }{
5048         \stex_get_symbol:n { ##1 }
5049         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5050           \l_stex_get_symbol_uri_str
5051         }
5052       }
5053     }
5054     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpa_seq
5055     \exp_args:Nnnx
5056     \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {,}}
5057     \str_if_empty:NF \sdefinitiontype {
5058       \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5059     }
5060     \str_if_empty:NF \sdefinitionname {
5061       \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5062     }
5063     \clist_set:Nn \l_tmpa_clist \sdefinitiontype
5064     \tl_clear:N \l_tmpa_tl
5065     \clist_map_inline:Nn \l_tmpa_clist {
5066       \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
5067         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
5068       }
5069     }
5070     \tl_if_empty:NTF \l_tmpa_tl {
5071       \__stex_statements_sdefinition_start:
5072     }{
5073       \l_tmpa_tl
5074     }
5075   }
5076   \stex_ref_new_doc_target:n \sdefinitionid
5077   \stex_smsmode_do:

```

```

5078 }{
5079   \stex_suppress_html:n {
5080     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5081   }
5082   \stex_if_smsmode:F {
5083     \clist_set:Nn \l_tmpa_clist \sdefinitiontype
5084     \tl_clear:N \l_tmpa_tl
5085     \clist_map_inline:Nn \l_tmpa_clist {
5086       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
5087         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
5088       }
5089     }
5090     \tl_if_empty:NTF \l_tmpa_tl {
5091       \__stex_statements_sdefinition_end:
5092     }{
5093       \l_tmpa_tl
5094     }
5095     \end{stex_annotate_env}
5096   }
5097 }

```

### **\stexpatchdefinition**

```

5098 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
5099   \par\noindent\titllemph{Definition\tl_if_empty:NF \sdefinitiontitle {
5100     ~(\sdefinitiontitle)
5101   }~}
5102 }
5103 \cs_new_protected:Nn \__stex_statements_sdefinition_end: { \par\medskip}
5104
5105 \newcommand\stexpatchdefinition[3] [] {
5106   \str_set:Nx \l_tmpa_str{ #1 }
5107   \str_if_empty:NTF \l_tmpa_str {
5108     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
5109     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
5110   }{
5111     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
5112     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
5113   }
5114 }

```

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

### **\inlinedef inline:**

```

5115 \keys_define:nn {stex / inlinedef }{
5116   type      .str_set_x:N = \sdefinitiontype,
5117   id        .str_set_x:N = \sdefinitionid,
5118   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5119   name      .str_set_x:N = \sdefinitionname
5120 }
5121 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
5122   \str_clear:N \sdefinitiontype
5123   \str_clear:N \sdefinitionid
5124   \str_clear:N \sdefinitionname
5125   \clist_clear:N \l__stex_statements_sdefinition_for_clist

```



```

5126 \keys_set:nn { stex / inlinedef }{ #1 }
5127 }
5128 \NewDocumentCommand \inlinedef { 0{} m } {
5129   \beginingroup
5130   \__stex_statements_inlinedef_args:n{ #1 }
5131   \stex_reactivate_macro:N \definiendum
5132   \stex_reactivate_macro:N \definame
5133   \stex_reactivate_macro:N \Definame
5134   \stex_reactivate_macro:N \premise
5135   \stex_reactivate_macro:N \definiens
5136   \stex_ref_new_doc_target:n \sdefinitionid
5137   \stex_if_smsmode:TF{\stex_suppress_html:n {
5138     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5139   }}{
5140     \seq_clear:N \l_tmpa_seq
5141     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5142       \tl_if_empty:NF{ ##1 }{
5143         \stex_get_symbol:n { ##1 }
5144         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5145           \l_stex_get_symbol_uri_str
5146         }
5147       }
5148     }
5149     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpa_seq
5150     \exp_args:Nnx
5151     \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
5152       \str_if_empty:NF \sdefinitiontype {
5153         \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5154       }
5155       #2
5156       \str_if_empty:NF \sdefinitionname {
5157         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sdefinitionname}}
5158         \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5159       }
5160     }
5161   }
5162   \endgroup
5163   \stex_smsmode_do:
5164 }

```

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

## 32.2 Assertions

**sassertion**

```

5165 \keys_define:nn {stex / sassertion }{
5166   type      .str_set_x:N = \sassertiontype,
5167   id        .str_set_x:N = \sassertionid,
5168   title     .tl_set:N     = \sassertiontitle ,
5169   for       .clist_set:N  = \l__stex_statements_sassertion_for_clist ,
5170   name      .str_set_x:N  = \sassertionname
5171 }
5172 }

```

```

5173 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
5174   \str_clear:N \sassertiontype
5175   \str_clear:N \sassertionid
5176   \str_clear:N \sassertionname
5177   \clist_clear:N \l__stex_statements_sassertion_for_clist
5178   \tl_clear:N \sassertiontitle
5179   \keys_set:nn { stex / sassertion }{ #1 }
5180 }
5181
5182 %\tl_new:N \g__stex_statements_aftergroup_tl
5183
5184 \NewDocumentEnvironment{sassertion}{0{}}{
5185   \__stex_statements_sassertion_args:n{ #1 }
5186   \stex_reactivate_macro:N \premise
5187   \stex_reactivate_macro:N \conclusion
5188   \stex_if_smsmode:F {
5189     \seq_clear:N \l_tmpa_seq
5190     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5191       \tl_if_empty:nF{ ##1 }{
5192         \stex_get_symbol:n { ##1 }
5193         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5194           \l_stex_get_symbol_uri_str
5195         }
5196       }
5197     }
5198     \exp_args:Nnnx
5199     \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}
5200     \str_if_empty:NF \sassertiontype {
5201       \stex_annotate_invisible:nnn{type}{\sassertiontype}{ }
5202     }
5203     \str_if_empty:NF \sassertionname {
5204       \stex_annotate_invisible:nnn{statementname}{\sassertionname}{ }
5205     }
5206     \clist_set:Nn \l_tmpa_clist \sassertiontype
5207     \tl_clear:N \l_tmpa_tl
5208     \clist_map_inline:Nn \l_tmpa_clist {
5209       \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
5210         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
5211       }
5212     }
5213     \tl_if_empty:NTF \l_tmpa_tl {
5214       \__stex_statements_sassertion_start:
5215     }{
5216       \l_tmpa_tl
5217     }
5218   }
5219   \str_if_empty:NTF \sassertionid {
5220     \str_if_empty:NF \sassertionname {
5221       \stex_ref_new_doc_target:n { }
5222     }
5223   } {
5224     \stex_ref_new_doc_target:n \sassertionid
5225   }
5226   \stex_smsmode_do:

```

```

5227 }{
5228   \str_if_empty:NF \sassertionname {
5229     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5230     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5231   }
5232   \stex_if_smsmode:F {
5233     \clist_set:Nn \l_tmpa_clist \sassertiontype
5234     \tl_clear:N \l_tmpa_tl
5235     \clist_map_inline:Nn \l_tmpa_clist {
5236       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
5237         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
5238       }
5239     }
5240     \tl_if_empty:NTF \l_tmpa_tl {
5241       \__stex_statements_sassertion_end:
5242     }{
5243       \l_tmpa_tl
5244     }
5245     \end{stex_annotate_env}
5246   }
5247 }

```

## `\stexpatchassertion`

```

5248
5249 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
5250   \par\noindent\titllemph{Assertion~\tl_if_empty:NF \sassertiontitle {
5251     (\sassertiontitle)
5252   }~}
5253 }
5254 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
5255
5256 \newcommand\stexpatchassertion[3] [] {
5257   \str_set:Nx \l_tmpa_str{ #1 }
5258   \str_if_empty:NTF \l_tmpa_str {
5259     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
5260     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
5261   }{
5262     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
5263     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
5264   }
5265 }

```

(End definition for `\stexpatchassertion`. This function is documented on page [42](#).)

## `\inlineass` inline:

```

5266 \keys_define:nn {stex / inlineass }{
5267   type .str_set_x:N = \sassertiontype,
5268   id .str_set_x:N = \sassertionid,
5269   for .clist_set:N = \l__stex_statements_sassertion_for_clist ,
5270   name .str_set_x:N = \sassertionname
5271 }
5272 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
5273   \str_clear:N \sassertiontype
5274   \str_clear:N \sassertionid

```

```

5275 \str_clear:N \sassertionname
5276 \clist_clear:N \l__stex_statements_sassertion_for_clist
5277 \keys_set:nn { stex / inlineass }{ #1 }
5278 }
5279 \NewDocumentCommand \inlineass { 0{} m } {
5280 \begin{group}
5281 \stex_reactivate_macro:N \premise
5282 \stex_reactivate_macro:N \conclusion
5283 \__stex_statements_inlineass_args:n{ #1 }
5284 \str_if_empty:NTF \sassertionid {
5285 \str_if_empty:NF \sassertionname {
5286 \stex_ref_new_doc_target:n {}
5287 }
5288 } {
5289 \stex_ref_new_doc_target:n \sassertionid
5290 }
5291
5292 \stex_if_smsmode:TF{
5293 \str_if_empty:NF \sassertionname {
5294 \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sassertionname}}
5295 \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5296 }
5297 }{
5298 \seq_clear:N \l_tmpa_seq
5299 \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5300 \tl_if_empty:NF{ ##1 }{
5301 \stex_get_symbol:n { ##1 }
5302 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5303 \l_stex_get_symbol_uri_str
5304 }
5305 }
5306 }
5307 \exp_args:Nnx
5308 \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{
5309 \str_if_empty:NF \sassertiontype {
5310 \stex_annotate_invisible:nnn{typestrings}{\sassertiontype}{}
5311 }
5312 #2
5313 \str_if_empty:NF \sassertionname {
5314 \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sassertionname}}
5315 \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5316 \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
5317 }
5318 }
5319 }
5320 \end{group}
5321 \stex_smsmode_do:
5322 }

```

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

## 32.3 Examples

sexample

```

5323
5324 \keys_define:nn {stex / sexample }{
5325   type      .str_set_x:N = \exampletype,
5326   id         .str_set_x:N = \sexampleid,
5327   title      .tl_set:N     = \sexampletitle,
5328   name       .str_set_x:N = \sexamplename ,
5329   for        .clist_set:N = \l__stex_statements_sexample_for_clist,
5330 }
5331 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
5332   \str_clear:N \sexampletype
5333   \str_clear:N \sexampleid
5334   \str_clear:N \sexamplename
5335   \tl_clear:N \sexampletitle
5336   \clist_clear:N \l__stex_statements_sexample_for_clist
5337   \keys_set:nn { stex / sexample }{ #1 }
5338 }
5339
5340 \NewDocumentEnvironment{sexample}{0{}}{
5341   \__stex_statements_sexample_args:n{ #1 }
5342   \stex_reactivate_macro:N \premise
5343   \stex_reactivate_macro:N \conclusion
5344   \stex_if_smsmode:F {
5345     \seq_clear:N \l_tmpa_seq
5346     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5347       \tl_if_empty:NF{ ##1 }{
5348         \stex_get_symbol:n { ##1 }
5349         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5350           \l_stex_get_symbol_uri_str
5351         }
5352       }
5353     }
5354     \exp_args:Nnnx
5355     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
5356     \str_if_empty:NF \sexampletype {
5357       \stex_annotate_invisible:nnn{typestrings}{\sexampletype}{}
5358     }
5359     \str_if_empty:NF \sexamplename {
5360       \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
5361     }
5362     \clist_set:N \l_tmpa_clist \sexampletype
5363     \tl_clear:N \l_tmpa_tl
5364     \clist_map_inline:Nn \l_tmpa_clist {
5365       \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
5366         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
5367       }
5368     }
5369     \tl_if_empty:NTF \l_tmpa_tl {
5370       \__stex_statements_sexample_start:
5371     }{
5372       \l_tmpa_tl
5373     }

```

```

5374 }
5375 \str_if_empty:NF \sexampleid {
5376   \stex_ref_new_doc_target:n \sexampleid
5377 }
5378 \stex_smsmode_do:
5379 ){
5380   \str_if_empty:NF \sexamplename {
5381     \stex_suppress_html:n{\stex_symdecl_do:nn}{\sexamplename}}
5382   }
5383   \stex_if_smsmode:F {
5384     \clist_set:Nn \l_tmpa_clist \sexamplotype
5385     \tl_clear:N \l_tmpa_tl
5386     \clist_map_inline:Nn \l_tmpa_clist {
5387       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
5388         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
5389       }
5390     }
5391     \tl_if_empty:NTF \l_tmpa_tl {
5392       \__stex_statements_sexample_end:
5393     }{
5394       \l_tmpa_tl
5395     }
5396     \end{stex_annotate_env}
5397   }
5398 }

```

**\stexpatchexample**

```

5399
5400 \cs_new_protected:Nn \__stex_statements_sexample_start: {
5401   \par\noindent\titleemph{Example~\tl_if_empty:NF \sexamplotype {
5402     (\sexamplotype)
5403   }~}
5404 }
5405 \cs_new_protected:Nn \__stex_statements_sexample_end: { \par\medskip}
5406
5407 \newcommand\stexpatchexample[3]{} {
5408   \str_set:Nx \l_tmpa_str{ #1 }
5409   \str_if_empty:NTF \l_tmpa_str {
5410     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
5411     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
5412   }{
5413     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
5414     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
5415   }
5416 }

```

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

**\inlineex** inline:

```

5417 \keys_define:nn {stex / inlineex }{
5418   type .str_set_x:N = \sexamplotype,
5419   id .str_set_x:N = \sexampleid,
5420   for .clist_set:N = \l__stex_statements_sexample_for_clist ,
5421   name .str_set_x:N = \sexamplename

```

```

5422 }
5423 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
5424   \str_clear:N \sexamplotype
5425   \str_clear:N \sexampleid
5426   \str_clear:N \sexamplename
5427   \clist_clear:N \l__stex_statements_sexample_for_clist
5428   \keys_set:nn { stex / inlineex }{ #1 }
5429 }
5430 \NewDocumentCommand \inlineex { 0{ } m } {
5431   \beginngroup
5432   \stex_reactivate_macro:N \premise
5433   \stex_reactivate_macro:N \conclusion
5434   \__stex_statements_inlineex_args:n{ #1 }
5435   \str_if_empty:NF \sexampleid {
5436     \stex_ref_new_doc_target:n \sexampleid
5437   }
5438   \stex_if_smsmode:TF{
5439     \str_if_empty:NF \sexamplename {
5440       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5441     }
5442   }{
5443     \seq_clear:N \l_tmpa_seq
5444     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5445       \tl_if_empty:nF{ ##1 }{
5446         \stex_get_symbol:n { ##1 }
5447         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5448           \l_stex_get_symbol_uri_str
5449         }
5450       }
5451     }
5452     \exp_args:Nnx
5453     \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpa_seq {,}}{
5454       \str_if_empty:NF \sexamplotype {
5455         \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{ }
5456       }
5457       #2
5458       \str_if_empty:NF \sexamplename {
5459         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5460         \stex_annotate_invisible:nnn{statementname}{\sexamplename}{ }
5461       }
5462     }
5463   }
5464   \endgroup
5465   \stex_smsmode_do:
5466 }

```

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

## 32.4 Logical Paragraphs

sparagraph

```

5467 \keys_define:nn { stex / sparagraph } {
5468   id          .str_set_x:N    = \sparagraphid ,

```

```

5469 title .tl_set:N = \l_stex_sparagraph_title_tl ,
5470 type .str_set_x:N = \sparagraphtype ,
5471 for .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5472 from .tl_set:N = \sparagraphfrom ,
5473 to .tl_set:N = \sparagraphto ,
5474 start .tl_set:N = \l_stex_sparagraph_start_tl ,
5475 name .str_set:N = \sparagraphname
5476 }
5477
5478 \cs_new_protected:Nn \stex_sparagraph_args:n {
5479 \tl_clear:N \l_stex_sparagraph_title_tl
5480 \tl_clear:N \sparagraphfrom
5481 \tl_clear:N \sparagraphto
5482 \tl_clear:N \l_stex_sparagraph_start_tl
5483 \str_clear:N \sparagraphid
5484 \str_clear:N \sparagraphtype
5485 \clist_clear:N \l__stex_statements_sparagraph_for_clist
5486 \str_clear:N \sparagraphname
5487 \keys_set:nn { stex / sparagraph }{ #1 }
5488 }
5489 \newif\if@in@omtext\@in@omtextfalse
5490
5491 \NewDocumentEnvironment {sparagraph} { 0{} } {
5492 \stex_sparagraph_args:n { #1 }
5493 \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5494 \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
5495 }{
5496 \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
5497 }
5498 \@in@omtexttrue
5499 \stex_if_smsmode:F {
5500 \seq_clear:N \l_tmpa_seq
5501 \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5502 \tl_if_empty:NF{ ##1 }{
5503 \stex_get_symbol:n { ##1 }
5504 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5505 \l_stex_get_symbol_uri_str
5506 }
5507 }
5508 }
5509 \exp_args:Nnnx
5510 \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
5511 \str_if_empty:NF \sparagraphtype {
5512 \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{ }
5513 }
5514 \str_if_empty:NF \sparagraphfrom {
5515 \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{ }
5516 }
5517 \str_if_empty:NF \sparagraphto {
5518 \stex_annotate_invisible:nnn{to}{\sparagraphto}{ }
5519 }
5520 \str_if_empty:NF \sparagraphname {
5521 \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{ }
5522 }

```



```

5523 \clist_set:No \l_tmpa_clist \sparagraphtype
5524 \tl_clear:N \l_tmpa_tl
5525 \clist_map_inline:Nn \sparagraphtype {
5526   \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
5527     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
5528   }
5529 }
5530 \tl_if_empty:NTF \l_tmpa_tl {
5531   \__stex_statements_sparagraph_start:
5532 }{
5533   \l_tmpa_tl
5534 }
5535 }
5536 \clist_set:No \l_tmpa_clist \sparagraphtype
5537 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}
5538 {
5539   \stex_reactivate_macro:N \definiendum
5540   \stex_reactivate_macro:N \definame
5541   \stex_reactivate_macro:N \Definame
5542   \stex_reactivate_macro:N \premise
5543   \stex_reactivate_macro:N \definens
5544 }
5545 \str_if_empty:NTF \sparagraphid {
5546   \str_if_empty:NTF \sparagraphname {
5547     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5548       \stex_ref_new_doc_target:n {}
5549     }
5550   } {
5551     \stex_ref_new_doc_target:n {}
5552   }
5553 } {
5554   \stex_ref_new_doc_target:n \sparagraphid
5555 }
5556 \exp_args:NNx
5557 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5558   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5559     \tl_if_empty:nF{ ##1 }{
5560       \stex_get_symbol:n { ##1 }
5561       \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5562     }
5563   }
5564 }
5565 \stex_smsmode_do:
5566 \ignorespacesandpars
5567 }{
5568   \str_if_empty:NF \sparagraphname {
5569     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5570     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5571   }
5572   \stex_if_smsmode:F {
5573     \clist_set:No \l_tmpa_clist \sparagraphtype
5574     \tl_clear:N \l_tmpa_tl
5575     \clist_map_inline:Nn \l_tmpa_clist {
5576       \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{

```

```

5577         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
5578     }
5579 }
5580 \tl_if_empty:NTF \l_tmpa_tl {
5581     \__stex_statements_sparagraph_end:
5582 }{
5583     \l_tmpa_tl
5584 }
5585 \end{stex_annotate_env}
5586 }
5587 }

```

### \stexpatchparagraph

```

5588
5589 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
5590     \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5591         \tl_if_empty:NF \l_stex_sparagraph_title_tl {
5592             \titleemph{\l_stex_sparagraph_title_tl}:~
5593         }
5594     }{
5595         \titleemph{\l_stex_sparagraph_start_tl}~
5596     }
5597 }
5598 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
5599
5600 \newcommand\stexpatchparagraph[3] [] {
5601     \str_set:Nx \l_tmpa_str{ #1 }
5602     \str_if_empty:NTF \l_tmpa_str {
5603         \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
5604         \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
5605     }{
5606         \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2 }
5607         \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
5608     }
5609 }
5610
5611 \keys_define:nn { stex / inlinepara } {
5612     id .str_set_x:N = \sparagraphid ,
5613     type .str_set_x:N = \sparagraphtype ,
5614     for .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5615     from .tl_set:N = \sparagraphfrom ,
5616     to .tl_set:N = \sparagraphto ,
5617     name .str_set:N = \sparagraphname
5618 }
5619 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
5620     \tl_clear:N \sparagraphfrom
5621     \tl_clear:N \sparagraphto
5622     \str_clear:N \sparagraphid
5623     \str_clear:N \sparagraphtype
5624     \clist_clear:N \l__stex_statements_sparagraph_for_clist
5625     \str_clear:N \sparagraphname
5626     \keys_set:nn { stex / inlinepara }{ #1 }
5627 }
5628 \NewDocumentCommand \inlinepara { 0{} m } {

```

```

5629 \begingroup
5630 \__stex_statements_inlinepara_args:n{ #1 }
5631 \clist_set:No \l_tmpa_clist \sparagraphtype
5632 \str_if_empty:NTF \sparagraphid {
5633   \str_if_empty:NTF \sparagraphname {
5634     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5635       \stex_ref_new_doc_target:n {}
5636     }
5637   } {
5638     \stex_ref_new_doc_target:n {}
5639   }
5640 } {
5641   \stex_ref_new_doc_target:n \sparagraphid
5642 }
5643 \stex_if_smsmode:TF{
5644   \str_if_empty:NF \sparagraphname {
5645     \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
5646   \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5647 }
5648 }{
5649   \seq_clear:N \l_tmpa_seq
5650   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5651     \tl_if_empty:nF{ ##1 }{
5652       \stex_get_symbol:n { ##1 }
5653       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5654         \l_stex_get_symbol_uri_str
5655       }
5656     }
5657   }
5658   \exp_args:Nnx
5659   \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
5660     \str_if_empty:NF \sparagraphtype {
5661       \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
5662     }
5663     \str_if_empty:NF \sparagraphfrom {
5664       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5665     }
5666     \str_if_empty:NF \sparagraphto {
5667       \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5668     }
5669     \str_if_empty:NF \sparagraphname {
5670       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
5671     \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
5672     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5673   }
5674   \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5675     \clist_map_inline:Nn \l_tmpa_seq {
5676       \stex_ref_new_sym_target:n {##1}
5677     }
5678   }
5679   #2
5680 }
5681 }
5682 \endgroup

```

```

5683 \stex_smsmode_do:
5684 }
5685
(End definition for \stexpatchparagraph. This function is documented on page 42.)
5686 \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>

```
5687 <*package>
5688 <@@=stex_sproof>
5689
5690 %%%%%%%%%%% sproof.dtx %%%%%%%%%%%
5691
```

### 33.2 Proofs

We first define some keys for the proof environment.

```
5692 \keys_define:nn { stex / spf } {
5693   id          .str_set_x:N = \spfid,
5694   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,
5695   from        .tl_set:N    = \l__stex_sproof_spf_from_tl ,
5696   proofend    .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
5697   type        .str_set_x:N = \spftype,
5698   title       .tl_set:N    = \spftitle,
5699   continues   .tl_set:N    = \l__stex_sproof_spf_continues_tl,
5700   functions   .tl_set:N    = \l__stex_sproof_spf_functions_tl,
5701   method      .tl_set:N    = \l__stex_sproof_spf_method_tl
5702 }
5703 \cs_new_protected:Nn \l__stex_sproof_spf_args:n {
5704   \str_clear:N \spfid
5705   \tl_clear:N \l__stex_sproof_spf_for_tl
5706   \tl_clear:N \l__stex_sproof_spf_from_tl
5707   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
5708   \str_clear:N \spftype
5709   \tl_clear:N \spftitle
5710   \tl_clear:N \l__stex_sproof_spf_continues_tl
5711   \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

```

5712 \tl_clear:N \l__stex_sproof_spf_method_tl
5713 \bool_set_false:N \l__stex_sproof_inc_counter_bool
5714 \keys_set:nn { stex / spf }{ #1 }
5715 }

```

`\c__stex_sproof_flow_str` We define this macro, so that we can test whether the `display` key has the value `flow`

```

5716 \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.

```

5717 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
5718 \cs_new_protected:Npn \sproofnumber {
5719   \int_set:Nn \l_tmpa_int {1}
5720   \bool_while_do:nn {
5721     \int_compare_p:nNn {
5722       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5723     } > 0
5724   }{
5725     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
5726     \int_incr:N \l_tmpa_int
5727   }
5728 }
5729 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
5730   \int_set:Nn \l_tmpa_int {1}
5731   \bool_while_do:nn {
5732     \int_compare_p:nNn {
5733       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5734     } > 0
5735   }{
5736     \int_incr:N \l_tmpa_int
5737   }
5738   \int_compare:nNnF \l_tmpa_int = 1 {
5739     \int_decr:N \l_tmpa_int
5740   }
5741   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
5742     \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



```

5787 }
5788 \clist_if_in:NnT \l_tmpa_clist {finnish}{
5789   \input{sproof-finnish.ldf}
5790 }
5791 \clist_if_in:NnT \l_tmpa_clist {french}{
5792   \input{sproof-french.ldf}
5793 }
5794 \clist_if_in:NnT \l_tmpa_clist {russian}{
5795   \input{sproof-russian.ldf}
5796 }
5797 \makeatother
5798 }{}
5799 }

```

spfsketch

```

5800 \newcommand\spfsketch[2] [] {
5801   \beginingroup
5802   \let \premise \stex_proof_premise:
5803   \__stex_sproof_spf_args:n{#1}
5804   \stex_if_smsmode:TF {
5805     \str_if_empty:NF \spfid {
5806       \stex_ref_new_doc_target:n \spfid
5807     }
5808   }{
5809     \seq_clear:N \l_tmpa_seq
5810     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5811       \tl_if_empty:nF{ ##1 }{
5812         \stex_get_symbol:n { ##1 }
5813         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5814           \l_stex_get_symbol_uri_str
5815         }
5816       }
5817     }
5818     \exp_args:Nnx
5819     \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
5820       \str_if_empty:NF \spftype {
5821         \stex_annotate_invisible:nnn{type}{\spftype}{-}
5822       }
5823       \clist_set:No \l_tmpa_clist \spftype
5824       \tl_set:Nn \l_tmpa_tl {
5825         \titleemph{
5826           \tl_if_empty:NTF \spftitle {
5827             \spf@proofsketch@kw
5828           }{
5829             \spftitle
5830           }
5831         }::~
5832       }
5833       \clist_map_inline:Nn \l_tmpa_clist {
5834         \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5835           \tl_clear:N \l_tmpa_tl
5836         }
5837       }
5838       \str_if_empty:NF \spfid {

```



```

5839         \stex_ref_new_doc_target:n \spfid
5840     }
5841     \l_tmpa_tl #2 \sproofend
5842 }
5843 }
5844 \endgroup
5845 \stex_smsmode_do:
5846 }
5847

```

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

EdN:9  
EdN:10 **spfeq** This is very similar to `\spfsketch`, but uses a computation array<sup>910</sup>

```

5848 \newenvironment{spfeq}[2][]{
5849   \__stex_sproof_spf_args:n{#1}
5850   \let \premise \stex_proof_premise:
5851   \stex_if_smsmode:TF {
5852     \str_if_empty:NF \spfid {
5853       \stex_ref_new_doc_target:n \spfid
5854     }
5855   }{
5856     \seq_clear:N \l_tmpa_seq
5857     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5858       \tl_if_empty:NF{ ##1 }{
5859         \stex_get_symbol:n { ##1 }
5860         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5861           \l_stex_get_symbol_uri_str
5862         }
5863       }
5864     }
5865     \exp_args:Nnnx
5866     \begin{stex_annotate_env}{spfeq}{\seq_use:Nn \l_tmpa_seq {,}}
5867     \str_if_empty:NF \spftype {
5868       \stex_annotate_invisible:nnn{type}{\spftype}{ }
5869     }
5870
5871     \clist_set:No \l_tmpa_clist \spftype
5872     \tl_clear:N \l_tmpa_tl
5873     \clist_map_inline:Nn \l_tmpa_clist {
5874       \tl_if_exist:cT {__stex_sproof_spfeq_##1_start:}{
5875         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_start:}}
5876       }
5877       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5878         \tl_set:Nn \l_tmpa_tl {\use:n{ }}
5879       }
5880     }
5881     \tl_if_empty:NTF \l_tmpa_tl {
5882       \__stex_sproof_spfeq_start:
5883     }{
5884       \l_tmpa_tl
5885     }{-#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

```

5886 \str_if_empty:NF \spfid {
5887 \stex_ref_new_doc_target:n \spfid
5888 }
5889 \begin{displaymath}\begin{array}{rc1l}
5890 }
5891 \stex_smsmode_do:
5892 }{
5893 \stex_if_smsmode:F {
5894 \end{array}\end{displaymath}
5895 \clist_set:No \l_tmpa_clist \spftype
5896 \tl_clear:N \l_tmpa_tl
5897 \clist_map_inline:Nn \l_tmpa_clist {
5898 \tl_if_exist:cT {__stex_sproof_spfeq_##1_end:}{
5899 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_end:}}
5900 }
5901 }
5902 \tl_if_empty:NTF \l_tmpa_tl {
5903 \__stex_sproof_spfeq_end:
5904 }{
5905 \l_tmpa_tl
5906 }
5907 \end{stex_annotate_env}
5908 }
5909 }
5910
5911 \cs_new_protected:Nn \__stex_sproof_spfeq_start: {
5912 \titleemph{
5913 \tl_if_empty:NTF \spftitle {
5914 \spf@proof@kw
5915 }{
5916 \spftitle
5917 }
5918 }:
5919 }
5920 \cs_new_protected:Nn \__stex_sproof_spfeq_end: {\sproofend}
5921
5922 \newcommand\stexpatchspfeq[3] [] {
5923 \str_set:Nx \l_tmpa_str{ #1 }
5924 \str_if_empty:NTF \l_tmpa_str {
5925 \tl_set:Nn \__stex_sproof_spfeq_start: { #2 }
5926 \tl_set:Nn \__stex_sproof_spfeq_end: { #3 }
5927 }{
5928 \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_start:\endcsname{ #2 }
5929 \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_end:\endcsname{ #3 }
5930 }
5931 }
5932

```

(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.

```

5933 \newenvironment{sproof}[2] []{

```

```

5934 \let \premise \stex_proof_premise:
5935 \intarray_gzero:N \l__stex_sproof_counter_intarray
5936 \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
5937 \__stex_sproof_spf_args:n{#1}
5938 \stex_if_smsmode:TF {
5939   \str_if_empty:NF \spfid {
5940     \stex_ref_new_doc_target:n \spfid
5941   }
5942 }{
5943   \seq_clear:N \l_tmpa_seq
5944   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5945     \tl_if_empty:NF{ ##1 }{
5946       \stex_get_symbol:n { ##1 }
5947       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5948         \l_stex_get_symbol_uri_str
5949       }
5950     }
5951   }
5952   \exp_args:Nnnx
5953   \begin{stex_annotate_env}{sproof}{\seq_use:Nn \l_tmpa_seq {},}}
5954   \str_if_empty:NF \spftype {
5955     \stex_annotate_invisible:nnn{type}{\spftype}{}
5956   }
5957
5958   \clist_set:No \l_tmpa_clist \spftype
5959   \tl_clear:N \l_tmpa_tl
5960   \clist_map_inline:Nn \l_tmpa_clist {
5961     \tl_if_exist:cT {\__stex_sproof_sproof_##1_start:}{
5962       \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_sproof_sproof_##1_start:}}
5963     }
5964     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5965       \tl_set:Nn \l_tmpa_tl {\use:n{}}
5966     }
5967   }
5968   \tl_if_empty:NTF \l_tmpa_tl {
5969     \__stex_sproof_sproof_start:
5970   }{
5971     \l_tmpa_tl
5972   }{~#2}
5973   \str_if_empty:NF \spfid {
5974     \stex_ref_new_doc_target:n \spfid
5975   }
5976   \begin{description}
5977 }
5978 \stex_smsmode_do:
5979 }{
5980 \stex_if_smsmode:F{
5981   \end{description}
5982   \clist_set:No \l_tmpa_clist \spftype
5983   \tl_clear:N \l_tmpa_tl
5984   \clist_map_inline:Nn \l_tmpa_clist {
5985     \tl_if_exist:cT {\__stex_sproof_sproof_##1_end:}{
5986       \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_sproof_sproof_##1_end:}}
5987     }

```

```

5988     }
5989     \tl_if_empty:NTF \l_tmpa_tl {
5990       \__stex_sproof_sproof_end:
5991     }{
5992       \l_tmpa_tl
5993     }
5994     \end{stex_annotate_env}
5995   }
5996 }
5997
5998 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
5999   \par\noindent\titleemph{
6000     \tl_if_empty:NTF \spftype {
6001       \spf@proof@kw
6002     }{
6003       \spftype
6004     }
6005   }:
6006 }
6007 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
6008
6009 \newcommand\stexpatchproof[3] [] {
6010   \str_set:Nx \l_tmpa_str{ #1 }
6011   \str_if_empty:NTF \l_tmpa_str {
6012     \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
6013     \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
6014   }{
6015     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
6016     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
6017   }
6018 }

```

\spfidea

```

6019 \newcommand\spfidea[2] []{
6020   \__stex_sproof_spf_args:n{#1}
6021   \titleemph{
6022     \tl_if_empty:NTF \spftype {Proof~Idea}{
6023       \spftype
6024     }:
6025   }~#2
6026   \sproofend
6027 }

```

(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

```

6028 \newenvironment{spfstep}[1] []{
6029   \__stex_sproof_spf_args:n{#1}
6030   \stex_if_smsmode:TF {

```

```

6031 \str_if_empty:NF \spfid {
6032   \stex_ref_new_doc_target:n \spfid
6033 }
6034 }{
6035   \@in@omtexttrue
6036   \seq_clear:N \l_tmpa_seq
6037   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6038     \tl_if_empty:nF{ ##1 }{
6039       \stex_get_symbol:n { ##1 }
6040       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6041         \l_stex_get_symbol_uri_str
6042       }
6043     }
6044   }
6045   \exp_args:Nnnx
6046   \begin{stex_annotate_env}{spfstep}{\seq_use:Nn \l_tmpa_seq {,}}
6047   \str_if_empty:NF \spftype {
6048     \stex_annotate_invisible:nnn{type}{\spftype}{}
6049   }
6050   \clist_set:No \l_tmpa_clist \spftype
6051   \tl_set:Nn \l_tmpa_tl {
6052     \item[\sproofnumber]
6053     \bool_set_true:N \l__stex_sproof_inc_counter_bool
6054   }
6055   \clist_map_inline:Nn \l_tmpa_clist {
6056     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6057       \tl_clear:N \l_tmpa_tl
6058     }
6059   }
6060   \l_tmpa_tl
6061   \tl_if_empty:NF \spftitle {
6062     {(\titleemph{\spftitle})\enspace}
6063   }
6064   \str_if_empty:NF \spfid {
6065     \stex_ref_new_doc_target:n \spfid
6066   }
6067 }
6068 \stex_smsmode_do:
6069 \ignorespacesandpars
6070 }{
6071   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6072     \__stex_sproof_inc_counter:
6073   }
6074   \stex_if_smsmode:F {
6075     \end{stex_annotate_env}
6076   }
6077 }

```

sproofcomment

```

6078 \newenvironment{sproofcomment}[1][]{
6079   \__stex_sproof_spf_args:n{#1}
6080   \clist_set:No \l_tmpa_clist \spftype
6081   \tl_set:Nn \l_tmpa_tl {
6082     \item[\sproofnumber]

```

```

6083 \bool_set_true:N \l__stex_sproof_inc_counter_bool
6084 }
6085 \clist_map_inline:Nn \l_tmpa_clist {
6086   \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6087     \tl_clear:N \l_tmpa_tl
6088   }
6089 }
6090 \l_tmpa_tl
6091 }{
6092   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6093     \__stex_sproof_inc_counter:
6094   }
6095 }

```

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.

```

6096 \newenvironment{subproof}[2][]{
6097   \__stex_sproof_spf_args:n{#1}
6098   \stex_if_smsmode:TF{
6099     \str_if_empty:NF \spfid {
6100       \stex_ref_new_doc_target:n \spfid
6101     }
6102   }{
6103     \seq_clear:N \l_tmpa_seq
6104     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6105       \tl_if_empty:nF{ ##1 }{
6106         \stex_get_symbol:n { ##1 }
6107         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6108           \l_stex_get_symbol_uri_str
6109         }
6110       }
6111     }
6112     \exp_args:Nnnx
6113     \begin{stex_annotate_env}{subproof}{\seq_use:Nn \l_tmpa_seq {,}}
6114     \str_if_empty:NF \spftype {
6115       \stex_annotate_invisible:nnn{type}{\spftype}{\}
6116     }
6117
6118     \clist_set:No \l_tmpa_clist \spftype
6119     \tl_set:Nn \l_tmpa_tl {
6120       \item[\sproofnumber]
6121       \bool_set_true:N \l__stex_sproof_inc_counter_bool
6122     }
6123     \clist_map_inline:Nn \l_tmpa_clist {
6124       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6125         \tl_clear:N \l_tmpa_tl
6126       }
6127     }
6128     \l_tmpa_tl
6129     \tl_if_empty:NF \spftitle {
6130       {(\titleemph{\spftitle})\enspace}
6131     }

```

```

6132     {~#2}
6133     \str_if_empty:NF \spfid {
6134       \stex_ref_new_doc_target:n \spfid
6135     }
6136   }
6137   \__stex_sproof_add_counter:
6138   \stex_smsmode_do:
6139 }{
6140   \__stex_sproof_remove_counter:
6141   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6142     \__stex_sproof_inc_counter:
6143   }
6144   \stex_if_smsmode:F{
6145     \end{stex_annotate_env}
6146   }
6147 }

```

**spfcases** In the **pfcases** environment, the start text is displayed as the first comment of the proof.

```

6148 \newenvironment{spfcases}[2][]{
6149   \tl_if_empty:nTF{#1}{
6150     \begin{subproof}[method=by-cases]{#2}
6151   }{
6152     \begin{subproof}[#1,method=by-cases]{#2}
6153   }
6154 }{
6155   \end{subproof}
6156 }

```

**spfcase** In the **pfcase** environment, the start text is displayed specification of the case after the **\item**

```

6157 \newenvironment{spfcase}[2][]{
6158   \__stex_sproof_spf_args:n{#1}
6159   \stex_if_smsmode:TF {
6160     \str_if_empty:NF \spfid {
6161       \stex_ref_new_doc_target:n \spfid
6162     }
6163   }{
6164     \seq_clear:N \l_tmpa_seq
6165     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6166       \tl_if_empty:nF{ ##1 }{
6167         \stex_get_symbol:n { ##1 }
6168         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6169           \l_stex_get_symbol_uri_str
6170         }
6171       }
6172     }
6173     \exp_args:Nnnx
6174     \begin{stex_annotate_env}{spfcase}{\seq_use:Nn \l_tmpa_seq {,}}
6175     \str_if_empty:NF \spftype {
6176       \stex_annotate_invisible:nnn{type}{\spftype}{}}
6177   }
6178   \clist_set:Nn \l_tmpa_clist \spftype
6179   \tl_set:Nn \l_tmpa_tl {
6180     \item[\sproofnumber]

```

```

6181     \bool_set_true:N \l__stex_sproof_inc_counter_bool
6182   }
6183   \clist_map_inline:Nn \l_tmpa_clist {
6184     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6185       \tl_clear:N \l_tmpa_tl
6186     }
6187   }
6188   \l_tmpa_tl
6189   \tl_if_empty:nF{#2}{
6190     \titleemph{#2}:~
6191   }
6192 }
6193 \__stex_sproof_add_counter:
6194 \stex_smsmode_do:
6195 ){
6196   \__stex_sproof_remove_counter:
6197   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6198     \__stex_sproof_inc_counter:
6199   }
6200   \stex_if_smsmode:F{
6201     \clist_set:No \l_tmpa_clist \spftype
6202     \tl_set:Nn \l_tmpa_tl{\sproofend}
6203     \clist_map_inline:Nn \l_tmpa_clist {
6204       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6205         \tl_clear:N \l_tmpa_tl
6206       }
6207     }
6208     \l_tmpa_tl
6209     \end{stex_annotate_env}
6210   }
6211 }

```

**spfcase** similar to **spfcase**, takes a third argument.

```

6212 \newcommand\spfcasesketch[3][]{
6213   \begin{spfcase}[#1]{#2}#3\end{spfcase}
6214 }

```

### 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.

```

6215 \keys_define:nn { stex / just }{
6216   id      .str_set:x:N = \l__stex_sproof_just_id_str,
6217   method  .tl_set:N    = \l__stex_sproof_just_method_tl,
6218   premises .tl_set:N    = \l__stex_sproof_just_premises_tl,
6219   args     .tl_set:N    = \l__stex_sproof_just_args_tl
6220 }

```

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**

```
6221 \newenvironment{justification}[1] [] {}{}
```

**\premise**

```
6222 \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.

```
6223 \newcommand\justarg[2] [] {#2}
```

```
6224 \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

```
6225 <*package>
6226
6227 %%%%%%%%%% others.dtx %%%%%%%%%%
6228
6229 <@@=stex_others>
        Warnings and error messages
6230 % None

\MSC Math subject classifier

6231 \NewDocumentCommand \MSC {m} {
6232 % TODO
6233 }

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

6234 \@ifpackageloaded{tikzinput}{
6235 \RequirePackage{stex-tikzinput}
6236 }{}

6237 </package>
```

## Chapter 35

# STEX -Metatheory Implementation

```
6238 <*package>
6239 <@@=stex_modules>
6240
6241 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
6242
6243 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
6244 \begingroup
6245 \stex_module_setup:nn{
6246   ns=\c_stex_metatheory_ns_str,
6247   meta=NONE
6248 }{Metatheory}
6249 \stex_reactivate_macro:N \symdecl
6250 \stex_reactivate_macro:N \notation
6251 \stex_reactivate_macro:N \symdef
6252 \ExplSyntaxOff
6253 \csname stex_suppress_html:n\endcsname{
6254   % is-a (a:A, a \in A, a is an A, etc.)
6255   \symdecl{isa}[args=ai]
6256   \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
6257   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
6258   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
6259
6260   % bind (\forall, \Pi, \lambda etc.)
6261   \symdecl{bind}[args=Bi]
6262   \notation{bind}[forall]{\comp\forall #1.;#2}{##1 \comp, ##2}
6263   \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6264   \notation{bind}[deffun]{\comp( #1 \comp{ }\to\; } #2}{##1 \comp, ##2}
6265
6266   % implicit bind
6267   \symdef{implicitbind}[args=Bi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6268
6269   % dummy variable
6270   \symdecl{dummyvar}
6271   \notation{dummyvar}[underscore]{\comp\_}
6272   \notation{dummyvar}[dot]{\comp\cdot}
```

```

6273 \notation{dummyvar}[dash]{\comp{\rm --}}
6274
6275 %fromto (function space, Hom-set, implication etc.)
6276 \symdecl{fromto}[args=ai]
6277 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
6278 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
6279
6280 % mapto (lambda etc.)
6281 \symdecl{mapto}[args=Bi]
6282 \notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
6283 \notation{mapto}[lambda]{\comp\lambda #1 \comp.; #2}{#1 \comp, #2}
6284 \notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.; #2}{#1 \comp, #2}
6285
6286 % function/operator application
6287 \symdecl{apply}[args=ia]
6288 \notation{apply}[prec=0;0x\infprec,parens]{#1 \comp( #2 \comp)}{##1 \comp, ##2}
6289 \notation{apply}[prec=0;0x\infprec,lambda]{#1 \; #2 }{##1 \; ; ##2}
6290
6291 % collection of propositions/booleans/truth values
6292 \symdecl{prop}[name=proposition]
6293 \notation{prop}[prop]{\comp{\rm prop}}
6294 \notation{prop}[BOOL]{\comp{\rm BOOL}}
6295
6296 \symdecl{judgmentholds}[args=1]
6297 \notation{judgmentholds}[vdash,op=\vdash]{\comp\vdash\; ; #1}
6298
6299 % sequences
6300 \symdecl{seqtype}[args=1]
6301 \notation{seqtype}[kleene]{#1^{\comp\ast}}
6302
6303 \symdecl{seqexpr}[args=a]
6304 \notation{seqexpr}[angle,prec=nobrackets]{\comp\angle #1\comp\rangle}{##1\comp,##2}
6305
6306 \symdef{sequence-index}[args=2,li,prec=nobrackets]{#{#1}_{#2}}
6307 \notation{sequence-index}[ui,prec=nobrackets]{#{#1}^{#2}}
6308
6309 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{,\ellipses}}{##1\comp,##2}
6310 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{,\ellipses,}#2}{##1\comp,##2}
6311 \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{,\ellipses,}#2\comp{,\ellipses,}#3}
6312
6313 % letin (''let'', local definitions, variable substitution)
6314 \symdecl{letin}[args=bii]
6315 \notation{letin}[let]{\comp{\rm let}}\;#1\comp{=}\;#2\; \comp{\rm in}}\;#3}
6316 \notation{letin}[subst]{#3 \comp[ #1 \comp/ #2 \comp]}
6317 \notation{letin}[frac]{#3 \comp[ \frac{#2}{#1} \comp]}
6318
6319 % structures
6320 \symdecl*{module-type}[args=1]
6321 \notation{module-type}{\comp{\mathtt{MOD}}} #1}
6322 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
6323 \notation{mathstruct}[angle,prec=nobrackets]{\comp\angle #1 \comp\rangle}{##1 \comp, ##2}
6324
6325 % objects
6326 \symdecl{object}

```

```

6327 \notation{object}{\comp{\mathtt{OBJECT}}}
6328
6329 }
6330 \ExplSyntaxOn
6331 \stex_add_to_current_module:n{
6332   \let\nappa\apply
6333   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
6334   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
6335   \def\livar{\csname sequence-index\endcsname[li]}
6336   \def\uivar{\csname sequence-index\endcsname[ui]}
6337   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
6338   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
6339   \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
6340 }
6341 \__stex_modules_end_module:
6342 \endgroup
6343 \</package>

```

## Chapter 36

# Tikzinput Implementation

```
6344 <*package>
6345
6346 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
6347
6348 \ProvidesExplPackage{tikzinput}{2022/02/26}{3.0.1}{tikzinput package}
6349 \RequirePackage{l3keys2e}
6350
6351 \keys_define:nn { tikzinput } {
6352   image .bool_set:N = \c_tikzinput_image_bool,
6353   image .default:n = false ,
6354   unknown .code:n = {}
6355 }
6356
6357 \ProcessKeysOptions { tikzinput }
6358
6359 \bool_if:NTF \c_tikzinput_image_bool {
6360   \RequirePackage{graphicx}
6361
6362   \providecommand\usetikzlibrary[]{}
6363   \newcommand\tikzinput[2] []{\includegraphics[#1]{#2}}
6364 }{
6365   \RequirePackage{tikz}
6366   \RequirePackage{standalone}
6367
6368   \newcommand \tikzinput [2] [] {
6369     \setkeys{Gin}{#1}
6370     \ifx \Gin@ewidth \Gin@exclamation
6371       \ifx \Gin@eheight \Gin@exclamation
6372         \input { #2 }
6373       \else
6374         \resizebox{!}{ \Gin@eheight }{
6375           \input { #2 }
6376         }
6377       \fi
6378     \else
6379       \ifx \Gin@eheight \Gin@exclamation
6380         \resizebox{ \Gin@ewidth }{!}{
6381           \input { #2 }
```

```

6382     }
6383     \else
6384         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
6385             \input { #2 }
6386         }
6387     \fi
6388 \fi
6389 }
6390 }
6391
6392 \newcommand \ctikzinput [2] [] {
6393     \begin{center}
6394         \tikzinput [1] {#2}
6395     \end{center}
6396 }
6397
6398 \@ifpackageloaded{stex}{
6399     \RequirePackage{stex-tikzinput}
6400 }{}
6401
6402 </package>
6403 <*stex>
6404 \ProvidesExplPackage{stex-tikzinput}{2022/02/26}{3.0.1}{stex-tikzinput}
6405 \RequirePackage{stex}
6406 \RequirePackage{tikzinput}
6407
6408 \newcommand\mhtikzinput [2] [] {%
6409     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
6410     \stex_in_repository:nn\Gin@mhrepos{
6411         \tikzinput [1]{\mhpath{##1}{#2}}
6412     }
6413 }
6414 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [1] {#2}\end{center}}
6415 </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

### 37.1 The document-structure Class

The functionality is spread over the `document-structure` class and package. The class provides the `document` environment and the `document-structure` element corresponds to it, whereas the package provides the concrete functionality.

```
6416 \*cls)
6417 \@@=document_structure)
6418 \ProvidesExplClass{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure Class}
6419 \RequirePackage{13keys2e}
```

### 37.2 Class Options

To initialize the `document-structure` class, we declare and process the necessary options using the `kvoptions` package for key/value options handling. For `omdoc.cls` this is quite simple. We have options `report` and `book`, which set the `\omdoc@cls@class` macro and pass on the macro to `omdoc.sty` for further processing.

`\omdoc@cls@class`

```
6420 \keys_define:nn{ document-structure / pkg }{
6421   class      .str_set_x:N = \c_document_structure_class_str,
6422   minimal    .bool_set:N = \c_document_structure_minimal_bool,
6423   report     .code:n      = {
6424     \ClassWarning{document-structure}{the option 'report' is deprecated, use 'class=report',
6425     \str_set:Nn \c_document_structure_class_str {report}
6426   },
6427   book       .code:n      = {
6428     \ClassWarning{document-structure}{the option 'book' is deprecated, use 'class=book', ins
6429     \str_set:Nn \c_document_structure_class_str {book}
6430   },
6431   bookpart   .code:n      = {
6432     \ClassWarning{document-structure}{the option 'bookpart' is deprecated, use 'class=book,t
6433     \str_set:Nn \c_document_structure_class_str {book}
6434     \str_set:Nn \c_document_structure_topsect_str {chapter}
6435   },
```



```

6436 docopt      .str_set_x:N = \c_document_structure_docopt_str,
6437 unknown     .code:n      = {
6438   \PassOptionsToPackage{ \CurrentOption }{ document-structure }
6439 }
6440 }
6441 \ProcessKeysOptions{ document-structure / pkg }
6442 \str_if_empty:NT \c_document_structure_class_str {
6443   \str_set:Nn \c_document_structure_class_str {article}
6444 }
6445 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
6446   {\c_document_structure_class_str}
6447

```

### 37.3 Beefing up the document environment

Now, – unless the option `minimal` is defined – we include the `stex` package

```

6448 \RequirePackage{document-structure}
6449 \bool_if:NF \c_document_structure_minimal_bool {

```

And define the environments we need. The top-level one is the `document` environment, which we redefined so that we can provide keyval arguments.

**document** For the moment we do not use them on the L<sup>A</sup>T<sub>E</sub>X level, but the document identifier is picked up by L<sup>A</sup>T<sub>E</sub>XML.<sup>12</sup>

```

6450 \keys_define:nn { document-structure / document }{
6451   id .str_set_x:N = \c_document_structure_document_id_str
6452 }
6453 \let\__document_structure_orig_document=\document
6454 \renewcommand{\document}[1][]{
6455   \keys_set:nn{ document-structure / document }{ #1 }
6456   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
6457   \__document_structure_orig_document
6458 }

```

Finally, we end the test for the `minimal` option.

```

6459 }
6460 \</cls>

```

### 37.4 Implementation: document-structure Package

```

6461 \<*package>
6462 \ProvidesExplPackage{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure}
6463 \RequirePackage{l3keys2e}

```

### 37.5 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>12</sup>EDNOTE: faking documentkeys for now. @HANG, please implement

```

6464
6465 \keys_define:nn{ document-structure / pkg }{
6466   class      .str_set_x:N = \c_document_structure_class_str,
6467   topsect    .str_set_x:N = \c_document_structure_topsect_str,
6468   % showignores .bool_set:N = \c_document_structure_showignores_bool,
6469 }
6470 \ProcessKeysOptions{ document-structure / pkg }
6471 \str_if_empty:NT \c_document_structure_class_str {
6472   \str_set:Nn \c_document_structure_class_str {article}
6473 }
6474 \str_if_empty:NT \c_document_structure_topsect_str {
6475   \str_set:Nn \c_document_structure_topsect_str {section}
6476 }

```

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

```

6477 \RequirePackage{xspace}
6478 \RequirePackage{comment}
6479 \AddToHook{begindocument}{
6480   \ltx@ifpackageloaded{babel}{
6481     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6482     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6483       \makeatletter\input{document-structure-ngerman.ldf}\makeatother
6484     }
6485   }{}
6486 }

```

`\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.

```

6487 \int_new:N \l_document_structure_section_level_int
6488 \str_case:VnF \c_document_structure_topsect_str {
6489   {part}}{
6490     \int_set:Nn \l_document_structure_section_level_int {0}
6491   }
6492   {chapter}}{
6493     \int_set:Nn \l_document_structure_section_level_int {1}
6494   }
6495 }{
6496   \str_case:VnF \c_document_structure_class_str {
6497     {book}}{
6498       \int_set:Nn \l_document_structure_section_level_int {0}
6499     }
6500     {report}}{
6501       \int_set:Nn \l_document_structure_section_level_int {0}
6502     }
6503   }{
6504     \int_set:Nn \l_document_structure_section_level_int {2}
6505   }
6506 }

```

## 37.6 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>13</sup>

EdN:13

```
6507 \def\current@section@level{document}%
6508 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
6509 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
6510 \cs_new_protected:Npn \skipomgroup {
6511   \ifcase\l_document_structure_section_level_int
6512   \or\stepcounter{part}
6513   \or\stepcounter{chapter}
6514   \or\stepcounter{section}
6515   \or\stepcounter{subsection}
6516   \or\stepcounter{subsubsection}
6517   \or\stepcounter{paragraph}
6518   \or\stepcounter{subparagraph}
6519   \fi
6520 }
```

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

`blindfragment`

```
6521 \newcommand\at@begin@blindomgroup[1]{%
6522 \newenvironment{blindfragment}
6523 {
6524   \int_incr:N\l_document_structure_section_level_int
6525   \at@begin@blindomgroup\l_document_structure_section_level_int
6526 }{}}
```

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

```
6527 \newcommand\omgroup@nonum[2]{
6528   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
6529   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
6530 }
```

*(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 `rdfmata` package has been loaded. In the end we call `\sref@label@id` to enable crossreferencing.

```
6531 \newcommand\omgroup@num[2]{
```

<sup>13</sup>EDNOTE: MK: we may have to experiment with the more powerful uppercasing macro from `mfirstuc.sty` once we internationalize.

```

6532 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
6533   \@nameuse{#1}{#2}
6534 }{
6535   \cs_if_exist:NTF\rdfmata@sectioning{
6536     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
6537   }{
6538     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
6539   }
6540 }
6541 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
6542 }

```

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

**sfragment**

```

6543 \keys_define:nn { document-structure / omgroup }{
6544   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
6545   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
6546   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
6547   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
6548   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
6549   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
6550   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
6551   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
6552   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
6553   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
6554 }
6555 \cs_new_protected:Nn \__document_structure_omgroup_args:n {
6556   \str_clear:N \l__document_structure_omgroup_id_str
6557   \str_clear:N \l__document_structure_omgroup_date_str
6558   \clist_clear:N \l__document_structure_omgroup_creators_clist
6559   \clist_clear:N \l__document_structure_omgroup_contributors_clist
6560   \tl_clear:N \l__document_structure_omgroup_srccite_tl
6561   \tl_clear:N \l__document_structure_omgroup_type_tl
6562   \tl_clear:N \l__document_structure_omgroup_short_tl
6563   \tl_clear:N \l__document_structure_omgroup_display_tl
6564   \tl_clear:N \l__document_structure_omgroup_intro_tl
6565   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
6566   \keys_set:nn { document-structure / omgroup } { #1 }
6567 }

```

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 omgroup, i.e. after the section heading.

```

6568 \newif\if@mainmatter\@mainmattertrue
6569 \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.

```

6570 \keys_define:nn { document-structure / sectioning }{
6571   name .str_set_x:N = \l__document_structure_sect_name_str ,
6572   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
6573   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
6574   clear .default:n = {true} ,
6575   num .bool_set:N = \l__document_structure_sect_num_bool ,

```

```

6576   num      .default:n      = {true}
6577 }
6578 \cs_new_protected:Nn \__document_structure_sect_args:n {
6579   \str_clear:N \l__document_structure_sect_name_str
6580   \str_clear:N \l__document_structure_sect_ref_str
6581   \bool_set_false:N \l__document_structure_sect_clear_bool
6582   \bool_set_false:N \l__document_structure_sect_num_bool
6583   \keys_set:nn { document-structure / sectioning } { #1 }
6584 }
6585 \newcommand\omdoc@sectioning[3][]{
6586   \__document_structure_sect_args:n {#1}
6587   \let\omdoc@sect@name\l__document_structure_sect_name_str
6588   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
6589   \if@mainmatter% numbering not overridden by frontmatter, etc.
6590     \bool_if:NTF \l__document_structure_sect_num_bool {
6591       \omgroup@num{#2}{#3}
6592     }{
6593       \omgroup@nonum{#2}{#3}
6594     }
6595     \def\current@section@level{\omdoc@sect@name}
6596   \else
6597     \omgroup@nonum{#2}{#3}
6598   \fi
6599 }% 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.

```

6600 \newcommand\omgroup@redefine@addtocontents[1]{%
6601 %\edef\__document_structureimport{#1}%
6602 %\@for\@I:=\__document_structureimport\do{%
6603 %\edef\@path{\csname module@\@I @path\endcsname}%
6604 %\@ifundefined{tf@toc}\relax%
6605 %   {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
6606 %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
6607 %\def\addcontentsline##1##2##3{%
6608 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6609 %\else% hyperref.sty not loaded
6610 %\def\addcontentsline##1##2##3{%
6611 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6612 %\fi
6613 }% 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.

```

6614 \newenvironment{sfragment}[2][]{% keys, title
6615 {
6616   \__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.

```

6617   \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
6618     \omgroup@redefine@addtocontents{
6619       \@ifundefined{module@id}\used@modules%

```

```

6620     %{\@ifundefined{module@}\module@id @path}{\used@modules}\module@id}
6621   }
6622 }

```

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

```

6623 \int_incr:N\l_document_structure_section_level_int
6624 \ifcase\l_document_structure_section_level_int
6625   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
6626   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
6627   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
6628   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
6629   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
6630   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
6631   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
6632 \fi
6633 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
6634 \str_if_empty:NF \l__document_structure_omgroup_id_str {
6635   \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
6636 }
6637 }% for customization
6638 {}

```

and finally, we localize the sections

```

6639 \newcommand\omdoc@part@kw{Part}
6640 \newcommand\omdoc@chapter@kw{Chapter}
6641 \newcommand\omdoc@section@kw{Section}
6642 \newcommand\omdoc@subsection@kw{Subsection}
6643 \newcommand\omdoc@subsubsection@kw{Subsubsection}
6644 \newcommand\omdoc@paragraph@kw{paragraph}
6645 \newcommand\omdoc@subparagraph@kw{subparagraph}

```

## 37.7 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`

```

6646 \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).

```

6647 \cs_if_exist:NTF\frontmatter{
6648   \let\__document_structure_orig_frontmatter\frontmatter
6649   \let\frontmatter\relax
6650 }{
6651   \tl_set:Nn\__document_structure_orig_frontmatter{
6652     \clearpage
6653     \@mainmatterfalse
6654     \pagenumbering{roman}

```

```

6655 }
6656 }
6657 \cs_if_exist:NTF\backmatter{
6658   \let\__document_structure_orig_backmatter\backmatter
6659   \let\backmatter\relax
6660 }{
6661   \tl_set:Nn\__document_structure_orig_backmatter{
6662     \clearpage
6663     \@mainmatterfalse
6664     \pagenumbering{roman}
6665   }
6666 }

```

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.

```

6667 \newenvironment{frontmatter}{
6668   \__document_structure_orig_frontmatter
6669 }{
6670   \cs_if_exist:NTF\mainmatter{
6671     \mainmatter
6672   }{
6673     \clearpage
6674     \@mainmattertrue
6675     \pagenumbering{arabic}
6676   }
6677 }

```

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

```

6678 \newenvironment{backmatter}{
6679   \__document_structure_orig_backmatter
6680 }{
6681   \cs_if_exist:NTF\mainmatter{
6682     \mainmatter
6683   }{
6684     \clearpage
6685     \@mainmattertrue
6686     \pagenumbering{arabic}
6687   }
6688 }

```

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

```

6689 \@mainmattertrue\pagenumbering{arabic}

```

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

```

6690 \def \c__document_structure_document_str{document}
6691 \newcommand\afterprematurestop{}
6692 \def\prematurestop@endomgroup{
6693   \unless\ifx\@currenvir\c__document_structure_document_str
6694     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter\expandafter
6695     \expandafter\prematurestop@endomgroup

```

```

6696 \fi
6697 }
6698 \providecommand\prematurestop{
6699 \message{Stopping~sTeX~processing~prematurely}
6700 \prematurestop@endumgroup
6701 \afterprematurestop
6702 \end{document}
6703 }

```

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

## 37.8 Global Variables

**\setSGvar** set a global variable

```

6704 \RequirePackage{etoolbox}
6705 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

**\useSGvar** use a global variable

```

6706 \newrobustcmd\useSGvar[1]{%
6707 \@ifundefined{sTeX@Gvar@#1}
6708 {\PackageError{document-structure}
6709 {The sTeX Global variable #1 is undefined}
6710 {set it with \protect\setSGvar}}
6711 \@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.

```

6712 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
6713 \@ifundefined{sTeX@Gvar@#1}
6714 {\PackageError{document-structure}
6715 {The sTeX Global variable #1 is undefined}
6716 {set it with \protect\setSGvar}}
6717 {\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.

```
6718 \*cls)
6719 \@@=notesslides)
6720 \ProvidesExplClass{notesslides}{2022/02/28}{3.1.0}{notesslides Class}
6721 \RequirePackage{13keys2e}
6722
6723 \keys_define:nn{notesslides / cls}{
6724   class .code:n = {
6725     \PassOptionsToClass{\CurrentOption}{document-structure}
6726     \str_if_eq:nnT{#1}{book}{
6727       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
6728     }
6729     \str_if_eq:nnT{#1}{report}{
6730       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
6731     }
6732   },
6733   notes .bool_set:N = \c__notesslides_notes_bool ,
6734   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
6735   unknown .code:n = {
6736     \PassOptionsToClass{\CurrentOption}{document-structure}
6737     \PassOptionsToClass{\CurrentOption}{beamer}
6738     \PassOptionsToPackage{\CurrentOption}{notesslides}
6739   }
6740 }
6741 \ProcessKeysOptions{ notesslides / cls }
6742 \bool_if:NTF \c__notesslides_notes_bool {
6743   \PassOptionsToPackage{notes=true}{notesslides}
6744 }{
6745   \PassOptionsToPackage{notes=false}{notesslides}
6746 }
6747 \</cls)
```

now we do the same for the notesslides package.

```

6748 <*package>
6749 \ProvidesExplPackage{notesslides}{2022/02/28}{3.1.0}{notesslides Package}
6750 \RequirePackage{13keys2e}
6751
6752 \keys_define:nn{notesslides / pkg}{
6753   topsect      .str_set_x:N = \c__notesslides_topsect_str,
6754   defaulttopsect .str_set_x:N = \c__notesslides_defaulttopsec_str,
6755   notes        .bool_set:N = \c__notesslides_notes_bool ,
6756   slides       .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
6757   sectocframes .bool_set:N = \c__notesslides_sectocframes_bool ,
6758   frameimages  .bool_set:N = \c__notesslides_frameimages_bool ,
6759   fiboxed      .bool_set:N = \c__notesslides_fiboxed_bool ,
6760   nopproblems  .bool_set:N = \c__notesslides_nopproblems_bool,
6761   unknown      .code:n      = {
6762     \PassOptionsToClass{\CurrentOption}{stex}
6763     \PassOptionsToClass{\CurrentOption}{tikzinput}
6764   }
6765 }
6766 \ProcessKeysOptions{ notesslides / pkg }
6767 \newif\ifnotes
6768 \bool_if:NTF \c__notesslides_notes_bool {
6769   \notesttrue
6770 }{
6771   \notesfalse
6772 }
6773

```

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

```

6774 \str_if_empty:NTF \c__notesslides_topsect_str {
6775   \str_set_eq:NN \__notesslides_topsect \c__notesslides_defaulttopsec_str
6776 }{
6777   \str_set_eq:NN \__notesslides_topsect \c__notesslides_topsect_str
6778 }
6779 </package>

```

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

```

6780 <*cls>
6781 \bool_if:NTF \c__notesslides_notes_bool {
6782   \LoadClass{document-structure}
6783 }{
6784   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
6785   \newcounter{Item}
6786   \newcounter{paragraph}
6787   \newcounter{subparagraph}
6788   \newcounter{Hfootnote}
6789   \RequirePackage{document-structure}
6790 }

```

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

```

6791 \RequirePackage{notesslides}
6792 </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 `STEX` packages. The first batch of packages we want are loaded on `notesslides.sty`. These are the general ones, we will load the `STEX`-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.

```

6793 \*package>
6794 \bool_if:NT \c__notesslides_notes_bool {
6795   \RequirePackage{a4wide}
6796   \RequirePackage{marginnote}
6797   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
6798   \RequirePackage{mdframed}
6799   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
6800   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
6801 }
6802 \RequirePackage{stex-tikzinput}
6803 \RequirePackage{etoolbox}
6804 \RequirePackage{amssymb}
6805 \RequirePackage{amsmath}
6806 \RequirePackage{comment}
6807 \RequirePackage{textcomp}
6808 \RequirePackage{url}
6809 \RequirePackage{graphicx}
6810 \RequirePackage{pgf}

```

## 38.2 Notes and Slides

For the lecture notes cases, we also provide the `\usetheme` macro that would otherwise come from the `beamer` class. While the latter loads `beamertheme<theme>.sty`, the notes version loads `beamernotestheme<theme>.sty`.<sup>14</sup>

```

6811 \bool_if:NT \c__notesslides_notes_bool {
6812   \renewcommand\usetheme[2][\usepackage[#1]{beamernotestheme#2}]
6813 }
6814
6815
6816 \NewDocumentCommand \libusetheme {0{} m} {
6817   \bool_if:NTF \c__notesslides_notes_bool {
6818     \libusepackage[#1]{beamernotestheme#2}
6819   }{
6820     \libusepackage[#1]{beamertheme#2}
6821   }
6822 }

```

We define the sizes of slides in the notes. Somehow, we cannot get by with the same here.

```

6823 \newcounter{slide}
6824 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
6825 \newlength{\slideheight}\setlength{\slideheight}{9cm}

```

---

<sup>14</sup>EdNOTE: MK: This is not ideal, but I am not sure that I want to be able to provide the full theme functionality there.

**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.

```

6826 \bool_if:NTF \c__notesslides_notes_bool {
6827   \renewenvironment{note}{\ignorespaces}{}
6828 }{
6829   \excludecomment{note}
6830 }

```

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.

```

6831 \bool_if:NT \c__notesslides_notes_bool {
6832   \newlength{\slideframewidth}
6833   \setlength{\slideframewidth}{1.5pt}

```

**frame** We first define the keys.

```

6834 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
6835   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
6836     \bool_set_true:N #1
6837   }{
6838     \bool_set_false:N #1
6839   }
6840 }
6841 \keys_define:nn{notesslides / frame}{
6842   label .str_set_x:N = \l__notesslides_frame_label_str,
6843   allowframebreaks .code:n = {
6844     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
6845   },
6846   allowdisplaybreaks .code:n = {
6847     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
6848   },
6849   fragile .code:n = {
6850     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
6851   },
6852   shrink .code:n = {
6853     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
6854   },
6855   squeeze .code:n = {
6856     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
6857   },
6858   t .code:n = {
6859     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
6860   },
6861 }
6862 \cs_new_protected:Nn \__notesslides_frame_args:n {
6863   \str_clear:N \l__notesslides_frame_label_str
6864   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
6865   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
6866   \bool_set_true:N \l__notesslides_frame_fragile_bool
6867   \bool_set_true:N \l__notesslides_frame_shrink_bool
6868   \bool_set_true:N \l__notesslides_frame_squeeze_bool
6869   \bool_set_true:N \l__notesslides_frame_t_bool

```

```

6870 \keys_set:nn { notesslides / frame }{ #1 }
6871 }

```

We define the environment, read them, and construct the slide number and label.

```

6872 \renewenvironment{frame}[1][]{
6873   \__notesslides_frame_args:n{#1}
6874   \sffamily
6875   \stepcounter{slide}
6876   \def\@currentlabel{\theslide}
6877   \str_if_empty:NF \l__notesslides_frame_label_str {
6878     \label{\l__notesslides_frame_label_str}
6879   }

```

We redefine the `itemize` environment so that it looks more like the one in `beamer`.

```

6880 \def\itemize@level{outer}
6881 \def\itemize@outer{outer}
6882 \def\itemize@inner{inner}
6883 \renewcommand\newpage{\addtocounter{framenumber}{1}}
6884 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
6885 \renewenvironment{itemize}{
6886   \ifx\itemize@level\itemize@outer
6887     \def\itemize@label{$\rhd$}
6888   \fi
6889   \ifx\itemize@level\itemize@inner
6890     \def\itemize@label{$\scriptstyle\rhd$}
6891   \fi
6892   \begin{list}
6893     {\itemize@label}
6894     {\setlength{\labelsep}{.3em}
6895      \setlength{\labelwidth}{.5em}
6896      \setlength{\leftmargin}{1.5em}
6897     }
6898   \edef\itemize@level{\itemize@inner}
6899 }{
6900   \end{list}
6901 }

```

We create the box with the `mdframed` environment from the `equinymous` package.

```

6902 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth]
6903 }{
6904   \medskip\miko@slidelabel\end{mdframed}
6905 }

```

Now, we need to redefine the `frametitle` (we are still in course notes mode).

`\frametitle`

```

6906 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
6907 }

```

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

EdN:15

`\pause`

```

15
6908 \bool_if:NT \c__notesslides_notes_bool {
6909   \newcommand\pause{}
6910 }

```

---

<sup>15</sup>EdNOTE: MK: fake it in notes mode for now

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

**nparagraph**

```
6911 \bool_if:NTF \c__notesslides_notes_bool {
6912   \newenvironment{nparagraph}[1] [] {\begin{sparagraph}[#1]}\end{sparagraph}}
6913 }{
6914   \excludecomment{nparagraph}
6915 }
```

**nfragment**

```
6916 \bool_if:NTF \c__notesslides_notes_bool {
6917   \newenvironment{nfragment}[2] [] {\begin{sfragment}[#1]{#2}}\end{sfragment}}
6918 }{
6919   \excludecomment{nfragment}
6920 }
```

**ndefinition**

```
6921 \bool_if:NTF \c__notesslides_notes_bool {
6922   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}\end{sdefinition}}
6923 }{
6924   \excludecomment{ndefinition}
6925 }
```

**nassertion**

```
6926 \bool_if:NTF \c__notesslides_notes_bool {
6927   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}\end{sassertion}}
6928 }{
6929   \excludecomment{nassertion}
6930 }
```

**nsproof**

```
6931 \bool_if:NTF \c__notesslides_notes_bool {
6932   \newenvironment{nsproof}[2] [] {\begin{sproof}[#1]{#2}}\end{sproof}}
6933 }{
6934   \excludecomment{nsproof}
6935 }
```

**nexample**

```
6936 \bool_if:NTF \c__notesslides_notes_bool {
6937   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}\end{sexample}}
6938 }{
6939   \excludecomment{nexample}
6940 }
```

**\inputref@\*skip** We customize the hooks for in \inputref.

```
6941 \def\inputref@preskip{\smallskip}
6942 \def\inputref@postskip{\medskip}
```

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

`\inputref*`

```
6943 \let\orig@inputref\inputref
6944 \def\inputref{\@ifstar\ninputref\orig@inputref}
6945 \newcommand\ninputref[2][] {
6946   \bool_if:NT \c__notesslides_notes_bool {
6947     \orig@inputref[#1]{#2}
6948   }
6949 }
```

(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>}`.

```
6950 \newlength{\slidelogoheight}
6951
6952 \bool_if:NTF \c__notesslides_notes_bool {
6953   \setlength{\slidelogoheight}{.4cm}
6954 }{
6955   \setlength{\slidelogoheight}{1cm}
6956 }
6957 \newsavebox{\slidelogo}
6958 \sbox{\slidelogo}{\text{\TeX}}
6959 \newrobustcmd{\setslidelogo}[1]{\def\source{#1}}
6960 \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
6961 }
```

(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.

```
6962 \def\source{Michael Kohlhase}% customize locally
6963 \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.

```
6964 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
6965 \newsavebox{\cclogo}
6966 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
6967 \newif\ifcchref\cchreffalse
6968 \AtBeginDocument{
6969   \@ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
6970 }
6971 \def\licensing{
6972   \ifcchref
```

```

6973     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
6974 \else
6975     {\usebox{\cclogo}}
6976 \fi
6977 }
6978 \newrobustcmd{\setlicensing}[2][]{
6979   \def\@url{#1}
6980   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
6981   \ifx\@url\@empty
6982     \def\licensing{\usebox{\cclogo}}
6983   \else
6984     \def\licensing{
6985       \ifcchref
6986         \href{#1}{\usebox{\cclogo}}
6987       \else
6988         {\usebox{\cclogo}}
6989       \fi
6990     }
6991   \fi
6992 }

```

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

EdN:16

\slidelabel Now, we set up the slide label for the article mode.<sup>16</sup>

```

6993 \newrobustcmd\miko@slidelabel{
6994   \vbox to \slidelogoheight{
6995     \vss\hbox to \slidewidth
6996     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
6997   }
6998 }

```

(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.

```

7000 \def\Gin@mhrepos{}
7001 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7002 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
7003 \newrobustcmd\frameimage[2][]{
7004   \stepcounter{slide}
7005   \bool_if:NT \c__notesslides_frameimages_bool {
7006     \def\Gin@ewidth{}\setkeys{Gin}{#1}
7007     \bool_if:NF \c__notesslides_notes_bool { \vfill }
7008     \begin{center}
7009       \bool_if:NTF \c__notesslides_fiboxed_bool {
7010         \fbox{
7011           \ifx\Gin@ewidth\@empty
7012             \ifx\Gin@mhrepos\@empty
7013               \mhgraphics[width=\slidewidth,#1]{#2}
7014             \else

```

---

<sup>16</sup>EdNOTE: see that we can use the themes for the slides some day. This is all fake.



```

7014         \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7015     \fi
7016 \else% Gin@ewidth empty
7017     \ifx\Gin@mhrepos\@empty
7018         \mhgraphics[#1]{#2}
7019     \else
7020         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7021     \fi
7022 \fi% Gin@ewidth empty
7023 }
7024 }{
7025     \ifx\Gin@ewidth\@empty
7026     \ifx\Gin@mhrepos\@empty
7027         \mhgraphics[width=\slidewidth,#1]{#2}
7028     \else
7029         \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7030     \fi
7031     \ifx\Gin@mhrepos\@empty
7032         \mhgraphics[#1]{#2}
7033     \else
7034         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7035     \fi
7036 \fi% Gin@ewidth empty
7037 }
7038 \end{center}
7039 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
7040 \bool_if:NF \c__notesslides_notes_bool { \vfill }
7041 }
7042 } % 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.

```

7043 \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

```

7044 \AddToHook{begindocument}{
7045     \definecolor{green}{rgb}{0,.5,0}
7046     \definecolor{purple}{cmyk}{.3,1,0,.17}
7047 }

```

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`.

```

7048 % \def\STpresent#1{\textcolor{blue}{#1}}
7049 \def\defemph#1{\textcolor{magenta}{#1}}
7050 \def\symrefemph#1{\textcolor{cyan}{#1}}
7051 \def\compemph#1{\textcolor{blue}{#1}}
7052 \def\titleemph#1{\textcolor{blue}{#1}}
7053 \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.

```

7054 \pgfdeclareimage[width=.8em]{miko@small@dbend}{stex-dangerous-bend}
7055 \def\smalltextwarning{
7056   \pgfuseimage{miko@small@dbend}
7057   \xspace
7058 }
7059 \pgfdeclareimage[width=1.2em]{miko@dbend}{stex-dangerous-bend}
7060 \newrobustcmd\textwarning{
7061   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
7062   \xspace
7063 }
7064 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{stex-dangerous-bend}
7065 \newrobustcmd\bigtextwarning{
7066   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
7067   \xspace
7068 }

```

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

```

7069 \newrobustcmd\putgraphicsat[3]{
7070   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
7071 }
7072 \newrobustcmd\putat[2]{
7073   \begin{picture}(0,0)\put(#1){#2}\end{picture}
7074 }

```

## 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`.

```

7075 \bool_if:NT \c__notesslides_sectocframes_bool {
7076   \str_if_eq:VnTF \__notesslidesstopsect{part}{
7077     \newcounter{chapter}\counterwithin*{section}{chapter}
7078   }{
7079     \str_if_eq:VnT \__notesslidesstopsect{chapter}{
7080       \newcounter{chapter}\counterwithin*{section}{chapter}
7081     }
7082   }
7083 }

```

`\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
7084 \def\part@prefix{}
7085 \@ifpackageloaded{document-structure}{}{
7086   \str_case:VnF \__notesslidesstopsect {
7087     {part}{
7088       \int_set:Nn \l_document_structure_section_level_int {0}
7089       \def\thesection{\arabic{chapter}.\arabic{section}}

```

```

7090     \def\part@prefix{\arabic{chapter}.}
7091   }
7092   {chapter}{
7093     \int_set:Nn \l_document_structure_section_level_int {1}
7094     \def\thesection{\arabic{chapter}.\arabic{section}}
7095     \def\part@prefix{\arabic{chapter}.}
7096   }
7097 }{
7098   \int_set:Nn \l_document_structure_section_level_int {2}
7099   \def\part@prefix{}
7100 }
7101 }
7102
7103 \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 choses the L<sup>A</sup>T<sub>E</sub>X sectioning macros according to `\section@level`.

#### sfragment

```

7104 \renewenvironment{sfragment}[2][]{
7105   \_document_structure_omgroup_args:n { #1 }
7106   \int_incr:N \l_document_structure_section_level_int
7107   \bool_if:NT \c__notesslides_sectocframes_bool {
7108     \stepcounter{slide}
7109     \begin{frame}[noframenumbering]
7110     \vfill\Large\centering
7111     \red{
7112       \ifcase\l_document_structure_section_level_int\or
7113         \stepcounter{part}
7114         \def\_notesslideslabel{\omdoc@part@kw~\Roman{part}}
7115         \def\currentsectionlevel{\omdoc@part@kw}
7116       \or
7117         \stepcounter{chapter}
7118         \def\_notesslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
7119         \def\currentsectionlevel{\omdoc@chapter@kw}
7120       \or
7121         \stepcounter{section}
7122         \def\_notesslideslabel{\part@prefix\arabic{section}}
7123         \def\currentsectionlevel{\omdoc@section@kw}
7124       \or
7125         \stepcounter{subsection}
7126         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
7127         \def\currentsectionlevel{\omdoc@subsection@kw}
7128       \or
7129         \stepcounter{subsubsection}
7130         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
7131         \def\currentsectionlevel{\omdoc@subsubsection@kw}
7132       \or
7133         \stepcounter{paragraph}
7134         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
7135         \def\currentsectionlevel{\omdoc@paragraph@kw}
7136       \else
7137         \def\_notesslideslabel{}

```

```

7138         \def\currentsectionlevel{\omdoc@paragraph@kw}
7139         \fi% end ifcase
7140         \_notesslideslabel%\sref@label@id\_notesslideslabel
7141         \quad #2%
7142     }%
7143     \vfill%
7144     \end{frame}%
7145 }
7146 \str_if_empty:NF \l__document_structure_omgroup_id_str {
7147     \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
7148 }
7149 }{}
7150 }

```

We set up a beamer template for theorems like ams style, but without a block environment.

```

7151 \def\inserttheorembodyfont{\normalfont}
7152 %\bool_if:NF \c__notesslides_notes_bool {
7153 % \defbeamertemplate{theorem begin}{miko}
7154 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
7155 % \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
7156 % \inserttheorempunctuation\inserttheorembodyfont\hspace}
7157 % \defbeamertemplate{theorem end}{miko}{\}

```

and we set it as the default one.

```

7158 % \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.

```

7159 % \expandafter\def\csname Parent2\endcsname{}
7160 %}
7161
7162 \AddToHook{begindocument}{% this does not work for some reason
7163     \setbeamertemplate{theorems}[ams style]
7164 }
7165 \bool_if:NT \c__notesslides_notes_bool {
7166     \renewenvironment{columns}[1][\]{}%
7167     \par\noindent%
7168     \begin{minipage}%
7169     \slidewidth\centering\leavevmode%
7170 }{}%
7171     \end{minipage}\par\noindent%
7172 }%
7173 \newsavebox\columnbox%
7174 \renewenvironment<>{column}[2][\]{}%
7175     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
7176 }{}%
7177     \end{minipage}\end{lrbox}\usebox\columnbox%
7178 }%
7179 }
7180 \bool_if:NTF \c__notesslides_noproblems_bool {
7181     \newenvironment{problems}{}{}
7182 }{
7183     \excludecomment{problems}
7184 }

```

## 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.

```

7185 \gdef\printexcursions{}
7186 \newcommand\excursionref[2]{% label, text
7187   \bool_if:NT \c__notesslides_notes_bool {
7188     \begin{sparagraph}[title=Excursion]
7189       #2 \sref[fallback=the appendix]{#1}.
7190     \end{sparagraph}
7191   }
7192 }
7193 \newcommand\activate@excursion[2][]{
7194   \gappto\printexcursions{\inputref{#1}{#2}}
7195 }
7196 \newcommand\excursion[4][]{% repos, label, path, text
7197   \bool_if:NT \c__notesslides_notes_bool {
7198     \activate@excursion[#1]{#3}\excursionref{#2}{#4}
7199   }
7200 }

```

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

`\excursiongroup`

```

7201 \keys_define:nn{notesslides / excursiongroup }{
7202   id          .str_set_x:N = \l__notesslides_excursion_id_str,
7203   intro       .tl_set:N   = \l__notesslides_excursion_intro_tl,
7204   mhrepos     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
7205 }
7206 \cs_new_protected:Nn \__notesslides_excursion_args:n {
7207   \tl_clear:N \l__notesslides_excursion_intro_tl
7208   \str_clear:N \l__notesslides_excursion_id_str
7209   \str_clear:N \l__notesslides_excursion_mhrepos_str
7210   \keys_set:nn {notesslides / excursiongroup }{ #1 }
7211 }
7212 \newcommand\excursiongroup[1][]{
7213   \__notesslides_excursion_args:n{ #1 }
7214   \ifdefempty\printexcursions{}% only if there are excursions
7215   {\begin{note}
7216     \begin{sfragment}[#1]{Excursions}%
7217     \ifdefempty\l__notesslides_excursion_intro_tl{\{
7218       \inputref[\l__notesslides_excursion_mhrepos_str]{
7219         \l__notesslides_excursion_intro_tl
7220       }
7221     }
7222     \printexcursions%
7223     \end{sfragment}
7224     \end{note}}
7225 }
7226 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{\fi
7227 \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.

```
7228 <*package>
7229 <@@=problems>
7230 \ProvidesExplPackage{problem}{2022/02/26}{3.0.1}{Semantic Markup for Problems}
7231 \RequirePackage{13keys2e,stex}
7232
7233 \keys_define:nn { problem / pkg }{
7234   notes      .default:n    = { true },
7235   notes      .bool_set:N   = \c__problems_notes_bool,
7236   gnotes     .default:n    = { true },
7237   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
7238   hints      .default:n    = { true },
7239   hints      .bool_set:N   = \c__problems_hints_bool,
7240   solutions  .default:n    = { true },
7241   solutions  .bool_set:N   = \c__problems_solutions_bool,
7242   pts        .default:n    = { true },
7243   pts        .bool_set:N   = \c__problems_pts_bool,
7244   min        .default:n    = { true },
7245   min        .bool_set:N   = \c__problems_min_bool,
7246   boxed      .default:n    = { true },
7247   boxed      .bool_set:N   = \c__problems_boxed_bool,
7248   unknown    .code:n       = {}
7249 }
7250 \newif\ifsolutions
7251
7252 \ProcessKeysOptions{ problem / pkg }
7253 \bool_if:NTF \c__problems_solutions_bool {
7254   \solutionstrue
7255 }{
7256   \solutionsfalse
7257 }
```

Then we make sure that the necessary packages are loaded (in the right versions).

```
7258 \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.

```
7259 \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.

```
7260 \def\prob@problem@kw{Problem}
7261 \def\prob@solution@kw{Solution}
7262 \def\prob@hint@kw{Hint}
7263 \def\prob@note@kw{Note}
7264 \def\prob@gnote@kw{Grading}
7265 \def\prob@pt@kw{pt}
7266 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
7267 \AddToHook{begindocument}{
7268   \ltx@ifpackageloaded{babel}{
7269     \makeatletter
7270     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7271     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7272       \input{problem-ngerman.ldf}
7273     }
7274     \clist_if_in:NnT \l_tmpa_clist {finnish}{
7275       \input{problem-finnish.ldf}
7276     }
7277     \clist_if_in:NnT \l_tmpa_clist {french}{
7278       \input{problem-french.ldf}
7279     }
7280     \clist_if_in:NnT \l_tmpa_clist {russian}{
7281       \input{problem-russian.ldf}
7282     }
7283     \makeatother
7284   }{ }
7285 }
```

## 39.2 Problems and Solutions

We now prepare the KeyVal support for problems. The key macros just set appropriate internal macros.

```
7286 \keys_define:nn{ problem / problem }{
7287   id      .str_set_x:N = \l__problems_prob_id_str,
7288   pts     .tl_set:N    = \l__problems_prob_pts_tl,
7289   min     .tl_set:N    = \l__problems_prob_min_tl,
7290   title   .tl_set:N    = \l__problems_prob_title_tl,
7291   type    .tl_set:N    = \l__problems_prob_type_tl,
7292   refnum  .int_set:N    = \l__problems_prob_refnum_int
7293 }
7294 \cs_new_protected:Nn \__problems_prob_args:n {
```

```

7295 \str_clear:N \l__problems_prob_id_str
7296 \tl_clear:N \l__problems_prob_pts_tl
7297 \tl_clear:N \l__problems_prob_min_tl
7298 \tl_clear:N \l__problems_prob_title_tl
7299 \tl_clear:N \l__problems_prob_type_tl
7300 \int_zero_new:N \l__problems_prob_refnum_int
7301 \keys_set:nn { problem / problem }{ #1 }
7302 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
7303   \let\l__problems_prob_refnum_int\undefined
7304 }
7305 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

7306 \newcounter{problem}
7307 \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.

```

7308 \newcommand\prob@label[1]{#1}

```

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

`\prob@number` We consolidate the problem number into a reusable internal macro

```

7309 \newcommand\prob@number{
7310   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
7311     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
7312   }{
7313     \int_if_exist:NTF \l__problems_prob_refnum_int {
7314       \prob@label{\int_use:N \l__problems_prob_refnum_int }
7315     }{
7316       \prob@label\theproblem
7317     }
7318   }
7319 }

```

(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.

```

7320 \newcommand\prob@title[3]{%
7321   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
7322     #2 \l__problems_inclprob_title_tl #3
7323   }{
7324     \tl_if_exist:NTF \l__problems_prob_title_tl {
7325       #2 \l__problems_prob_title_tl #3
7326     }{
7327       #1
7328     }
7329   }
7330 }

```



(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.

```

7331 \def\prob@heading{
7332   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
7333   %\sref@label{id}\prob@problem@kw~\prob@number}{~}
7334 }

```

(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`

```

7335 \newenvironment{sproblem}[1][{}]{
7336   \__problems_prob_args:n{#1}%\sref@target%
7337   \@in@omtexttrue% we are in a statement (for inline definitions)
7338   \stepcounter{problem}\record@problem
7339   \def\current@section@level{\prob@problem@kw}
7340   \tl_if_exist:NTF \l__problems_inclprob_type_tl {
7341     \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
7342   }{
7343     \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
7344   }
7345   \str_if_exist:NTF \l__problems_inclprob_id_str {
7346     \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
7347   }{
7348     \str_set_eq:NN \sproblemid \l__problems_prob_id_str
7349   }
7350
7351
7352   \clist_set:No \l_tmpa_clist \sproblemtype
7353   \tl_clear:N \l_tmpa_tl
7354   \clist_map_inline:Nn \l_tmpa_clist {
7355     \tl_if_exist:cT {\__problems_sproblem_##1_start:}{
7356       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_start:}}
7357     }
7358   }
7359   \tl_if_empty:NTF \l_tmpa_tl {
7360     \__problems_sproblem_start:
7361   }{
7362     \l_tmpa_tl
7363   }
7364   \stex_ref_new_doc_target:n \sproblemid
7365 }{
7366   \clist_set:No \l_tmpa_clist \sproblemtype
7367   \tl_clear:N \l_tmpa_tl
7368   \clist_map_inline:Nn \l_tmpa_clist {
7369     \tl_if_exist:cT {\__problems_sproblem_##1_end:}{
7370       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_end:}}
7371     }

```

```

7372 }
7373 \tl_if_empty:NTF \l_tmpa_tl {
7374   \__problems_sproblem_end:
7375 }{
7376   \l_tmpa_tl
7377 }
7378
7379
7380 \smallskip
7381 }
7382
7383
7384 \cs_new_protected:Nn \__problems_sproblem_start: {
7385   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
7386 }
7387 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
7388
7389 \newcommand\stexpatchproblem[3][ ] {
7390   \str_set:Nx \l_tmpa_str{ #1 }
7391   \str_if_empty:NTF \l_tmpa_str {
7392     \tl_set:Nn \__problems_sproblem_start: { #2 }
7393     \tl_set:Nn \__problems_sproblem_end: { #3 }
7394   }{
7395     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
7396     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
7397   }
7398 }
7399
7400
7401 \bool_if:NT \c__problems_boxed_bool {
7402   \surroundwithmdframed{problem}
7403 }

```

**\record@problem** This macro records information about the problems in the \*.aux file.

```

7404 \def\record@problem{
7405   \protected@write\@auxout{}
7406   {
7407     \string\@problem{\prob@number}
7408     {
7409       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7410         \l__problems_inclprob_pts_tl
7411       }{
7412         \l__problems_prob_pts_tl
7413       }
7414     }%
7415     {
7416       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7417         \l__problems_inclprob_min_tl
7418       }{
7419         \l__problems_prob_min_tl
7420       }
7421     }
7422   }
7423 }

```

(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).

```
7424 \def\@problem#1#2#3{}
```

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

**solution** The **solution** environment is similar to the **problem** environment, only that it is independent of the boxed mode. It also has it's own keys that we need to define first.

```
7425 \keys_define:nn { problem / solution }{
7426   id                .str_set_x:N = \l__problems_solution_id_str ,
7427   for               .tl_set:N    = \l__problems_solution_for_tl ,
7428   height            .dim_set:N   = \l__problems_solution_height_dim ,
7429   creators          .clist_set:N = \l__problems_solution_creators_clist ,
7430   contributors      .clist_set:N = \l__problems_solution_contributors_clist ,
7431   srccite           .tl_set:N    = \l__problems_solution_srccite_tl
7432 }
7433 \cs_new_protected:Nn \__problems_solution_args:n {
7434   \str_clear:N \l__problems_solution_id_str
7435   \tl_clear:N \l__problems_solution_for_tl
7436   \tl_clear:N \l__problems_solution_srccite_tl
7437   \clist_clear:N \l__problems_solution_creators_clist
7438   \clist_clear:N \l__problems_solution_contributors_clist
7439   \dim_zero:N \l__problems_solution_height_dim
7440   \keys_set:nn { problem / solution }{ #1 }
7441 }
```

the next step is to define a helper macro that does what is needed to start a solution.

```
7442 \newcommand\@startsolution[1][{}]{
7443   \__problems_solution_args:n { #1 }
7444   \@in@omtexttrue% we are in a statement.
7445   \bool_if:NF \c__problems_boxed_bool { \hrule }
7446   \smallskip\noindent
7447   {\textbf\prob@solution@kw : \enspace}
7448   \begin{small}
7449   \def\current@section@level{\prob@solution@kw}
7450   \ignorespacesandpars
7451 }
```

**\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.

```
7452 \newcommand\startsolutions{
7453   \specialcomment{solution}{\@startsolution}{
7454     \bool_if:NF \c__problems_boxed_bool {
7455       \hrule\medskip
7456     }
7457     \end{small}%
7458   }
7459   \bool_if:NT \c__problems_boxed_bool {
7460     \surroundwithmdframed{solution}
7461   }
7462 }
```

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

\stopsolutions

```
7463 \newcommand\stopsolutions{\excludecomment{solution}}
```

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

so it only remains to start/stop solutions depending on what option was specified.

```
7464 \ifsolutions
7465 \startsolutions
7466 \else
7467 \stopsolutions
7468 \fi
```

exnote

```
7469 \bool_if:NTF \c__problems_notes_bool {
7470 \newenvironment{exnote}[1][]{
7471 \par\smallskip\hrule\smallskip
7472 \noindent\textbf{\prob@note@kw : }\small
7473 }{
7474 \smallskip\hrule
7475 }
7476 }{
7477 \excludecomment{exnote}
7478 }
```

hint

```
7479 \bool_if:NTF \c__problems_notes_bool {
7480 \newenvironment{hint}[1][]{
7481 \par\smallskip\hrule\smallskip
7482 \noindent\textbf{\prob@hint@kw :~ }\small
7483 }{
7484 \smallskip\hrule
7485 }
7486 \newenvironment{exhint}[1][]{
7487 \par\smallskip\hrule\smallskip
7488 \noindent\textbf{\prob@hint@kw :~ }\small
7489 }{
7490 \smallskip\hrule
7491 }
7492 }{
7493 \excludecomment{hint}
7494 \excludecomment{exhint}
7495 }
```

gnote

```
7496 \bool_if:NTF \c__problems_notes_bool {
7497 \newenvironment{gnote}[1][]{
7498 \par\smallskip\hrule\smallskip
7499 \noindent\textbf{\prob@gnote@kw : }\small
7500 }{
7501 \smallskip\hrule
7502 }
7503 }{
7504 \excludecomment{gnote}
7505 }
```

## 39.3 Multiple Choice Blocks

EdN:17

mcb 17

```

7506 \newenvironment{mcb}{
7507   \begin{enumerate}
7508 }{
7509   \end{enumerate}
7510 }
```

we define the keys for the mcc macro

```

7511 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
7512   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
7513     \bool_set_true:N #1
7514   }{
7515     \bool_set_false:N #1
7516   }
7517 }
7518 \keys_define:nn { problem / mcc }{
7519   id          .str_set_x:N = \l__problems_mcc_id_str ,
7520   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
7521   T           .default:n    = { true } ,
7522   T           .bool_set:N   = \l__problems_mcc_t_bool ,
7523   F           .default:n    = { true } ,
7524   F           .bool_set:N   = \l__problems_mcc_f_bool ,
7525   Ttext       .code:n       = {
7526     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
7527   } ,
7528   Ftext       .code:n       = {
7529     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
7530   }
7531 }
7532 \cs_new_protected:Nn \l__problems_mcc_args:n {
7533   \str_clear:N \l__problems_mcc_id_str
7534   \tl_clear:N \l__problems_mcc_feedback_tl
7535   \bool_set_true:N \l__problems_mcc_t_bool
7536   \bool_set_true:N \l__problems_mcc_f_bool
7537   \bool_set_true:N \l__problems_mcc_Ttext_bool
7538   \bool_set_false:N \l__problems_mcc_Ftext_bool
7539   \keys_set:nn { problem / mcc }{ #1 }
7540 }
```

\mcc

```

7541 \newcommand\mcc[2][]{
7542   \l__problems_mcc_args:n{ #1 }
7543   \item #2
7544   \ifsolutions
7545     \\\
7546     \bool_if:NT \l__problems_mcc_t_bool {
7547       % TODO!
7548       % \ifcsstring{mcc@T}{T}{\mcc@Ttext}%
7549     }
7550     \bool_if:NT \l__problems_mcc_f_bool {
```

---

<sup>17</sup>EdNOTE: MK: maybe import something better here from a dedicated MC package

```

7551      % TODO!
7552      % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
7553    }
7554    \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
7555      !
7556    }{
7557      \l__problems_mcc_feedback_tl
7558    }
7559    \fi
7560  } %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.

```

7561
7562 \keys_define:nn{ problem / inclproblem }{
7563   id      .str_set:N = \l__problems_inclprob_id_str,
7564   pts     .tl_set:N  = \l__problems_inclprob_pts_tl,
7565   min     .tl_set:N  = \l__problems_inclprob_min_tl,
7566   title   .tl_set:N  = \l__problems_inclprob_title_tl,
7567   refnum  .int_set:N  = \l__problems_inclprob_refnum_int,
7568   type    .tl_set:N  = \l__problems_inclprob_type_tl,
7569   mhrepos .str_set:N = \l__problems_inclprob_mhrepos_str
7570 }
7571 \cs_new_protected:Nn \l__problems_inclprob_args:n {
7572   \str_clear:N \l__problems_prob_id_str
7573   \tl_clear:N \l__problems_inclprob_pts_tl
7574   \tl_clear:N \l__problems_inclprob_min_tl
7575   \tl_clear:N \l__problems_inclprob_title_tl
7576   \tl_clear:N \l__problems_inclprob_type_tl
7577   \int_zero_new:N \l__problems_inclprob_refnum_int
7578   \str_clear:N \l__problems_inclprob_mhrepos_str
7579   \keys_set:nn { problem / inclproblem }{ #1 }
7580   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
7581     \let\l__problems_inclprob_pts_tl\undefined
7582   }
7583   \tl_if_empty:NT \l__problems_inclprob_min_tl {
7584     \let\l__problems_inclprob_min_tl\undefined
7585   }
7586   \tl_if_empty:NT \l__problems_inclprob_title_tl {
7587     \let\l__problems_inclprob_title_tl\undefined
7588   }
7589   \tl_if_empty:NT \l__problems_inclprob_type_tl {
7590     \let\l__problems_inclprob_type_tl\undefined
7591   }
7592   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
7593     \let\l__problems_inclprob_refnum_int\undefined
7594   }
7595 }

```

```

7596
7597 \cs_new_protected:Nn \__problems_inclprob_clear: {
7598   \let\l__problems_inclprob_id_str\undefined
7599   \let\l__problems_inclprob_pts_tl\undefined
7600   \let\l__problems_inclprob_min_tl\undefined
7601   \let\l__problems_inclprob_title_tl\undefined
7602   \let\l__problems_inclprob_type_tl\undefined
7603   \let\l__problems_inclprob_refnum_int\undefined
7604   \let\l__problems_inclprob_mhrepos_str\undefined
7605 }
7606 \__problems_inclprob_clear:
7607
7608 \newcommand\includeproblem[2][ ]{
7609   \__problems_inclprob_args:n{ #1 }
7610   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
7611     \input{#2}
7612   }{
7613     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
7614       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
7615     }
7616   }
7617   \__problems_inclprob_clear:
7618 }

```

(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.

```

7619 \AddToHook{enddocument}{
7620   \bool_if:NT \c__problems_pts_bool {
7621     \message{Total:~\arabic{pts}~points}
7622   }
7623   \bool_if:NT \c__problems_min_bool {
7624     \message{Total:~\arabic{min}~minutes}
7625   }
7626 }

```

The margin pars are reader-visible, so we need to translate

```

7627 \def\pts#1{
7628   \bool_if:NT \c__problems_pts_bool {
7629     \marginpar{#1~\prob@pt@kw}
7630   }
7631 }
7632 \def\min#1{
7633   \bool_if:NT \c__problems_min_bool {
7634     \marginpar{#1~\prob@min@kw}
7635   }
7636 }

```

`\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.

```

7637 \newcounter{pts}
7638 \def\show@pts{
7639   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7640     \bool_if:NT \c__problems_pts_bool {
7641       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
7642       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
7643     }
7644   }{
7645     \tl_if_exist:NT \l__problems_prob_pts_tl {
7646       \bool_if:NT \c__problems_pts_bool {
7647         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
7648         \addtocounter{pts}{\l__problems_prob_pts_tl}
7649       }
7650     }
7651   }
7652 }

```

(End definition for `\show@pts`. This function is documented on page ??.)  
and now the same for the minutes

`\show@min`

```

7653 \newcounter{min}
7654 \def\show@min{
7655   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7656     \bool_if:NT \c__problems_min_bool {
7657       \marginpar{\l__problems_inclprob_min_tl\ min}
7658       \addtocounter{min}{\l__problems_inclprob_min_tl}
7659     }
7660   }{
7661     \tl_if_exist:NT \l__problems_prob_min_tl {
7662       \bool_if:NT \c__problems_min_bool {
7663         \marginpar{\l__problems_prob_min_tl\ min}
7664         \addtocounter{min}{\l__problems_prob_min_tl}
7665       }
7666     }
7667   }
7668 }
7669 </package>

```

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



## Chapter 40

# Implementation: The hwexam Class

The functionality is spread over the `hwexam` class and package. The class provides the `document` environment and pre-loads some convenience packages, whereas the package provides the concrete functionality.

### 40.1 Class Options

To initialize the `hwexam` class, we declare and process the necessary options by passing them to the respective packages and classes they come from.

```
7670 <@@=hwexam>
7671 <*cls>
7672 \ProvidesExplClass{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
7673 \RequirePackage{l3keys2e}
7674 \DeclareOption*{
7675   \PassOptionsToClass{\CurrentOption}{document-structure}
7676   \PassOptionsToPackage{\CurrentOption}{stex}
7677   \PassOptionsToPackage{\CurrentOption}{hwexam}
7678   \PassOptionsToPackage{\CurrentOption}{tikzinput}
7679 }
7680 \ProcessOptions
```

We load `omdoc.cls`, and the desired packages. For the L<sup>A</sup>T<sub>E</sub>XML bindings, we make sure the right packages are loaded.

```
7681 \LoadClass{document-structure}
7682 \RequirePackage{stex}
7683 \RequirePackage{hwexam}
7684 \RequirePackage{tikzinput}
7685 \RequirePackage{graphicx}
7686 \RequirePackage{a4wide}
7687 \RequirePackage{amssymb}
7688 \RequirePackage{amstext}
7689 \RequirePackage{amsmath}
```

Finally, we register another keyword for the `document` environment. We give a default assignment type to prevent errors

```

7690 \newcommand\assig@default@type{\hwexam@assignment@kw}
7691 \def\document@hwexamtype{\assig@default@type}
7692 <@@=document_structure>
7693 \keys_define:nn { document-structure / document }{
7694 id .str_set_x:N = \c_document_structure_document_id_str,
7695 hwexamtype .tl_set:N = \document@hwexamtype
7696 }
7697 <@@=hwexam>
7698 </cls>

```

## Chapter 41

# Implementation: The hwexam Package

### 41.1 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.

```
7699 \*package>
7700 \ProvidesExplPackage{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
7701 \RequirePackage{13keys2e}
7702
7703 \newif\iftest\testfalse
7704 \DeclareOption{test}{\testtrue}
7705 \newif\ifmultiple\multiplefalse
7706 \DeclareOption{multiple}{\multipletrue}
7707 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
7708 \ProcessOptions
```

Then we make sure that the necessary packages are loaded (in the right versions).

```
7709 \RequirePackage{keyval}[1997/11/10]
7710 \RequirePackage{problem}
```

`\hwexam@*kw` For multilinguality, we define internal macros for keywords that can be specialized in `*.ldf` files.

```
7711 \newcommand\hwexam@assignment@kw{Assignment}
7712 \newcommand\hwexam@given@kw{Given}
7713 \newcommand\hwexam@due@kw{Due}
7714 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
7715 blank~for~extra~space}
7716 \def\hwexam@minutes@kw{minutes}
7717 \newcommand\correction@probs@kw{prob.}
7718 \newcommand\correction@pts@kw{total}
7719 \newcommand\correction@reached@kw{reached}
7720 \newcommand\correction@sum@kw{Sum}
7721 \newcommand\correction@grade@kw{grade}
7722 \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

```

7723 \AddToHook{begindocument}{
7724 \ltx@ifpackageloaded{babel}{
7725 \makeatletter
7726 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7727 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7728 \input{hwexam-ngerman.ldf}
7729 }
7730 \clist_if_in:NnT \l_tmpa_clist {finnish}{
7731 \input{hwexam-finnish.ldf}
7732 }
7733 \clist_if_in:NnT \l_tmpa_clist {french}{
7734 \input{hwexam-french.ldf}
7735 }
7736 \clist_if_in:NnT \l_tmpa_clist {russian}{
7737 \input{hwexam-russian.ldf}
7738 }
7739 \makeatother
7740 }{}
7741 }
7742

```

## 41.2 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.

```

7743 \newcounter{assignment}
7744 \numberproblemsin{assignment}
7745 \renewcommand\prob@label[1]{\assignment@number.#1}

```

We will prepare the keyval support for the `assignment` environment.

```

7746 \keys_define:nn { hwexam / assignment } {
7747 id .str_set:N = \l__hwexam_assign_id_str,
7748 number .int_set:N = \l__hwexam_assign_number_int,
7749 title .tl_set:N = \l__hwexam_assign_title_tl,
7750 type .tl_set:N = \l__hwexam_assign_type_tl,
7751 given .tl_set:N = \l__hwexam_assign_given_tl,
7752 due .tl_set:N = \l__hwexam_assign_due_tl,
7753 loadmodules .code:n = {
7754 \bool_set_true:N \l__hwexam_assign_loadmodules_bool
7755 }
7756 }
7757 \cs_new_protected:Nn \__hwexam_assignment_args:n {
7758 \str_clear:N \l__hwexam_assign_id_str
7759 \int_set:Nn \l__hwexam_assign_number_int {-1}
7760 \tl_clear:N \l__hwexam_assign_title_tl
7761 \tl_clear:N \l__hwexam_assign_type_tl
7762 \tl_clear:N \l__hwexam_assign_given_tl
7763 \tl_clear:N \l__hwexam_assign_due_tl
7764 \bool_set_false:N \l__hwexam_assign_loadmodules_bool

```

```

7765 \keys_set:nn { hwexam / assignment }{ #1 }
7766 }

```

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.

```

7767 \newcommand\given@due[2]{
7768 \bool_lazy_all:nF {
7769 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
7770 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
7771 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
7772 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
7773 }{ #1 }
7774
7775 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
7776 \tl_if_empty:NF \l__hwexam_assign_given_tl {
7777 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
7778 }
7779 }{
7780 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
7781 }
7782
7783 \bool_lazy_or:nnF {
7784 \bool_lazy_and_p:nn {
7785 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
7786 }{
7787 \tl_if_empty_p:V \l__hwexam_assign_due_tl
7788 }
7789 }{
7790 \bool_lazy_and_p:nn {
7791 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
7792 }{
7793 \tl_if_empty_p:V \l__hwexam_assign_due_tl
7794 }
7795 }{ ,~ }
7796
7797 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
7798 \tl_if_empty:NF \l__hwexam_assign_due_tl {
7799 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
7800 }
7801 }{
7802 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
7803 }
7804
7805 \bool_lazy_all:nF {
7806 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
7807 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
7808 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
7809 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
7810 }{ #2 }
7811 }

```

`\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.

```

7812 \newcommand\assignment@title[3]{
7813 \tl_if_empty:NTF \l__hwexam_inclasssign_title_tl {
7814 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
7815 #1
7816 }{
7817 #2\l__hwexam_assign_title_tl#3
7818 }
7819 }{
7820 #2\l__hwexam_inclasssign_title_tl#3
7821 }
7822 }

```

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

`\assignment@number` Like `\assignment@title` only for the number, and no around part.

```

7823 \newcommand\assignment@number{
7824 \int_compare:nNnTF \l__hwexam_inclasssign_number_int = {-1} {
7825 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
7826 \arabic{assignment}
7827 } {
7828 \int_use:N \l__hwexam_assign_number_int
7829 }
7830 }{
7831 \int_use:N \l__hwexam_inclasssign_number_int
7832 }
7833 }

```

(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.

```

7834 \newenvironment{assignment}[1][ ]{
7835 \__hwexam_assignment_args:n { #1 }
7836 %\sref@target
7837 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
7838 \global\stepcounter{assignment}
7839 }{
7840 \global\setcounter{assignment}{\int_use:N\l__hwexam_assign_number_int}
7841 }
7842 \setcounter{problem}{0}
7843 \def\current@section@level{\document@hwexamtype}
7844 %\sref@label@id{\document@hwexamtype \thesection}
7845 \begin{@assignment}
7846 }{
7847 \end{@assignment}
7848 }

```

In the multi-assignment case we just use the omdoc environment for suitable sectioning.

```

7849 \def\ass@title{
7850 \protect\document@hwexamtype~\arabic{assignment}
7851 \assignment@title{}\{;\}{} -- \given@due{}\}{}
7852 }
7853 \ifmultiple
7854 \newenvironment{@assignment}{
7855 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
7856 \begin{sfragment}[loadmodules]{\ass@title}
7857 }{
7858 \begin{sfragment}{\ass@title}
7859 }
7860 }{
7861 \end{sfragment}
7862 }

```

for the single-page case we make a title block from the same components.

```

7863 \else
7864 \newenvironment{@assignment}{
7865 \begin{center}\bf
7866 \Large@title\strut\
7867 \document@hwexamtype~\arabic{assignment}\assignment@title{}\{;\}{}{\}{}
7868 \large\given@due{--;\}{}\}{}
7869 \end{center}
7870 }{}
7871 \fi% multiple

```

### 41.3 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.

```

7872 \keys_define:nn { hwexam / inclassignment } {
7873 %id .str_set_x:N = \l__hwexam_assign_id_str,
7874 number .int_set:N = \l__hwexam_inclassign_number_int,
7875 title .tl_set:N = \l__hwexam_inclassign_title_tl,
7876 type .tl_set:N = \l__hwexam_inclassign_type_tl,
7877 given .tl_set:N = \l__hwexam_inclassign_given_tl,
7878 due .tl_set:N = \l__hwexam_inclassign_due_tl,
7879 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
7880 }
7881 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
7882 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
7883 \tl_clear:N \l__hwexam_inclassign_title_tl
7884 \tl_clear:N \l__hwexam_inclassign_type_tl
7885 \tl_clear:N \l__hwexam_inclassign_given_tl
7886 \tl_clear:N \l__hwexam_inclassign_due_tl
7887 \str_clear:N \l__hwexam_inclassign_mhrepos_str
7888 \keys_set:nn { hwexam / inclassignment }{ #1 }
7889 }
7890 \__hwexam_inclassignment_args:n {}
7891
7892 \newcommand\inputassignment[2][{}]{

```

```

7893 \_hwexam_inclassnment_args:n { #1 }
7894 \str_if_empty:NTF \l_hwexam_inclassn_mhrepos_str {
7895   \input{#2}
7896 }{
7897   \stex_in_repository:nn{\l_hwexam_inclassn_mhrepos_str}{
7898     \input{\mhp{path}\l_hwexam_inclassn_mhrepos_str}{#2}}
7899 }
7900 }
7901 \_hwexam_inclassnment_args:n {}
7902 }
7903 \newcommand\includeassignment[2][]{
7904   \newpage
7905   \inputassignment[#1]{#2}
7906 }

```

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

## 41.4 Typesetting Exams

\quizheading

```

7907 \ExplSyntaxOff
7908 \newcommand\quizheading[1]{%
7909   \def\@tas{#1}%
7910   \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
7911   \ifx\@tas\@empty\else%
7912     \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
7913   \fi%
7914 }
7915 \ExplSyntaxOn

```

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

\testheading

```

7916
7917 \def\hwexamheader{\input{hwexam-default.header}}
7918
7919 \def\hwexamminutes{
7920   \tl_if_empty:NTF \testheading@duration {
7921     {\testheading@min}~\hwexam@minutes@kw
7922   }{
7923     \testheading@duration
7924   }
7925 }
7926
7927 \keys_define:nn { hwexam / testheading } {
7928   min .tl_set:N = \testheading@min,
7929   duration .tl_set:N = \testheading@duration,
7930   reqpts .tl_set:N = \testheading@reqpts,
7931   tools .tl_set:N = \testheading@tools
7932 }
7933 \cs_new_protected:Nn \_hwexam_testheading_args:n {
7934   \tl_clear:N \testheading@min
7935   \tl_clear:N \testheading@duration

```



```

7936 \tl_clear:N \testheading@reqpts
7937 \tl_clear:N \testheading@tools
7938 \keys_set:nn { hwexam / testheading }{ #1 }
7939 }
7940 \newenvironment{testheading}[1][ ]{
7941 \_hwexam_testheading_args:n{ #1 }
7942 \newcount\check@time\check@time=\testheading@min
7943 \advance\check@time by -\theassignment@totalmin
7944 \newif\if@bonuspoints
7945 \tl_if_empty:NTF \testheading@reqpts {
7946 \@bonuspointsfalse
7947 }{
7948 \newcount\bonus@pts
7949 \bonus@pts=\theassignment@totalpts
7950 \advance\bonus@pts by -\testheading@reqpts
7951 \edef\bonus@pts{\the\bonus@pts}
7952 \@bonuspointstrue
7953 }
7954 \edef\check@time{\the\check@time}
7955
7956 \makeatletter\hwexamheader\makeatother
7957 }{
7958 \newpage
7959 }

```

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

\testspace

```

7960 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

7961 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

7962 \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.

```

7963 <@=problems>
7964 \renewcommand\@problem[3]{
7965 \stepcounter{assignment@probs}
7966 \def\__problemspts{#2}
7967 \ifx\__problemspts\@empty\else
7968 \addtocounter{assignment@totalpts}{#2}
7969 \fi
7970 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
7971 \xdef\correction@probs{\correction@probs & #1}%
7972 \xdef\correction@pts{\correction@pts & #2}
7973 \xdef\correction@reached{\correction@reached &}

```

```

7974 }
7975 <@@=hwexam>

```

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

`\correction@table` This macro generates the correction table

```

7976 \newcounter{assignment@probs}
7977 \newcounter{assignment@totalpts}
7978 \newcounter{assignment@totalmin}
7979 \def\correction@probs{\correction@probs@kw}
7980 \def\correction@pts{\correction@pts@kw}
7981 \def\correction@reached{\correction@reached@kw}
7982 \stepcounter{assignment@probs}
7983 \newcommand\correction@table{
7984 \resizebox{\textwidth}{!}{%
7985 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
7986 &\multicolumn{\theassignment@probs}{c|}{}%|
7987 {\footnotesize\correction@forgrading@kw} &\\ \hline
7988 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
7989 \correction@pts & \theassignment@totalpts & \\ \hline
7990 \correction@reached & & \[.7cm]\hline
7991 \end{tabular}}
7992 </package>

```

(End definition for \correction@table. This function is documented on page ??.)

## 41.5 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}

```