

# The sTeX3 Package \*

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

2022-03-08

## Abstract

sTeX is a collection of L<sup>A</sup>T<sub>E</sub>X packages that allow to markup documents semantically without leaving the document format, 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 2022-03-08)

# 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>102</b>
25.1	Generic Path Handling . . . . .	102
25.2	PWD and kpsewhich . . . . .	104
25.3	File Hooks and Tracking . . . . .	105
25.4	MathHub Repositories . . . . .	106
25.5	Using Content in Archives . . . . .	110
<b>26</b>	<b>gTeX-References Implementation</b>	<b>115</b>
26.1	Document URIs and URLs . . . . .	115
26.2	Setting Reference Targets . . . . .	117
26.3	Using References . . . . .	119
<b>27</b>	<b>gTeX-Modules Implementation</b>	<b>122</b>
27.1	The smodule environment . . . . .	126
27.2	Invoking modules . . . . .	131
<b>28</b>	<b>gTeX-Module Inheritance Implementation</b>	<b>133</b>
28.1	SMS Mode . . . . .	133
28.2	Inheritance . . . . .	136

<b>29</b>	<b>STEX-Symbols Implementation</b>	<b>141</b>
29.1	Symbol Declarations . . . . .	141
29.2	Notations . . . . .	148
29.3	Variables . . . . .	157
<b>30</b>	<b>STEX-Terms Implementation</b>	<b>164</b>
30.1	Symbol Invocations . . . . .	164
30.2	Terms . . . . .	171
30.3	Notation Components . . . . .	175
30.4	Variables . . . . .	177
30.5	Sequences . . . . .	179
<b>31</b>	<b>STEX-Structural Features Implementation</b>	<b>180</b>
31.1	Imports with modification . . . . .	181
31.2	The feature environment . . . . .	187
31.3	Structure . . . . .	188
<b>32</b>	<b>STEX-Statements Implementation</b>	<b>197</b>
32.1	Definitions . . . . .	197
32.2	Assertions . . . . .	202
32.3	Examples . . . . .	205
32.4	Logical Paragraphs . . . . .	208
<b>33</b>	<b>The Implementation</b>	<b>213</b>
33.1	Package Options . . . . .	213
33.2	Proofs . . . . .	213
33.3	Justifications . . . . .	224
<b>34</b>	<b>STEX-Others Implementation</b>	<b>226</b>
<b>35</b>	<b>STEX-Metatheory Implementation</b>	<b>227</b>
<b>36</b>	<b>Tikzinput Implementation</b>	<b>230</b>
<b>37</b>	<b>document-structure.sty Implementation</b>	<b>232</b>
37.1	The document-structure Class . . . . .	232
37.2	Class Options . . . . .	232
37.3	Beefing up the document environment . . . . .	233
37.4	Implementation: document-structure Package . . . . .	233
37.5	Package Options . . . . .	233
37.6	Document Structure . . . . .	235
37.7	Front and Backmatter . . . . .	238
37.8	Global Variables . . . . .	240

<b>38 NotesSlides – Implementation</b>	<b>241</b>
38.1 Class and Package Options . . . . .	241
38.2 Notes and Slides . . . . .	243
38.3 Header and Footer Lines . . . . .	247
38.4 Frame Images . . . . .	248
38.5 Colors and Highlighting . . . . .	249
38.6 Sectioning . . . . .	250
38.7 Excursions . . . . .	253
<b>39 The Implementation</b>	<b>254</b>
39.1 Package Options . . . . .	254
39.2 Problems and Solutions . . . . .	255
39.3 Multiple Choice Blocks . . . . .	261
39.4 Including Problems . . . . .	262
39.5 Reporting Metadata . . . . .	263
<b>40 Implementation: The hwexam Class</b>	<b>265</b>
40.1 Class Options . . . . .	265
<b>41 Implementation: The hwexam Package</b>	<b>267</b>
41.1 Package Options . . . . .	267
41.2 Assignments . . . . .	268
41.3 Including Assignments . . . . .	271
41.4 Typesetting Exams . . . . .	272
41.5 Leftovers . . . . .	274

# 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  $\text{\texttt{S}\TeX}$  concept relates to the 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>
```

...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 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  $\hookrightarrow$  `<module-URI>?<constant-name>`.

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

### Example 3

Input:

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

Output:

Given a `foo`, we can...

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

### Example 4

Input:

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

Output:

`this` is a symbol taking two arguments.

`\notation`

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

#### Example 5

Input:

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

Output:

First:  $a$ ; Second:  $b$

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

`\comp`

Unfortunately, we have no highlighting whatsoever now. That is because we need to tell  $\text{\TeX}$  explicitly which parts of the notation are *notation components* which *should* be highlighted. We can do so with the `\comp` command.

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

#### Example 6

Input:

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

Output:

First:  $a$ ; Second:  $b$



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

Note that it is required that

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

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

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



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

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

## `\symdef`

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

### Example 7

Input:

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

Output:

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

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



---

`\setnotation`

---

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

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

### Operator Notations

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

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

#### Example 8

Input:

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

Output:

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

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

### 3.3.3 Argument Types

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

## b-Type Arguments

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

$\hookrightarrow$  **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=bihi]
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  $\text{lcm}(a, b)$  in *english*, but is called *kleinstes gemeinsames Vielfaches* in *german* and consequently denoted as  $\text{kgV}(a, b)$  there.

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

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

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

### 3.4.2 Simple Inheritance and Namespaces

---

`\importmodule`  
`\usemodule`

---

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

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

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



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

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

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

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

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



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

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

The module `Foo` must either be declared in the



file  $\langle top-directory \rangle / some/path/Foo[. \langle lang \rangle].tex$ , or in  $\langle top-directory \rangle / some/path[. \langle lang \rangle].tex$  (which are checked in that order).

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

---

`\STEXexport`

---

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



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

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

### 3.4.3 The `mathstructure` Environment

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

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

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

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

How this works is again best demonstrated by example:

#### Example 24

Input:

```

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

Output:

A `monoid` is...

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

#### Example 25

Input:

```

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

Output:

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

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

#### Example 26

Input:

```

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

```

Output:

Test:  $a \cdot c \cdot c$

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  $m$ .

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

---

**\arg**

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

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

#### Example 34

Input:

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

Output:

Addition is...

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

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

#### Example 35

Input:

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

Output:

adding  $k$  yields...

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

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

### Example 36

Input:

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

Output:

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

## 4.3 Referencing Symbols and Statements

TODO: references documentation

## Chapter 5

# sTeX Statements

### 5.1 Definitions, Theorems, Examples, Paragraphs

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

The corresponding environments for that are:

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

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

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

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

#### Example 37

Input:

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

Output:

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

`\definiendum`  
`\definame`  
`\definiens`

`sdefinition` (and `sparagraph` with `type=symdoc`) introduce three new macros: `definiendum` behaves like `symref` (and `definame` like `symname`), but highlights the references 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.
- $\rightarrow$  The MMT-system can use those (in lieu of an actual `sdefinition` in scope) to present to users, e.g. when hovering over symbols.
- $\rightsquigarrow$

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 `⟨content⟩` with

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

### 8.1.2 Babel Languages

---

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

---

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

### 8.1.3 Auxiliary Methods

---

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

---

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

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

---

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

---

## Chapter 9

# STEX-MathHub

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

### 9.1 Macros and Environments

---

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

---

#### 9.1.1 Files, Paths, URIs

---

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

---

---

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

---

---

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

---

---

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

---

---

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

---

---

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

---



---

<code>\stex_filestack_push:n</code>	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>mhrepos</code> . It then resolves the file path in <code>\mhgraphics[mhrepos=Foo/Bar]{foo/bar.png}</code> relative to the <code>source-</code> folder of the <code>Foo/Bar</code> -archive. <code>\cmhgraphics</code> additional wraps the image in a <code>center</code> -environment.
<hr/> <hr/> <code>\lstinputmhlisting</code> <code>\clstinputmhlisting</code>	Like <code>\mhgraphics</code> , but only defined if the <code>listings</code> -package is loaded, and with <code>\lstinputlisting</code> instead of <code>\includegraphics</code> .

# Chapter 10

## STEX-References

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

### 10.1 Macros and Environments

---

---

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

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

---

---

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

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

---

---

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

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

---

---

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

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

---

---

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

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

#### 10.1.1 Setting Reference Targets

---

---

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

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

---

---

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

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

### 10.1.2 Using References

---

**\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** `\stex_invoke_module:n`

---

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

---

**`\stex_activate_module:n`**

---

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

## Chapter 12

# STEX-Module Inheritance

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

### 12.1 Macros and Environments

#### 12.1.1 SMS Mode

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

---

`\g_stex_smsmode_allowedmacros_tl`

---

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

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

---

`\g_stex_smsmode_allowedmacros_escape_tl`

---

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

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

---

`\g_stex_smsmode_allowedenvs_seq`

---

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

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

---

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

---

Tests whether SMS mode is currently active.



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

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

### 12.1.2 Imports and Inheritance

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

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

---

**\stex\_import\_module\_uri:nn**

---

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

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

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

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

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

---

<code>\symdecl</code>	<code>\symdecl{<i>macroname</i>}[<i>args</i>]</code>
-----------------------	--

---

Declares a new symbol with semantic macro `\macroname`. Optional arguments are:

- **name**: An (OMDOC) name. By default equal to `<macroname>`.
- **type**: An (ideally semantic) term, representing a *type*. Not used by ST<sub>E</sub>X, but passed on to MMT for semantic services.
- **def**: An (ideally semantic) term, representing a *definiens*. Not used by ST<sub>E</sub>X, 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. `\symdecl{plus}[args=ii]` allows for `\plus{2}{2}`.
  - a** an *associative* argument; i.e. a sequence of arbitrarily many arguments provided as a comma-separated list, e.g. `\symdecl{plus}[args=a]` allows for `\plus{2,2,2}`.
  - b** a *variable* argument. Is treated by ST<sub>E</sub>X 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. `\symdecl{forall}[args=bi]` allows for `\forall{x \in \mathbb{N}}{x \geq 0}`.

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

# Chapter 14

## STEX-Terms

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

### 14.1 Macros and Environments

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

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

EdN:3

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

## 17.3 Limitations

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

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

## Chapter 18

# sTeX-Metatheory

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

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

### 18.1 Symbols



**Part III**  
**Extensions**

## Chapter 19

# Tikzinput

### 19.1 Macros and Environments

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

## Chapter 20

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

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

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

### 20.1 Introduction

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

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

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

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

## 20.2 The User Interface

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

### 20.2.1 Package and Class Options

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

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

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

### 20.2.2 Document Structure

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

```

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

```

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

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

`blindfragment`

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

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

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

Example 3: A typical Document Structure of a Book

`\skipomgroup`

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

`\currentsectionlevel`  
`\CurrentSectionLevel`

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

---

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

### 20.2.3 Ignoring Inputs

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

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

**narrative/content markup** In  $\text{\TeX}$  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,  $\text{\TeX}$  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 L<sup>A</sup>T<sub>E</sub>XML 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 L<sup>A</sup>T<sub>E</sub>X. 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  $\text{\TeX}$  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> | <ul style="list-style-type: none"><li>• The options <code>slides</code> and <code>notes</code> switch between slides mode and notes mode (see Section 21.2.2).</li></ul> |
| <code>notes</code>  |  |



<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 `noframenumbers` option so that the frame numbering is kept in sync between the slides and the course notes.

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

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

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

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

### 21.2.3 Header and Footer Lines of the Slides

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

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

`\setslidelogo`  
`\setsource`  
`\setlicensing`  
  
`\frameimage`  
  
`\mhframeimage`

### 21.2.4 Frame Images

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

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.

```
\title{320101 General Computer Science (Fall 2010)}
\begin{testheading}[duration=one hour,min=60,reqpts=27]
  Good luck to all students!
\end{testheading}
```

---

Name:

## 320101 General Computer Science (Fall 2010)

2022-03-08

You have one hour (sharp) for the test;

Write the solutions to the sheet.

The estimated time for solving this exam is 58 minutes, leaving you 2 minutes for revising your exam.

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

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

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

[illegible]

good luck

Example 8: A generated test heading.

Part IV

# Implementation

## Chapter 24

# ST<sub>E</sub>X -Basics Implementation

### 24.1 The ST<sub>E</sub>XDocument Class

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

```
1 <*cls>
2
3 %%%%%%%%% basics.dtx %%%%%%%%%
4
5 \RequirePackage{expl3,l3keys2e}
6 \ProvidesExplClass{stex}{2022/03/03}{3.1.0}{sTeX document class}
7 \LoadClass[border=1px,varwidth]{standalone}
8 \setlength\textwidth{15cm}
9
10 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
11 \ProcessOptions
12
13 \RequirePackage{stex}
14 </cls>
```

### 24.2 Preliminaries

```
15 <*package>
16
17 %%%%%%%%% basics.dtx %%%%%%%%%
18
19 \RequirePackage{expl3,l3keys2e,ltxcmds}
20 \ProvidesExplPackage{stex}{2022/03/03}{3.1.0}{sTeX package}
21
22 %\RequirePackage{morewrites}
23 %\RequirePackage{amsmath}
24
25 Package options:
26 \keys_define:nn { stex } {
```

```

26 debug      .clist_set:N = \c_stex_debug_clist ,
27 lang       .clist_set:N = \c_stex_languages_clist ,
28 mathhub    .tl_set_x:N   = \mathhub ,
29 sms        .bool_set:N   = \c_stex_persist_mode_bool ,
30 image      .bool_set:N   = \c_tikzinput_image_bool ,
31 unknown    .code:n       = {}
32 }
33 \ProcessKeysOptions { stex }

```

**\stex** The  $\TeX$  logo:

**\sTeX**

```

34 \protected\def\stex{
35   \texorpdfstring{\raisebox{-.5ex}{S}\kern-.5ex\TeX}{sTeX}\xspace%
36 }
37 \let\sTeX\stex

```

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

## 24.3 Messages and logging

```

38 <@@=stex_log>
    Warnings and error messages
39 \msg_new:nnn{stex}{error/unknownlanguage}{
40   Unknown~language:~#1
41 }
42 \msg_new:nnn{stex}{warning/nomathhub}{
43   MATHHUB~system~variable~not~found~and~no~
44   \detokenize{\mathhub}~value~set!
45 }
46 \msg_new:nnn{stex}{error/deactivated-macro}{
47   The~\detokenize{#1}~command~is~only~allowed~in~#2!
48 }

```

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

```

49 \cs_new_protected:Nn \stex_debug:nn {
50   \clist_if_in:NnTF \c_stex_debug_clist { all } {
51     \msg_set:nnn{stex}{debug / #1}{
52       \\Debug~#1:~#2\\
53     }
54     \msg_none:nn{stex}{debug / #1}
55   }{
56     \clist_if_in:NnT \c_stex_debug_clist { #1 } {
57       \msg_set:nnn{stex}{debug / #1}{
58         \\Debug~#1:~#2\\
59       }
60       \msg_none:nn{stex}{debug / #1}
61     }
62   }
63 }

```

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

Redirecting messages:

```

64 \clist_if_in:NnTF \c_stex_debug_clist {all} {
65   \msg_redirect_module:nnn{ stex }{ none }{ term }

```

```

66 }{
67   \clist_map_inline:Nn \c_stex_debug_clist {
68     \msg_redirect_name:nnn{ stex }{ debug / ##1 }{ term }
69   }
70 }
71
72 \stex_debug:nn{log}{debug~mode~on}

```

## 24.4 HTML Annotations

```

73 <@=stex_annotate>
74 \RequirePackage{rustex}

```

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

```

75 \rustex_add_Namespace:nn{stex}{http://kwarc.info/ns/sTeX}

```

Conditionals for L<sup>A</sup>T<sub>E</sub>XML:

`\if@latexml`

```

76 \ifcsname if@latexml\endcsname\else
77   \expandafter\newif\csname if@latexml\endcsname\@latexmlfalse
78 \fi

```

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

`\latexml_if_p:`

`\latexml_if:TF`

```

79 \prg_new_conditional:Nnn \latexml_if: {p, T, F, TF} {
80   \if@latexml
81     \prg_return_true:
82   \else:
83     \prg_return_false:
84   \fi:
85 }

```

(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.

```

86 \tl_new:N \l__stex_annotate_arg_tl
87 \tl_const:Nx \c__stex_annotate_emptyarg_tl {
88   \rustex_if:TF {
89     \rustex_direct_HTML:n { \c_ampersand_str lrm; }
90   }{-}
91 }

```

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

`\__stex_annotate_checkempty:n`

```

92 \cs_new_protected:Nn \__stex_annotate_checkempty:n {
93   \tl_set:Nn \l__stex_annotate_arg_tl { #1 }
94   \tl_if_empty:NT \l__stex_annotate_arg_tl {
95     \tl_set_eq:NN \l__stex_annotate_arg_tl \c__stex_annotate_emptyarg_tl
96   }
97 }

```

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



```

\stex_if_do_html_p: Whether to (locally) produce HTML output
\stex_if_do_html:TF
  98 \bool_new:N \_stex_html_do_output_bool
  99 \bool_set_true:N \_stex_html_do_output_bool
  100
  101 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
  102   \bool_if:nTF \_stex_html_do_output_bool
  103     \prg_return_true: \prg_return_false:
  104 }

```

(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
  105 \cs_new_protected:Nn \stex_suppress_html:n {
  106   \exp_args:Nne \use:nn {
  107     \bool_set_false:N \_stex_html_do_output_bool
  108     #1
  109   }{
  110     \stex_if_do_html:T {
  111       \bool_set_true:N \_stex_html_do_output_bool
  112     }
  113   }
  114 }

```

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

`\stex_annotate:nnv` 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`

```

  115 \rustex_if:TF{
  116   \cs_new_protected:Nn \stex_annotate:nnn {
  117     \__stex_annotate_checkempty:n { #3 }
  118     \rustex_annotate_HTML:nn {
  119       property="stex:#1" ~
  120       resource="#2"
  121     } {
  122       \mode_if_vertical:TF{
  123         \tl_use:N \l__stex_annotate_arg_tl\par
  124       }{
  125         \tl_use:N \l__stex_annotate_arg_tl
  126       }
  127     }
  128   }
  129   \cs_new_protected:Nn \stex_annotate_invisible:n {
  130     \__stex_annotate_checkempty:n { #1 }
  131     \rustex_annotate_HTML:nn {
  132       stex:visible="false" ~
  133       style:display="none"
  134     } {
  135       \mode_if_vertical:TF{
  136         \tl_use:N \l__stex_annotate_arg_tl\par
  137       }{
  138         \tl_use:N \l__stex_annotate_arg_tl
  139       }
  140     }
  141   }

```

```

140     }
141   }
142   \cs_new_protected:Nn \stex_annotate_invisible:nnn {
143     \__stex_annotate_checkempty:n { #3 }
144     \rustex_annotate_HTML:nn {
145       property="stex:#1" ~
146       resource="#2" ~
147       stex:visible="false" ~
148       style:display="none"
149     } {
150       \mode_if_vertical:TF{
151         \tl_use:N \l__stex_annotate_arg_tl\par
152       }{
153         \tl_use:N \l__stex_annotate_arg_tl
154       }
155     }
156   }
157   \NewDocumentEnvironment{stex_annotate_env} { m m } {
158     \par
159     \rustex_annotate_HTML_begin:n {
160       property="stex:#1" ~
161       resource="#2"
162     }
163   }{
164     \par\rustex_annotate_HTML_end:
165   }
166 }{
167   \latexml_if:TF {
168     \cs_new_protected:Nn \stex_annotate:nnn {
169       \__stex_annotate_checkempty:n { #3 }
170       \mode_if_math:TF {
171         \cs:w latexml@annotate@math\cs_end:{#1}{#2}{
172           \tl_use:N \l__stex_annotate_arg_tl
173         }
174       }{
175         \cs:w latexml@annotate@text\cs_end:{#1}{#2}{
176           \tl_use:N \l__stex_annotate_arg_tl
177         }
178       }
179     }
180     \cs_new_protected:Nn \stex_annotate_invisible:n {
181       \__stex_annotate_checkempty:n { #1 }
182       \mode_if_math:TF {
183         \cs:w latexml@invisible@math\cs_end:{
184           \tl_use:N \l__stex_annotate_arg_tl
185         }
186       } {
187         \cs:w latexml@invisible@text\cs_end:{
188           \tl_use:N \l__stex_annotate_arg_tl
189         }
190       }
191     }
192     \cs_new_protected:Nn \stex_annotate_invisible:nnn {
193       \__stex_annotate_checkempty:n { #3 }

```

```

194     \cs:w latexml@annotate@invisible\cs_end:{#1}{#2}{
195       \tl_use:N \l__stex_annotate_arg_tl
196     }
197   }
198   \NewDocumentEnvironment{stex_annotate_env} { m m } {
199     \par\begin{latexml@annotateenv}{#1}{#2}
200   }{
201     \par\end{latexml@annotateenv}
202   }
203 }{
204   \cs_new_protected:Nn \stex_annotate:nnn {#3}
205   \cs_new_protected:Nn \stex_annotate_invisible:n {}
206   \cs_new_protected:Nn \stex_annotate_invisible:nnn {}
207   \NewDocumentEnvironment{stex_annotate_env} { m m } {}{}
208 }
209 }

```

(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

```

210 <@@=stex_language>

```

`\c_stex_languages_prop`  
`\c_stex_language_abbrevs_prop`

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

```

211 \prop_const_from_keyval:Nn \c_stex_languages_prop {
212   en = english ,
213   de = ngerman ,
214   ar = arabic ,
215   bg = bulgarian ,
216   ru = russian ,
217   fi = finnish ,
218   ro = romanian ,
219   tr = turkish ,
220   fr = french
221 }
222
223 \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop {
224   english = en ,
225   ngerman = de ,
226   arabic = ar ,
227   bulgarian = bg ,
228   russian = ru ,
229   finnish = fi ,
230   romanian = ro ,
231   turkish = tr ,
232   french = fr
233 }
234 % todo: chinese simplified (zhs)
235 %       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:

```

236 \clist_if_empty:NF \c_stex_languages_clist {
237   \clist_clear:N \l_tmpa_clist
238   \clist_map_inline:Nn \c_stex_languages_clist {
239     \prop_get:NnNTF \c_stex_languages_prop { #1 } \l_tmpa_str {
240       \clist_put_right:No \l_tmpa_clist \l_tmpa_str
241     } {
242       \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
243     }
244   }
245   \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
246   \RequirePackage[\clist_use:Nn \l_tmpa_clist,]{babel}
247 }

```

## 24.6 Auxiliary Methods

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

```

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

```

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

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

```

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

```

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

**\ignorespacesandpars**

```

257 \protected\def\ignorespacesandpars{
258   \begingroup\catcode13=10\relax
259   \@ifnextchar\par{
260     \endgroup\expandafter\ignorespacesandpars\@gobble
261   }{
262     \endgroup
263   }
264 }
265 \</package>

```

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

## Chapter 25

# STEX -MathHub Implementation

```
266 <*package>
267
268 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
269
270 <@@=stex_path>
271
272 Warnings and error messages
273 \msg_new:nnn{stex}{error/norepository}{
274   No~archive~#1~found~in~#2
275 }
276 \msg_new:nnn{stex}{error/notinarchive}{
277   Not~currently~in~an~archive,~but~\detokenize{#1}~
278   needs~one!
279 }
280 \msg_new:nnn{stex}{error/nofile}{
281   \detokenize{#1}~could~not~find~file~#2
282 }
283 \msg_new:nnn{stex}{error/twofiles}{
284   \detokenize{#1}~found~two~candidates~for~#2
285 }
```

### 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`

```
284 \cs_new_protected:Nn \stex_path_from_string:Nn {
285   \str_set:Nx \l_tmpa_str { #2 }
286   \str_if_empty:NTF \l_tmpa_str {
287     \seq_clear:N #1
288   }{
289     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
290     \sys_if_platform_windows:T{
291       \seq_clear:N \l_tmpa_tl
```

```

292 \seq_map_inline:Nn #1 {
293   \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
294   \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
295 }
296 \seq_set_eq:NN #1 \l_tmpa_tl
297 }
298 \stex_path_canonicalize:N #1
299 }
300 }
301

```

(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`

```

302 \cs_new_protected:Nn \stex_path_to_string:NN {
303   \exp_args:Nne \str_set:Nn #2 { \seq_use:Nn #1 / }
304 }
305
306 \cs_new:Nn \stex_path_to_string:N {
307   \seq_use:Nn #1 /
308 }

```

(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`

```

309 \str_const:Nn \c__stex_path_dot_str {.}
310 \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.

```

311 \cs_new_protected:Nn \stex_path_canonicalize:N {
312   \seq_if_empty:NF #1 {
313     \seq_clear:N \l_tmpa_seq
314     \seq_get_left:NN #1 \l_tmpa_tl
315     \str_if_empty:NT \l_tmpa_tl {
316       \seq_put_right:Nn \l_tmpa_seq {}
317     }
318     \seq_map_inline:Nn #1 {
319       \str_set:Nn \l_tmpa_tl { ##1 }
320       \str_if_eq:NNF \l_tmpa_tl \c__stex_path_dot_str {
321         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
322           \seq_if_empty:NNTF \l_tmpa_seq {
323             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
324               \c__stex_path_up_str
325             }
326           }{
327             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
328             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
329               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
330                 \c__stex_path_up_str
331               }
332             }{

```

```

333         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
334     }
335 }
336 }{
337     \str_if_empty:NF \l_tmpa_tl {
338         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
339     }
340 }
341 }
342 }
343 \seq_gset_eq:NN #1 \l_tmpa_seq
344 }
345 }

```

(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 $\underline{TF}$`

```

346 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
347     \seq_if_empty:NTF #1 {
348         \prg_return_false:
349     }{
350         \seq_get_left:NN #1 \l_tmpa_tl
351         \sys_if_platform_windows:TF{
352             \str_if_in:NnTF \l_tmpa_tl {:}{
353                 \prg_return_true:
354             }{
355                 \prg_return_false:
356             }
357         }{
358             \str_if_empty:NTF \l_tmpa_tl {
359                 \prg_return_true:
360             }{
361                 \prg_return_false:
362             }
363         }
364     }
365 }

```

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

## 25.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

366 \str_new:N\l_stex_kpsewhich_return_str
367 \cs_new_protected:Nn \stex_kpsewhich:n {
368     \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
369     \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
370     \tl_trim_spaces:N \l_stex_kpsewhich_return_str
371 }

```

(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`

```

372 \sys_if_platform_windows:TF{
373   \begingroup\escapechar=-1\catcode'\=12
374   \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
375   \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
376   \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_
377   }}{
378   \stex_kpsewhich:n{-var-value~PWD}
379   }
380
381 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
382 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
383 \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

```

384 <@@=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

```

385 \seq_gclear_new:N\g__stex_files_stack

```

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`  
`\c_stex_mainfile_str`

```

386 \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
387 \stex_path_from_string:Nn \c_stex_mainfile_seq
388   \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`

```

389 \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`

```

390 \cs_new_protected:Nn \stex_filestack_push:n {
391   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
392   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
393     \stex_path_from_string:Nn\g_stex_currentfile_seq{
394       \c_stex_pwd_str/#1
395     }
396   }
397   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
398   \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
399 }

```



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

`\stex_filestack_pop:`

```

400 \cs_new_protected:Nn \stex_filestack_pop: {
401   \seq_if_empty:NF\g__stex_files_stack{
402     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
403   }
404   \seq_if_empty:NTF\g__stex_files_stack{
405     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
406   }{
407     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
408     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
409   }
410 }
```

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

Hooks for the current file:

```

411 \AddToHook{file/before}{
412   \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
413 }
414 \AddToHook{file/after}{
415   \stex_filestack_pop:
416 }
```

## 25.4 MathHub Repositories

417 `<@=stex_mathhub>`

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

```

418 \str_if_empty:NTF\mathhub{
419   \sys_if_platform_windows:TF{
420     \begingroup\escapechar=-1\catcode'\=12
421     \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str MATHHUB\c_percent_str}
422     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
423     \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_kpsewhich_return_str}}
424   }{
425     \stex_kpsewhich:n{-var-value-MATHHUB}
426   }
427   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
428 }
429 \str_if_empty:NTF\c_stex_mathhub_str{
430   \msg_warning:nn{stex}{warning/nomathhub}
431 }{
432   \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
433   \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
434 }
435 }{
436   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
437   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
438     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
439       \c_stex_pwd_str/\mathhub
440     }
441   }
```

```

441 }
442 \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
443 \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
444 }

```

(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

```

445 \cs_new_protected:Nn \_stex_mathhub_do_manifest:n {
446   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
447     \str_set:Nx \l_tmpa_str { #1 }
448     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
449     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
450     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
451     \_stex_mathhub_find_manifest:N \l_tmpa_seq
452     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
453       \msg_error:nnxx{stex}{error/norepository}{#1}{
454         \stex_path_to_string:N \c_stex_mathhub_str
455       }
456     } {
457       \exp_args:No \_stex_mathhub_parse_manifest:n { \l_tmpa_str }
458     }
459   }
460 }

```

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

`\l__stex_mathhub_manifest_file_seq`

```

461 \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`:

```

462 \cs_new_protected:Nn \_stex_mathhub_find_manifest:N {
463   \seq_set_eq:NN\l_tmpa_seq #1
464   \bool_set_true:N\l_tmpa_bool
465   \bool_while_do:Nn \l_tmpa_bool {
466     \seq_if_empty:NTF \l_tmpa_seq {
467       \bool_set_false:N\l_tmpa_bool
468     }{
469       \file_if_exist:nTF{
470         \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
471       }{
472         \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
473         \bool_set_false:N\l_tmpa_bool
474       }{
475         \file_if_exist:nTF{
476           \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
477         }{
478           \seq_put_right:Nn\l_tmpa_seq{META-INF}
479           \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}

```

```

480         \bool_set_false:N\l_tmpa_bool
481     }{
482         \file_if_exist:nTF{
483             \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
484         }{
485             \seq_put_right:Nn\l_tmpa_seq{meta-inf}
486             \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
487             \bool_set_false:N\l_tmpa_bool
488         }{
489             \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
490         }
491     }
492 }
493 }
494 }
495 \seq_set_eq:NN\l__stex_mathhub_manifest_file_seq\l_tmpa_seq
496 }

```

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

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

```

497 \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:

```

498 \cs_new_protected:Nn \__stex_mathhub_parse_manifest:n {
499     \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
500     \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
501     \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
502         \str_set:Nn \l_tmpa_str {##1}
503         \exp_args:NNoo \seq_set_split:Nnn
504             \l_tmpb_seq \c_colon_str \l_tmpa_str
505         \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
506             \exp_args:NNe \str_set:Nn \l_tmpb_tl {
507                 \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
508             }
509             \exp_args:No \str_case:nnTF \l_tmpa_tl {
510                 {id} {
511                     \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
512                     { id } \l_tmpb_tl
513                 }
514                 {narration-base} {
515                     \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
516                     { narr } \l_tmpb_tl
517                 }
518                 {url-base} {
519                     \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
520                     { docurl } \l_tmpb_tl
521                 }
522                 {source-base} {
523                     \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
524                     { ns } \l_tmpb_tl
525                 }

```

```

526     {ns} {
527         \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
528         { ns } \l_tmpb_tl
529     }
530     {dependencies} {
531         \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
532         { deps } \l_tmpb_tl
533     }
534     }{}{}
535     }{}
536 }
537 \ior_close:N \c__stex_mathhub_manifest_ior
538 }

```

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

`\stex_set_current_repository:n`

```

539 \cs_new_protected:Nn \stex_set_current_repository:n {
540     \stex_require_repository:n { #1 }
541     \prop_set_eq:Nc \l_stex_current_repository_prop {
542         c_stex_mathhub_#1_manifest_prop
543     }
544 }

```

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

`\stex_require_repository:n`

```

545 \cs_new_protected:Nn \stex_require_repository:n {
546     \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
547         \stex_debug:nn{mathhub}{Opening~archive:~#1}
548         \_stex_mathhub_do_manifest:n { #1 }
549     }
550 }

```

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

`\l_stex_current_repository_prop`

Current MathHub repository

```

551 %\prop_new:N \l_stex_current_repository_prop
552
553 \_stex_mathhub_find_manifest:N \c_stex_pwd_seq
554 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
555     \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
556 } {
557     \_stex_mathhub_parse_manifest:n { main }
558     \prop_get:Nn \c_stex_mathhub_main_manifest_prop {id}
559     \l_tmpa_str
560     \prop_set_eq:cN { c_stex_mathhub\_l_tmpa_str_manifest_prop }
561     \c_stex_mathhub_main_manifest_prop
562     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
563     \stex_debug:nn{mathhub}{Current~repository:~
564         \prop_item:Nn \l_stex_current_repository_prop {id}
565     }
566 }

```

(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.

```

567 \cs_new_protected:Nn \stex_in_repository:nn {
568   \str_set:Nx \l_tmpa_str { #1 }
569   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
570   \str_if_empty:NTF \l_tmpa_str {
571     \prop_if_exist:NTF \l_stex_current_repository_prop {
572       \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
573       \exp_args:Ne \l_tmpa_cs{
574         \prop_item:Nn \l_stex_current_repository_prop { id }
575     }
576   }{
577     \l_tmpa_cs{}
578   }
579 }{
580   \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}
581   \stex_require_repository:n \l_tmpa_str
582   \str_set:Nx \l_tmpa_str { #1 }
583   \exp_args:Nne \use:nn {
584     \stex_set_current_repository:n \l_tmpa_str
585     \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
586   }{
587     \stex_debug:nn{mathhub}{switching~back~to:~
588     \prop_if_exist:NTF \l_stex_current_repository_prop {
589       \prop_item:Nn \l_stex_current_repository_prop { id }::~
590     \meaning\l_stex_current_repository_prop
591   }{
592     no~repository
593   }
594 }
595 \prop_if_exist:NTF \l_stex_current_repository_prop {
596   \stex_set_current_repository:n {
597     \prop_item:Nn \l_stex_current_repository_prop { id }
598   }
599 }{
600   \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
601 }
602 }
603 }
604 }

```

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

## 25.5 Using Content in Archives

`\mhpath`

```

605 \def \mhpath #1 #2 {
606   \exp_args:Ne \tl_if_empty:nTF{#1}{
607     \c_stex_mathhub_str /
608     \prop_item:Nn \l_stex_current_repository_prop { id }
609     / source / #2
610 }{
611   \c_stex_mathhub_str / #1 / source / #2

```

```

612 }
613 }

```

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

`\inputref`  
`\mhinput`

```

614 \newif \ifinputref \inputreffalse
615
616 \cs_new_protected:Nn \__stex_mathhub_mhinput:nn {
617   \stex_in_repository:nn {#1} {
618     \ifinputref
619       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
620     \else
621       \inputreftrue
622       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
623     \inputreffalse
624   \fi
625 }
626 }
627 \NewDocumentCommand \mhinput { 0{} m}{
628   \stex_mhinput:nn{ #1 }{ #2 }
629 }
630
631 \cs_new_protected:Nn \__stex_mathhub_inputref:nn {
632   \stex_in_repository:nn {#1} {
633     \bool_lazy_any:nTF {
634       {\rustex_if_p:}
635       {\latexml_if_p:}
636     } {
637       \str_clear:N \l_tmpa_str
638       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
639         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
640       }
641       \stex_annotate_invisible:nnn{inputref}{
642         \l_tmpa_str / #2
643       }{}
644     }{
645       \begingroup
646         \inputreftrue
647         \input{ \c_stex_mathhub_str / ##1 / source / #2 }
648       \endgroup
649     }
650   }
651 }
652 \NewDocumentCommand \inputref { 0{} m}{
653   \__stex_mathhub_inputref:nn{ #1 }{ #2 }
654 }

```

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

`\addmhbibresource`

```

655 \cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {
656   \stex_in_repository:nn {#1} {
657     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
658   }

```

```

659 }
660 \newcommand\addmhbibresource[2][{}{
661   \_stex_mathhub_mhbibresource:nn{ #1 }{ #2 }
662 }

```

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

### \libinput

```

663 \cs_new_protected:Npn \libinput #1 {
664   \prop_if_exist:NF \l_stex_current_repository_prop {
665     \msg_error:nnn{stex}{error/notinarchive}\libinput
666   }
667   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
668     \msg_error:nnn{stex}{error/notinarchive}\libinput
669   }
670   \seq_clear:N \l__stex_mathhub_libinput_files_seq
671   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
672   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
673
674   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
675     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}
676     \IfFileExists{ \l_tmpa_str }{
677       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
678     }{}
679     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
680     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
681   }
682
683   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
684   \IfFileExists{ \l_tmpa_str }{
685     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
686   }{}
687
688   \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
689     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
690   }{
691     \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
692       \input{ ##1 }
693     }
694   }
695 }

```

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

### \libusepackage

```

696 \NewDocumentCommand \libusepackage {0{} m} {
697   \prop_if_exist:NF \l_stex_current_repository_prop {
698     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
699   }
700   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
701     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
702   }
703   \seq_clear:N \l__stex_mathhub_libinput_files_seq
704   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
705   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str

```

```

706
707 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
708   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2}
709   \IfFileExists{ \l_tmpa_str.sty }{
710     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
711   }{}
712   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
713   \seq_put_right:No \l_tmpa_seq \l_tmpa_str
714 }
715
716 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2}
717 \IfFileExists{ \l_tmpa_str.sty }{
718   \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
719 }{}
720
721 \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
722   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
723 }{
724   \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
725     \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
726       \usepackage[#1]{ #1 }
727     }
728   }{
729     \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
730   }
731 }
732 }

```

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

`\mhgraphics`  
`\cmhgraphics`

```

733
734 \AddToHook{begindocument}{
735   \ltx@ifpackageloaded{graphicx}{
736     \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
737     \newcommand\mhgraphics[2][]{\%
738       \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
739       \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}}
740     \newcommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
741   }{}

```

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

`\lstinputmhlisting`  
`\clstinputmhlisting`

```

742 \ltx@ifpackageloaded{listings}{
743   \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
744   \newcommand\lstinputmhlisting[2][]{\%
745     \def\lst@mhrepos{}\setkeys{lst}{#1}%
746     \lstinputlisting[#1]{\mhp\lst@mhrepos{#2}}}
747   \newcommand\clstinputmhlisting[2][]{\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
748 }{}
749 }
750
751 </package>

```



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

## Chapter 26

# STEX -References Implementation

```
752 <*package>
753
754 %%%%%%%%%%% references.dtx %%%%%%%%%%%
755
756 <@@=stex_refs>
    Warnings and error messages
757
```

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

```
758 %\iow_new:N \c__stex_refs_refs_iow
759 \AddToHook{begindocument}{
760 % \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
761 }
762 \AddToHook{enddocument}{
763 % \iow_close:N \c__stex_refs_refs_iow
764 }
```

`\STEXreftitle`

```
765 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
766
767 \NewDocumentCommand \STEXreftitle { m } {
768 \tl_gset:Nx \g__stex_refs_title_tl { #1 }
769 }
```

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

### 26.1 Document URIs and URLs

`\l_stex_current_docns_str`

```
770 \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:`

```
771 \cs_new_protected:Nn \stex_get_document_uri: {  
772   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq  
773   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str  
774   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str  
775   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str  
776   \seq_put_right:No \l_tmpa_seq \l_tmpb_str  
777  
778   \str_clear:N \l_tmpa_str  
779   \prop_if_exist:NT \l_stex_current_repository_prop {  
780     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {  
781       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}  
782     }  
783   }  
784  
785   \str_if_empty:NTF \l_tmpa_str {  
786     \str_set:Nx \l_stex_current_docns_str {  
787       file:/\stex_path_to_string:N \l_tmpa_seq  
788     }  
789   }{  
790     \bool_set_true:N \l_tmpa_bool  
791     \bool_while_do:Nn \l_tmpa_bool {  
792       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str  
793       \exp_args:No \str_case:nnTF { \l_tmpb_str } {  
794         {source} { \bool_set_false:N \l_tmpa_bool }  
795       }{}{  
796         \seq_if_empty:NT \l_tmpa_seq {  
797           \bool_set_false:N \l_tmpa_bool  
798         }  
799       }  
800     }  
801  
802     \seq_if_empty:NTF \l_tmpa_seq {  
803       \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str  
804     }{  
805       \str_set:Nx \l_stex_current_docns_str {  
806         \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq  
807       }  
808     }  
809   }  
810 }
```

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

`\l_stex_current_docurl_str`

```
811 \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:`

```
812 \cs_new_protected:Nn \stex_get_document_url: {  
813   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq  
814   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str  
815   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
```

```

816 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
817 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
818
819 \str_clear:N \l_tmpa_str
820 \prop_if_exist:NT \l_stex_current_repository_prop {
821   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
822     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
823       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
824     }
825   }
826 }
827
828 \str_if_empty:NTF \l_tmpa_str {
829   \str_set:Nx \l_stex_current_docurl_str {
830     file:/\stex_path_to_string:N \l_tmpa_seq
831   }
832 }{
833   \bool_set_true:N \l_tmpa_bool
834   \bool_while_do:Nn \l_tmpa_bool {
835     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
836     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
837       {source} { \bool_set_false:N \l_tmpa_bool }
838     }{}{
839       \seq_if_empty:NT \l_tmpa_seq {
840         \bool_set_false:N \l_tmpa_bool
841       }
842     }
843   }
844
845   \seq_if_empty:NTF \l_tmpa_seq {
846     \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
847   }{
848     \str_set:Nx \l_stex_current_docurl_str {
849       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
850     }
851   }
852 }
853 }

```

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

## 26.2 Setting Reference Targets

```

854 \str_const:Nn \c__stex_refs_url_str{URL}
855 \str_const:Nn \c__stex_refs_ref_str{REF}
856 \str_new:N \l__stex_refs_curr_label_str
857 % @currentlabel -> number
858 % @currentlabelname -> title
859 % @currentHref -> name.number <- id of some kind
860 % \theH# -> \arabic{section}
861 % \the# -> number
862 % \hyper@makecurrent{#}
863 \int_new:N \l__stex_refs_unnamed_counter_int

```

`\stex_ref_new_doc_target:n`

```

864 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
865   \stex_get_document_uri:
866   \str_clear:N \l__stex_refs_curr_label_str
867   \str_set:Nx \l_tmpa_str { #1 }
868   \str_if_empty:NT \l_tmpa_str {
869     \int_incr:N \l__stex_refs_unnamed_counter_int
870     \str_set:Nx \l_tmpa_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
871   }
872   \str_set:Nx \l__stex_refs_curr_label_str {
873     \l_stex_current_docns_str?\l_tmpa_str
874   }
875   \seq_if_exist:cF{g__stex_refs_labels_\l_tmpa_str_seq}{
876     \seq_new:c {g__stex_refs_labels_\l_tmpa_str_seq}
877   }
878   \seq_if_in:coF{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str {
879     \seq_gput_right:co{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str
880   }
881   \stex_if_smsmode:TF {
882     \stex_get_document_url:
883     \str_gset_eq:cN {sref_url_\l__stex_refs_curr_label_str_str}\l_stex_current_docurl_str
884     \str_gset_eq:cN {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_url_str
885   }{
886     %\iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter\unexpanded\expandafter{
887     \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
888     \immediate\write\@auxout{\stexauxadddocref{\l_stex_current_docns_str}{\l_tmpa_str}}
889     \str_gset:cx {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_ref_str
890   }
891 }

```

(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.

```

892 \cs_new_protected:Npn \stexauxadddocref #1 #2 {
893   \str_set:Nn \l_tmpa_str {#1?#2}
894   \str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
895   \seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
896     \seq_new:c {g__stex_refs_labels_#2_seq}
897   }
898   \seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
899     \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
900   }
901 }

```

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

```

902 \AtEndDocument{
903   \def\stexauxadddocref#1 #2 {}{}
904 }

```

`\stex_ref_new_sym_target:n`

```

905 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
906   \stex_if_smsmode:TF {
907     \str_if_exist:cF{sref_sym_#1_type}{
908       \stex_get_document_url:
909       \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str

```

```

910     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
911   }
912 }{
913   \str_if_empty:NF \l__stex_refs_curr_label_str {
914     \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
915     \immediate\write\@auxout{
916       \exp_not:N\expandafter\def\exp_not:N\csname \exp_not:N\detokenize{sref_sym_#1_label_
917         \l__stex_refs_curr_label_str
918       }
919     }
920   }
921 }
922 }

```

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

## 26.3 Using References

```

923 \str_new:N \l__stex_refs_indocument_str

```

**\sref** Optional arguments:

```

924
925 \keys_define:nn { stex / sref } {
926   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
927   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
928   pre           .tl_set:N = \l__stex_refs_pre_tl ,
929   post          .tl_set:N = \l__stex_refs_post_tl ,
930 }
931 \cs_new_protected:Nn \__stex_refs_args:n {
932   \tl_clear:N \l__stex_refs_linktext_tl
933   \tl_clear:N \l__stex_refs_fallback_tl
934   \tl_clear:N \l__stex_refs_pre_tl
935   \tl_clear:N \l__stex_refs_post_tl
936   \str_clear:N \l__stex_refs_repo_str
937   \keys_set:nn { stex / sref } { #1 }
938 }

```

The actual macro:

```

939 \NewDocumentCommand \sref { 0{} m}{
940   \__stex_refs_args:n { #1 }
941   \str_if_empty:NTF \l__stex_refs_indocument_str {
942     \str_set:Nx \l_tmpa_str { #2 }
943     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
944     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
945       \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
946         \seq_get_left:cNF {g__stex_refs_labels_\l_tmpa_str _seq} \l_tmpa_str {
947           \str_clear:N \l_tmpa_str
948         }
949       }{
950         \str_clear:N \l_tmpa_str
951       }
952     }{
953       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
954       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str

```

```

955 \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
956 \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str_seq}{
957   \str_set_eq:NN \l_tmpc_str \l_tmpa_str
958   \str_clear:N \l_tmpa_str
959   \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str_seq} {
960     \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
961       \str_range:nnn { ##1 }{ -\l_tmpa_int}{ -1 }
962     }{
963       \seq_map_break:n {
964         \str_set:Nn \l_tmpa_str { ##1 }
965       }
966     }
967   }
968 }{
969   \str_clear:N \l_tmpa_str
970 }
971 }
972 \str_if_empty:NTF \l_tmpa_str {
973   \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_lin
974 }{
975   \str_if_eq:cNTF {sref_\l_tmpa_str_type} \c__stex_refs_ref_str {
976     \tl_if_empty:NTF \l__stex_refs_linktext_tl {
977       \cs_if_exist:cTF{autoref}{
978         \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
979       }{
980         \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
981       }
982     }{
983       \ltx@ifpackageloaded{hyperref}{
984         \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
985       }{
986         \l__stex_refs_linktext_tl
987       }
988     }
989   }{
990     \ltx@ifpackageloaded{hyperref}{
991       \href{\use:c{sref_url_\l_tmpa_str_str}}{\tl_if_empty:NTF \l__stex_refs_linktext_t
992     }{
993       \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs
994     }
995   }
996 }
997 }{
998   % TODO
999 }
1000 }

```

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

## `\srefsym`

```

1001 \NewDocumentCommand \srefsym { 0{} m}{
1002   \stex_get_symbol:n { #2 }
1003   \__stex_refs_sym_aux:nn{##1}{\l_stex_get_symbol_uri_str}
1004 }

```

```

1005
1006 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1007   \str_if_exist:cTF {sref_sym_#2 _label_str }{
1008     \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1009   }{
1010     \__stex_refs_args:n { #1 }
1011     \str_if_empty:NTF \l__stex_refs_indocument_str {
1012       \tl_if_exist:cTF{sref_sym_#2 _type}{
1013         % doc uri in \l_tmpb_str
1014         \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1015         \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1016           % reference
1017           \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1018             \cs_if_exist:cTF{autoref}{
1019               \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1020             }{
1021               \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1022             }
1023           }{
1024             \ltx@ifpackageloaded{hyperref}{
1025               \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1026             }{
1027               \l__stex_refs_linktext_tl
1028             }
1029           }
1030         }{
1031           % URL
1032           \ltx@ifpackageloaded{hyperref}{
1033             \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl \
1034           }{
1035             \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_re
1036           }
1037         }
1038       }{
1039         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_l
1040       }
1041     }{
1042       % TODO
1043     }
1044   }
1045 }

```

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

**\srefsymuri**

```

1046 \cs_new_protected:Npn \srefsymuri #1 #2 {
1047   \__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1048 }

```

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

```

1049 </package>

```



## Chapter 27

# STEX -Modules Implementation

```
1050 <*package>
1051
1052 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1053
1054 <@@=stex_modules>
1055
1056   Warnings and error messages
1057   \msg_new:nnn{stex}{error/unknownmodule}{
1058     No~module~#1~found
1059   }
1060   \msg_new:nnn{stex}{error/syntax}{
1061     Syntax~error:~#1
1062   }
1063   \msg_new:nnn{stex}{error/siglanguage}{
1064     Module~#1~declares~signature~#2,~but~does~not~
1065     declare~its~language
1066   }
1067   \msg_new:nnn{stex}{warning/deprecated}{
1068     #1~is~deprecated;~please~use~#2~instead!
1069   }
1070   \msg_new:nnn{stex}{error/conflictingmodules}{
1071     Conflicting~imports~for~module~#1
1072   }
1073
1074 \l_stex_current_module_str The current module:
1075 \str_new:N \l_stex_current_module_str
1076
1077 (End definition for \l_stex_current_module_str. This variable is documented on page 54.)
1078
1079 \l_stex_all_modules_seq Stores all available modules
1080 \seq_new:N \l_stex_all_modules_seq
1081
1082 (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
1074 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1075   \str_if_empty:NTF \l_stex_current_module_str
1076   \prg_return_false: \prg_return_true:
1077 }

```

(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
1078 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1079   \prop_if_exist:cTF { c_stex_module_#1_prop }
1080   \prg_return_true: \prg_return_false:
1081 }

```

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

`\stex_add_to_current_module:n` Only allowed within modules:

```

\STEXexport
1082 \cs_new_protected:Nn \stex_add_to_current_module:n {
1083   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1084 }
1085 \cs_new_protected:Npn \STEXexport {
1086   \begingroup
1087   \newlinechar=-1\relax
1088   \endlinechar=-1\relax
1089   %\catcode'\ = 9\relax
1090   \expandafter\endgroup\__stex_modules_export:n
1091 }
1092 \cs_new_protected:Nn \__stex_modules_export:n {
1093   \ignorespaces #1
1094   \stex_add_to_current_module:n { \ignorespaces #1 }
1095   \stex_smsmode_do:
1096 }
1097 \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
1098 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1099   \str_set:Nx \l_tmpa_str { #1 }
1100   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1101 }

```

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

```

\stex_add_import_to_current_module:n
1102 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1103   \str_set:Nx \l_tmpa_str { #1 }
1104   \exp_args:Nno
1105   \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1106     \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1107   }
1108 }

```

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

`\stex_collect_imports:n`

```

1109 \cs_new_protected:Nn \stex_collect_imports:n {
1110   \seq_clear:N \l_stex_collect_imports_seq
1111   \__stex_modules_collect_imports:n {#1}
1112 }
1113 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1114   \seq_map_inline:cn {c_stex_module_#1_imports} {
1115     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1116       \__stex_modules_collect_imports:n { ##1 }
1117     }
1118   }
1119   \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1120     \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1121   }
1122 }

```

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

`\stex_do_up_to_module:n`

```

1123 \int_new:N \l__stex_modules_group_depth_int
1124 \tl_new:N \l__stex_modules_aftergroup_tl
1125 \cs_new_protected:Nn \stex_do_up_to_module:n {
1126   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1127     #1
1128   }{
1129     #1
1130     \expandafter \tl_gset:Nn \expandafter \l__stex_modules_aftergroup_tl \expandafter { \l__
1131       \aftergroup\__stex_modules_aftergroup_do:
1132     }
1133   }
1134   \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1135     \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1136       \l__stex_modules_aftergroup_tl
1137       \tl_clear:N \l__stex_modules_aftergroup_tl
1138     }{
1139       \l__stex_modules_aftergroup_tl
1140       \aftergroup\__stex_modules_aftergroup_do:
1141     }
1142   }
1143   \cs_new_protected:Nn \stex_reset_up_to_module: {
1144
1145     \tl_gset_eq:NN \l__stex_modules_aftergroup_tl \l__stex_modules_aftergroup_outer_tl
1146   }

```

(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).

1147

(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.

```

1148 \str_new:N \l_stex_modules_ns_str
1149 \str_new:N \l_stex_modules_subpath_str
1150 \cs_new_protected:Nn \__stex_modules_compute_namespace:nN {
1151   \str_set:Nx \l_tmpa_str { #1 }
1152   \seq_set_eq:NN \l_tmpa_seq #2
1153   % split off file extension
1154   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1155   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1156   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1157   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1158
1159   \bool_set_true:N \l_tmpa_bool
1160   \bool_while_do:Nn \l_tmpa_bool {
1161     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1162     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1163       {source} { \bool_set_false:N \l_tmpa_bool }
1164     }{}{
1165       \seq_if_empty:NT \l_tmpa_seq {
1166         \bool_set_false:N \l_tmpa_bool
1167       }
1168     }
1169   }
1170
1171   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
1172   \str_if_empty:NTF \l_stex_modules_subpath_str {
1173     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
1174   }{
1175     \str_set:Nx \l_stex_modules_ns_str {
1176       \l_tmpa_str/\l_stex_modules_subpath_str
1177     }
1178   }
1179 }
1180
1181 \cs_new_protected:Nn \stex_modules_current_namespace: {
1182   \str_clear:N \l_stex_modules_subpath_str
1183   \prop_if_exist:NTF \l_stex_current_repository_prop {
1184     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1185     \__stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1186   }{
1187     % split off file extension
1188     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1189     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1190     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1191     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1192     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1193     \str_set:Nx \l_stex_modules_ns_str {
1194       file:/\stex_path_to_string:N \l_tmpa_seq
1195     }
1196   }
1197 }

```

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

## 27.1 The smodule environment

smodule arguments:

```

1198 \keys_define:nn { stex / module } {
1199   title      .tl_set:N      = \smoduletitle ,
1200   type       .str_set_x:N   = \smoduletype ,
1201   id         .str_set_x:N   = \smoduleid ,
1202   deprecate  .str_set_x:N   = \l_stex_module_deprecate_str ,
1203   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1204   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1205   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1206   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1207   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1208   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1209   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1210 }
1211
1212 \cs_new_protected:Nn \__stex_modules_args:n {
1213   \str_clear:N \smoduletitle
1214   \str_clear:N \smoduletype
1215   \str_clear:N \smoduleid
1216   \str_clear:N \l_stex_module_ns_str
1217   \str_clear:N \l_stex_module_deprecate_str
1218   \str_clear:N \l_stex_module_lang_str
1219   \str_clear:N \l_stex_module_sig_str
1220   \str_clear:N \l_stex_module_creators_str
1221   \str_clear:N \l_stex_module_contributors_str
1222   \str_clear:N \l_stex_module_meta_str
1223   \str_clear:N \l_stex_module_srccite_str
1224   \keys_set:nn { stex / module } { #1 }
1225 }
1226
1227 % module parameters here? In the body?
1228
```

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

```

1229 \cs_new_protected:Nn \stex_module_setup:nn {
1230   \tl_gset_eq:NN \l__stex_modules_aftergroup_outer_tl \l__stex_modules_aftergroup_tl
1231   \tl_clear:N \l__stex_modules_aftergroup_tl
1232   \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1233   \str_set:Nx \l_stex_module_name_str { #2 }
1234   \__stex_modules_args:n { #1 }

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

1235   \stex_if_in_module:TF {
1236     % Nested module
1237     \prop_get:cnN {c_stex_module\l_stex_current_module_str_prop}
1238     { ns } \l_stex_module_ns_str
1239     \str_set:Nx \l_stex_module_name_str {
1240       \prop_item:cn {c_stex_module\l_stex_current_module_str_prop}
1241       { name } / \l_stex_module_name_str
1242     }
1243   }{

```

```

1244 % not nested:
1245 \str_if_empty:NT \l_stex_module_ns_str {
1246   \stex_modules_current_namespace:
1247   \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1248   \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1249     / {\l_stex_module_ns_str}
1250   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1251   \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1252     \str_set:Nx \l_stex_module_ns_str {
1253       \stex_path_to_string:N \l_tmpa_seq
1254     }
1255   }
1256 }
1257 }

```

Next, we determine the language of the module:

```

1258 \str_if_empty:NT \l_stex_module_lang_str {
1259   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1260   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1261   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1262   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1263   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1264     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1265       inferred~from~file~name}
1266     \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1267   }
1268 }
1269
1270 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
1271   \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1272     \l_tmpa_str {
1273       \ltx@ifpackageloaded{babel}{
1274         \exp_args:Nx \selectlanguage { \l_tmpa_str }
1275       }{}
1276     } {
1277       \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
1278     }
1279 }}

```

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

```

1280 \str_if_empty:NTF \l_stex_module_sig_str {
1281   \exp_args:Nnx \prop_gset_from_keyval:cn {
1282     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1283   } {
1284     name      = \l_stex_module_name_str ,
1285     ns        = \l_stex_module_ns_str ,
1286     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1287     lang      = \l_stex_module_lang_str ,
1288     sig       = \l_stex_module_sig_str ,
1289     deprecate = \l_stex_module_deprecate_str ,
1290     meta      = \l_stex_module_meta_str
1291   }
1292   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}

```

```

1293 \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1294 \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1295 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1296 \str_if_empty:NT \l_stex_module_meta_str {
1297   \str_set:Nx \l_stex_module_meta_str {
1298     \c_stex_metatheory_ns_str ? Metatheory
1299   }
1300 }
1301 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1302   \bool_set_true:N \l_stex_in_meta_bool
1303   \exp_args:Nx \stex_add_to_current_module:n {
1304     \bool_set_true:N \l_stex_in_meta_bool
1305     \stex_activate_module:n {\l_stex_module_meta_str}
1306     \bool_set_false:N \l_stex_in_meta_bool
1307   }
1308   \stex_activate_module:n {\l_stex_module_meta_str}
1309   \bool_set_false:N \l_stex_in_meta_bool
1310 }
1311 }{
1312   \str_if_empty:NT \l_stex_module_lang_str {
1313     \msg_error:nnxx{stex}{error/siglanguage}{
1314       \l_stex_module_ns_str?\l_stex_module_name_str
1315     }{\l_stex_module_sig_str}
1316   }
1317
1318   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1319   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1320   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1321   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1322   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1323   \str_set:Nx \l_tmpa_str {
1324     \stex_path_to_string:N \l_tmpa_seq /
1325     \l_tmpa_str . \l_stex_module_sig_str .tex
1326   }
1327   \IfFileExists \l_tmpa_str {
1328     \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1329       \str_clear:N \l_stex_current_module_str
1330       \seq_clear:N \l_stex_all_modules_seq
1331       \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1332     }
1333   }{
1334     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1335   }
1336   \stex_if_smsmode:F {
1337     \stex_activate_module:n {
1338       \l_stex_module_ns_str ? \l_stex_module_name_str
1339     }
1340   }
1341   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1342 }
1343 \str_if_empty:NF \l_stex_module_deprecate_str {
1344   \msg_warning:nnxx{stex}{warning/deprecated}{

```

```

1345     Module~\l_stex_current_module_str
1346   }{
1347     \l_stex_module_deprecate_str
1348   }
1349 }
1350 \seq_put_right:Nx \l_stex_all_modules_seq {
1351   \l_stex_module_ns_str ? \l_stex_module_name_str
1352 }
1353 }

```

(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}`

```

1354 \cs_new_protected:Nn \_stex_modules_begin_module: {
1355   \stex_reactivate_macro:N \STEXexport
1356   \stex_reactivate_macro:N \importmodule
1357   \stex_reactivate_macro:N \symdecl
1358   \stex_reactivate_macro:N \notation
1359   \stex_reactivate_macro:N \symdef
1360
1361   \stex_debug:nn{modules}{
1362     New~module:\\
1363     Namespace:~\l_stex_module_ns_str\\
1364     Name:~\l_stex_module_name_str\\
1365     Language:~\l_stex_module_lang_str\\
1366     Signature:~\l_stex_module_sig_str\\
1367     Metatheory:~\l_stex_module_meta_str\\
1368     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1369   }
1370
1371   \stex_if_smsmode:F{
1372     \begin{stex_annotate_env} {theory} {
1373       \l_stex_module_ns_str ? \l_stex_module_name_str
1374     }
1375
1376     \stex_annotate_invisible:nnn{header}{} {
1377       \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1378       \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1379       \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1380         \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1381       }
1382       \str_if_empty:NF \smoduletype {
1383         \stex_annotate:nnn{type}{\smoduletype}{}
1384       }
1385     }
1386   }
1387   % TODO: Inherit metatheory for nested modules?
1388 }
1389 \iffalse \end{stex_annotate_env} \fi %^^A make syntax highlighting work again

```

(End definition for `\_stex_modules_begin_module:.`)



```

1390 \cs_new_protected:Nn \__stex_modules_end_module: {
1391   \stex_debug:nn{modules}{Closing~module~\prop_item:cn {c_stex_module\_l_stex_current_module}}
1392 }

```

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

The core environment

```

1393 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1394 \NewDocumentEnvironment { smodule } { 0{} m } {
1395   \stex_module_setup:nn{#1}{#2}
1396   \par
1397   \stex_if_smsmode:F{
1398     \tl_clear:N \l_tmpa_tl
1399     \clist_map_inline:Nn \smoduletype {
1400       \tl_if_exist:cT {\__stex_modules_smodule_##1_start:}{
1401         \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_modules_smodule_##1_start:}}
1402       }
1403     }
1404     \tl_if_empty:NTF \l_tmpa_tl {
1405       \__stex_modules_smodule_start:
1406     }{
1407       \l_tmpa_tl
1408     }
1409   }
1410   \__stex_modules_begin_module:
1411   \str_if_empty:NF \smoduleid {
1412     \stex_ref_new_doc_target:n \smoduleid
1413   }
1414   \stex_smsmode_do:
1415 } {
1416   \__stex_modules_end_module:
1417   \stex_if_smsmode:F {
1418     \end{stex_annotate_env}
1419     \clist_set:No \l_tmpa_clist \smoduletype
1420     \tl_clear:N \l_tmpa_tl
1421     \clist_map_inline:Nn \l_tmpa_clist {
1422       \tl_if_exist:cT {\__stex_modules_smodule_##1_end:}{
1423         \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_modules_smodule_##1_end:}}
1424       }
1425     }
1426     \tl_if_empty:NTF \l_tmpa_tl {
1427       \__stex_modules_smodule_end:
1428     }{
1429       \l_tmpa_tl
1430     }
1431   }
1432 }

```

**\stexpatchmodule**

```

1433 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1434 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1435
1436 \newcommand\stexpatchmodule[3] [] {

```

```

1437 \str_set:Nx \l_tmpa_str{ #1 }
1438 \str_if_empty:NTF \l_tmpa_str {
1439   \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1440   \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1441 }{
1442   \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1443   \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1444 }
1445 }

```

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

## 27.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n
1446 \NewDocumentCommand \STEXModule { m } {
1447   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1448   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1449   \tl_set:Nn \l_tmpa_tl {
1450     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1451   }
1452   \seq_map_inline:Nn \l_stex_all_modules_seq {
1453     \str_set:Nn \l_tmpb_str { ##1 }
1454     \str_if_eq:eeT { \l_tmpa_str } {
1455       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1456     } {
1457       \seq_map_break:n {
1458         \tl_set:Nn \l_tmpa_tl {
1459           \stex_invoke_module:n { ##1 }
1460         }
1461       }
1462     }
1463   }
1464   \l_tmpa_tl
1465 }
1466
1467 \cs_new_protected:Nn \stex_invoke_module:n {
1468   \stex_debug:nn{modules}{Invoking~module~#1}
1469   \peek_charcode_remove:NTF ! {
1470     \__stex_modules_invoke_uri:nN { #1 }
1471   } {
1472     \peek_charcode_remove:NTF ? {
1473       \__stex_modules_invoke_symbol:nn { #1 }
1474     } {
1475       \msg_error:nnx{stex}{error/syntax}{
1476         ?~or~!~expected~after~
1477         \c_backslash_str STEXModule{#1}
1478       }
1479     }
1480   }
1481 }
1482
1483 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {

```

```

1484 \str_set:Nn #2 { #1 }
1485 }
1486
1487 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1488   \stex_invoke_symbol:n{#1?#2}
1489 }

```

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

`\stex_activate_module:n`

```

1490 \bool_new:N \l_stex_in_meta_bool
1491 \bool_set_false:N \l_stex_in_meta_bool
1492 \cs_new_protected:Nn \stex_activate_module:n {
1493   \stex_debug:nn{modules}{Activating~module~#1}
1494   \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1495     \msg_error:nnn{stex}{error/conflictingmodules}{ #1 }
1496   }
1497   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1498     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1499     \use:c{ c_stex_module_#1_code }
1500   }
1501 }

```

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

```

1502 </package>

```

## Chapter 28

# STEX -Module Inheritance Implementation

```
1503 <*package>
1504
1505 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1506
```

### 28.1 SMS Mode

```
1507 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1508 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1509 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1510 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1511
1512 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1513   \makeatletter
1514   \makeatother
1515   \ExplSyntaxOn
1516   \ExplSyntaxOff
1517   \rustexBREAK
1518 }
1519
1520 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1521   \symdef
1522   \importmodule
1523   \notation
1524   \symdecl
1525   \STEXexport
1526   \inlineass
1527   \inlinedef
1528   \inlineex
1529   \endinput
1530   \setnotation
```

```

1531 \copynotation
1532 }
1533
1534 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1535   \tl_to_str:n {
1536     smodule,
1537     copymodule,
1538     interpretmodule,
1539     sdefinition,
1540     sexample,
1541     sassertion,
1542     sparagraph
1543   }
1544 }

```

(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`

```

1545 \bool_new:N \g__stex_smsmode_bool
1546 \bool_set_false:N \g__stex_smsmode_bool
1547 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1548   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1549 }

```

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

`\_stex_smsmode_in_smsmode:nn`

```

1550 \cs_new_protected:Nn \_stex_smsmode_in_smsmode:nn {
1551   \vbox_set:Nn \l_tmpa_box {
1552     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1553     \bool_gset_true:N \g__stex_smsmode_bool
1554     #2
1555     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1556   }
1557   \box_clear:N \l_tmpa_box
1558 }

```

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

`\stex_file_in_smsmode:nn`

```

1559 \quark_new:N \q__stex_smsmode_break
1560
1561 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1562   \stex_filestack_push:n{#1}
1563   \_stex_smsmode_in_smsmode:nn{#1} {
1564     #2
1565     \everyeof{\q__stex_smsmode_break\noexpand}
1566     \expandafter\expandafter\expandafter
1567     \stex_smsmode_do:
1568     \csname @ @ input\endcsname "#1"\relax
1569   }
1570   \stex_filestack_pop:
1571 }

```

(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:

```

1572 \cs_new_protected:Npn \stex_smsmode_do: {
1573   \stex_if_smsmode:T {
1574     \__stex_smsmode_do:w
1575   }
1576 }
1577 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1578   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
1579     \expandafter\if\expandafter\relax\noexpand#1
1580     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1581   } \else\expandafter\__stex_smsmode_do:w\fi
1582 }{
1583   \__stex_smsmode_do:w % #1
1584 }
1585 }
1586 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1587   \cs_if_eq:NNTF #1 \q__stex_smsmode_break {
1588     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1589       #1\__stex_smsmode_do:w
1590     }{
1591       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1592         #1
1593       }{
1594         \cs_if_eq:NNTF \begin #1 {
1595           \__stex_smsmode_check_begin:n
1596         }{
1597           \cs_if_eq:NNTF \end #1 {
1598             \__stex_smsmode_check_end:n
1599           }{
1600             \__stex_smsmode_do:w
1601           }
1602         }
1603       }
1604     }
1605   }
1606 }
1607
1608 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1609   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1610     \begin{#1}
1611   }{
1612     \__stex_smsmode_do:w
1613   }
1614 }
1615 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1616   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1617     \end{#1}\__stex_smsmode_do:w
1618   }{
1619     \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1620   }
1621 }

```

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

## 28.2 Inheritance

```

1622 <@@=stex_importmodule>

\stex_import_module_uri:nn

1623 \cs_new_protected:Nn \stex_import_module_uri:nn {
1624   \str_set:Nx \l_stex_import_archive_str { #1 }
1625   \str_set:Nn \l_stex_import_path_str { #2 }
1626
1627   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1628   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1629   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1630
1631   \stex_modules_current_namespace:
1632   \bool_lazy_all:nTF {
1633     {\str_if_empty_p:N \l_stex_import_archive_str}
1634     {\str_if_empty_p:N \l_stex_import_path_str}
1635     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1636   }{
1637     \str_set_eq:NN \l_stex_import_path_str \l_stex_modules_subpath_str
1638     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1639   }{
1640     \str_if_empty:NT \l_stex_import_archive_str {
1641       \prop_if_exist:NT \l_stex_current_repository_prop {
1642         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1643       }
1644     }
1645     \str_if_empty:NTF \l_stex_import_archive_str {
1646       \str_if_empty:NF \l_stex_import_path_str {
1647         \str_set:Nx \l_stex_import_ns_str {
1648           \l_stex_module_ns_str / \l_stex_import_path_str
1649         }
1650       }
1651     }{
1652       \stex_require_repository:n \l_stex_import_archive_str
1653       \prop_get:cnN { c_stex_mathhub \l_stex_import_archive_str _manifest_prop } { ns }
1654       \l_stex_import_ns_str
1655       \str_if_empty:NF \l_stex_import_path_str {
1656         \str_set:Nx \l_stex_import_ns_str {
1657           \l_stex_import_ns_str / \l_stex_import_path_str
1658         }
1659       }
1660     }
1661   }
1662 }
```

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

<code>\l_stex_import_name_str</code>	Store the return values of <code>\stex_import_module_uri:nn</code> .
<code>\l_stex_import_archive_str</code>	<code>\str_new:N \l_stex_import_name_str</code>
<code>\l_stex_import_path_str</code>	<code>\str_new:N \l_stex_import_archive_str</code>
<code>\l_stex_import_ns_str</code>	<code>\str_new:N \l_stex_import_path_str</code>

```
1666 \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}}

1667 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1668   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1669
1670     % archive
1671     \str_set:Nx \l_tmpa_str { #2 }
1672     \str_if_empty:NTF \l_tmpa_str {
1673       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1674     } {
1675       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1676       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1677       \seq_put_right:Nn \l_tmpa_seq { source }
1678     }
1679
1680     % path
1681     \str_set:Nx \l_tmpb_str { #3 }
1682     \str_if_empty:NTF \l_tmpb_str {
1683       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1684
1685       \ltx@ifpackageloaded{babel} {
1686         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1687           { \language } \l_tmpb_str {
1688           \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1689         }
1690       } {
1691         \str_clear:N \l_tmpb_str
1692       }
1693
1694       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1695       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1696         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1697       }{
1698         \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1699         \IfFileExists{ \l_tmpa_str.tex }{
1700           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1701         }{
1702           % try english as default
1703           \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1704           \IfFileExists{ \l_tmpa_str.en.tex }{
1705             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1706           }{
1707             \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1708           }
1709         }
1710       }
1711
1712     } {
1713       \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1714       \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1715     }
```



```

1716 \ltx@ifpackageloaded{babel} {
1717   \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1718   { \language } \l_tmpb_str {
1719     \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1720   }
1721 } {
1722   \str_clear:N \l_tmpb_str
1723 }
1724
1725 \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1726
1727 \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1728 \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1729   \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1730 }{
1731   \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1732   \IfFileExists{ \l_tmpa_str/#4.tex }{
1733     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1734   }{
1735     % try english as default
1736     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1737     \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1738       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1739     }{
1740       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1741       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1742         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1743       }{
1744         \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1745         \IfFileExists{ \l_tmpa_str.tex }{
1746           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1747         }{
1748           % try english as default
1749           \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1750           \IfFileExists{ \l_tmpa_str.en.tex }{
1751             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1752           }{
1753             \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1754           }
1755         }
1756       }
1757     }
1758   }
1759 }
1760 }
1761
1762 \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
1763   \seq_clear:N \l_stex_all_modules_seq
1764   \str_clear:N \l_stex_current_module_str
1765   \str_set:Nx \l_tmpb_str { #2 }
1766   \str_if_empty:NF \l_tmpb_str {
1767     \stex_set_current_repository:n { #2 }
1768   }
1769   \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}

```

```

1770     }
1771
1772     \stex_if_module_exists:nF { #1 ? #4 } {
1773       \msg_error:nnx{stex}{error/unknownmodule}{
1774         #1?#4~(in~file~\g__stex_importmodule_file_str)
1775       }
1776     }
1777   }
1778   \stex_activate_module:n { #1 ? #4 }
1779 }

```

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

### `\importmodule`

```

1780 \NewDocumentCommand \importmodule { 0{ } m } {
1781   \stex_import_module_uri:nn { #1 } { #2 }
1782   \stex_debug:nn{modules}{Importing~module:~
1783     \l_stex_import_ns_str ? \l_stex_import_name_str
1784   }
1785   \stex_if_smsmode:F {
1786     \stex_import_require_module:nnnn
1787     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1788     { \l_stex_import_path_str } { \l_stex_import_name_str }
1789     \stex_annotate_invisible:nnn
1790     {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1791   }
1792   \exp_args:Nx \stex_add_to_current_module:n {
1793     \stex_import_require_module:nnnn
1794     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1795     { \l_stex_import_path_str } { \l_stex_import_name_str }
1796   }
1797   \exp_args:Nx \stex_add_import_to_current_module:n {
1798     \l_stex_import_ns_str ? \l_stex_import_name_str
1799   }
1800   \stex_smsmode_do:
1801   \ignorespacesandpars
1802 }
1803 \stex_deactivate_macro:Nn \importmodule {module-environments}

```

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

### `\usemodule`

```

1804 \NewDocumentCommand \usemodule { 0{ } m } {
1805   \stex_if_smsmode:F {
1806     \stex_import_module_uri:nn { #1 } { #2 }
1807     \stex_import_require_module:nnnn
1808     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1809     { \l_stex_import_path_str } { \l_stex_import_name_str }
1810     \stex_annotate_invisible:nnn
1811     {usemodule} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1812   }
1813   \stex_smsmode_do:
1814   \ignorespacesandpars
1815 }

```

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

1816 `\endpackage`

## Chapter 29

# STEX -Symbols Implementation

```
1817 <*package>
1818
1819 %%%%%%%%%% symbols.dtx %%%%%%%%%%
1820
    Warnings and error messages
1821 \msg_new:nnn{stex}{error/wrongargs}{
1822   args~value~in~symbol~declaration~for~#1~
1823   needs~to~be~i,~a,~b~or~B,~but~#2~given
1824 }
1825 \msg_new:nnn{stex}{error/unknownsymbol}{
1826   No~symbol~#1~found!
1827 }
1828 \msg_new:nnn{stex}{error/seqlength}{
1829   Expected~#1~arguments;~got~#2!
1830 }
```

### 29.1 Symbol Declarations

```
1831 <@@=stex_symdecl>

\stex_all_symbols:n Map over all available symbols
1832 \cs_new_protected:Nn \stex_all_symbols:n {
1833   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
1834   \seq_map_inline:Nn \l_stex_all_modules_seq {
1835     \seq_map_inline:cn{c_stex_module_##1_constants}{
1836       \__stex_symdecl_all_symbols_cs{##1?####1}
1837     }
1838   }
1839 }

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

\STEXsymbol
1840 \NewDocumentCommand \STEXsymbol { m } {
1841   \stex_get_symbol:n { #1 }
```

```

1842 \exp_args:No
1843 \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1844 }

```

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

`symdecl` arguments:

```

1845 \keys_define:nn { stex / symdecl } {
1846   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1847   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1848   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1849   type      .tl_set:N = \l_stex_symdecl_type_tl ,
1850   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,
1851   align     .str_set:N = \l_stex_symdecl_align_str , % TODO(?)
1852   gfc       .str_set:N = \l_stex_symdecl_gfc_str , % TODO(?)
1853   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
1854   def       .tl_set:N = \l_stex_symdecl_definiens_tl ,
1855   assoc     .choices:nn =
1856     {bin,binl,binr,pre,conj,pwconj}
1857     {\str_set:Nx \l_stex_symdecl_astype_str {\l_keys_choice_tl}}
1858 }
1859
1860 \bool_new:N \l_stex_symdecl_make_macro_bool
1861
1862 \cs_new_protected:Nn \__stex_symdecl_args:n {
1863   \str_clear:N \l_stex_symdecl_name_str
1864   \str_clear:N \l_stex_symdecl_args_str
1865   \str_clear:N \l_stex_symdecl_deprecate_str
1866   \str_clear:N \l_stex_symdecl_astype_str
1867   \bool_set_false:N \l_stex_symdecl_local_bool
1868   \tl_clear:N \l_stex_symdecl_type_tl
1869   \tl_clear:N \l_stex_symdecl_definiens_tl
1870
1871   \keys_set:nn { stex / symdecl } { #1 }
1872 }

```

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

```

1873
1874 \NewDocumentCommand \symdecl { s m O{} } {
1875   \__stex_symdecl_args:n { #3 }
1876   \IfBooleanTF #1 {
1877     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1878   } {
1879     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1880   }
1881   \stex_symdecl_do:n { #2 }
1882   \stex_smsmode_do:
1883 }
1884
1885 \cs_new_protected:Nn \stex_symdecl_do:nn {
1886   \__stex_symdecl_args:n{#1}
1887   \bool_set_false:N \l_stex_symdecl_make_macro_bool
1888   \stex_symdecl_do:n{#2}
1889 }

```

```

1890
1891 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

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

```

1892 \cs_new_protected:Nn \stex_symdecl_do:n {
1893   \stex_if_in_module:F {
1894     % TODO throw error? some default namespace?
1895   }
1896
1897   \str_if_empty:NT \l_stex_symdecl_name_str {
1898     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1899   }
1900
1901   \prop_if_exist:cT { l_stex_symdecl_
1902     \l_stex_current_module_str ?
1903     \l_stex_symdecl_name_str
1904   }_prop
1905   {
1906     % TODO throw error (beware of circular dependencies)
1907   }
1908
1909   \prop_clear:N \l_tmpa_prop
1910   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
1911   \seq_clear:N \l_tmpa_seq
1912   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1913   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1914
1915   \str_if_empty:NT \l_stex_symdecl_deprecate_str {
1916     \str_if_empty:NF \l_stex_module_deprecate_str {
1917       \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
1918     }
1919   }
1920   \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
1921
1922   \exp_args:No \stex_add_constant_to_current_module:n {
1923     \l_stex_symdecl_name_str
1924   }
1925
1926   % arity/args
1927   \int_zero:N \l_tmpb_int
1928
1929   \bool_set_true:N \l_tmpa_bool
1930   \str_map_inline:Nn \l_stex_symdecl_args_str {
1931     \token_case_meaning:NnF ##1 {
1932       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1933       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1934       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1935       {\tl_to_str:n a} {
1936         \bool_set_false:N \l_tmpa_bool
1937         \int_incr:N \l_tmpb_int
1938       }
1939       {\tl_to_str:n B} {

```

```

1940     \bool_set_false:N \l_tmpa_bool
1941     \int_incr:N \l_tmpb_int
1942   }
1943 }{
1944   \msg_error:nnxx{stex}{error/wrongargs}{
1945     \l_stex_current_module_str ?
1946     \l_stex_symdecl_name_str
1947   }{##1}
1948 }
1949 }
1950 \bool_if:NTF \l_tmpa_bool {
1951   % possibly numeric
1952   \str_if_empty:NTF \l_stex_symdecl_args_str {
1953     \prop_put:Nnn \l_tmpa_prop { args } {}
1954     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1955   }{
1956     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1957     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1958     \str_clear:N \l_tmpa_str
1959     \int_step_inline:nn \l_tmpa_int {
1960       \str_put_right:Nn \l_tmpa_str i
1961     }
1962     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1963   }
1964 } {
1965   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1966   \prop_put:Nnx \l_tmpa_prop { arity }
1967     { \str_count:N \l_stex_symdecl_args_str }
1968 }
1969 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
1970
1971 \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
1972   \prop_put:Nnx \l_tmpa_prop { defined }{ false }
1973 }{
1974   \prop_put:Nnx \l_tmpa_prop { defined }{ true }
1975 }
1976
1977 % semantic macro
1978
1979 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1980   \exp_args:Nx \stex_do_up_to_module:n {
1981     \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1982       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1983     }}
1984   }
1985
1986   \bool_if:NF \l_stex_symdecl_local_bool {
1987     \exp_args:Nx \stex_add_to_current_module:n {
1988       \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1989         \l_stex_current_module_str ? \l_stex_symdecl_name_str
1990       } }
1991     }
1992   }
1993 }

```

```

1994
1995 \stex_debug:nn{symbols}{New~symbol:~
1996   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
1997   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
1998   Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
1999   Definiens:~\exp_not:o { \l_stex_symdecl_definiens_tl }
2000 }
2001
2002 % circular dependencies require this:
2003
2004 \prop_if_exist:cF {
2005   \l_stex_symdecl_
2006   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2007   _prop
2008 } {
2009   \exp_args:Nx \stex_do_up_to_module:n {
2010     \prop_set_from_keyval:cn {
2011       \l_stex_symdecl_
2012       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2013       _prop
2014     } {\prop_to_keyval:N \l_tmpa_prop}
2015     \seq_clear:c {
2016       \l_stex_symdecl_
2017       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2018       _notations
2019     }
2020   }
2021 }
2022
2023
2024
2025 \bool_if:NF \l_stex_symdecl_local_bool {
2026   \exp_args:Nx
2027   \stex_add_to_current_module:n {
2028     \seq_clear:c {
2029       \l_stex_symdecl_
2030       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2031       _notations
2032     }
2033     \prop_set_from_keyval:cn {
2034       \l_stex_symdecl_
2035       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2036       _prop
2037     } {
2038       name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2039       module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2040       type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2041       args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2042       arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2043       assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
2044     }
2045   }
2046 }
2047

```



```

2048 \stex_if_smsmode:F {
2049 % \exp_args:Nx \stex_do_up_to_module:n {
2050 % \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2051 % \l_stex_current_module_str ? \l_stex_symdecl_name_str
2052 % }
2053 % }
2054 \stex_if_do_html:T {
2055 \stex_annotate_invisible:nnn {symdecl} {
2056 \l_stex_current_module_str ? \l_stex_symdecl_name_str
2057 } {
2058 \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}{}}{ $\l_st
2059 \stex_annotate_invisible:nnn{args}{}{
2060 \prop_item:Nn \l_tmpa_prop { args }
2061 }
2062 \stex_annotate_invisible:nnn{macroname}{#1}{}
2063 \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2064 \stex_annotate_invisible:nnn{definiens}{}
2065 { $\l_stex_symdecl_definiens_tl$ }
2066 }
2067 \str_if_empty:NF \l_stex_symdecl_assoc_type_str {
2068 \stex_annotate_invisible:nnn{assoc_type}{\l_stex_symdecl_assoc_type_str}{}
2069 }
2070 }
2071 }
2072 }
2073 }

```

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

`\stex_get_symbol:n`

```

2074 \str_new:N \l_stex_get_symbol_uri_str
2075
2076 \cs_new_protected:Nn \stex_get_symbol:n {
2077 \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2078 \tl_set:Nn \l_tmpa_tl { #1 }
2079 \__stex_symdecl_get_symbol_from_cs:
2080 }{
2081 % argument is a string
2082 % is it a command name?
2083 \cs_if_exist:cTF { #1 }{
2084 \cs_set_eq:Nc \l_tmpa_tl { #1 }
2085 \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2086 \str_if_empty:NTF \l_tmpa_str {
2087 \exp_args:Nx \cs_if_eq:NNTF {
2088 \tl_head:N \l_tmpa_tl
2089 } \stex_invoke_symbol:n {
2090 \__stex_symdecl_get_symbol_from_cs:
2091 }{
2092 \__stex_symdecl_get_symbol_from_string:n { #1 }
2093 }
2094 } {
2095 \__stex_symdecl_get_symbol_from_string:n { #1 }
2096 }
2097 }{

```

```

2098     % argument is not a command name
2099     \__stex_symdecl_get_symbol_from_string:n { #1 }
2100     % \l_stex_all_symbols_seq
2101   }
2102 }
2103 \str_if_eq:eeF {
2104   \prop_item:cn {
2105     l_stex_symdecl\l_stex_get_symbol_uri_str _prop
2106   }{ deprecate }
2107 }{}{
2108   \msg_warning:nnxx{stex}{warning/deprecated}{
2109     Symbol~\l_stex_get_symbol_uri_str
2110   }{
2111     \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{ deprecate }
2112   }
2113 }
2114 }
2115
2116 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2117   \tl_set:Nn \l_tmpa_tl {
2118     \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2119   }
2120   \str_set:Nn \l_tmpa_str { #1 }
2121   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2122
2123   \stex_all_symbols:n {
2124     \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2125       \seq_map_break:n{\seq_map_break:n{
2126         \tl_set:Nn \l_tmpa_tl {
2127           \str_set:Nn \l_stex_get_symbol_uri_str { ##1 }
2128         }
2129       }}
2130     }
2131   }
2132
2133   \l_tmpa_tl
2134 }
2135
2136 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
2137   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2138     { \tl_tail:N \l_tmpa_tl }
2139   \tl_if_single:NTF \l_tmpa_tl {
2140     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2141       \exp_after:wN \str_set:Nn \exp_after:wN
2142         \l_stex_get_symbol_uri_str \l_tmpa_tl
2143     }{
2144       % TODO
2145       % tail is not a single group
2146     }
2147   }{
2148     % TODO
2149     % tail is not a single group
2150   }
2151 }

```

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

## 29.2 Notations

```

2152 <@@=stex_notation>

      notation arguments:
2153 \keys_define:nn { stex / notation } {
2154   lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2155   variant   .tl_set_x:N = \l__stex_notation_variant_str ,
2156   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2157   op        .tl_set:N   = \l__stex_notation_op_tl ,
2158   primary   .bool_set:N = \l__stex_notation_primary_bool ,
2159   primary   .default:n  = {true} ,
2160   unknown   .code:n     = \str_set:Nx
2161             \l__stex_notation_variant_str \l_keys_key_str
2162 }
2163
2164 \cs_new_protected:Nn \_stex_notation_args:n {
2165   \str_clear:N \l__stex_notation_lang_str
2166   \str_clear:N \l__stex_notation_variant_str
2167   \str_clear:N \l__stex_notation_prec_str
2168   \tl_clear:N \l__stex_notation_op_tl
2169   \bool_set_false:N \l__stex_notation_primary_bool
2170
2171   \keys_set:nn { stex / notation } { #1 }
2172 }

\notation

2173 \NewDocumentCommand \notation { s m O{} } {
2174   \_stex_notation_args:n { #3 }
2175   \tl_clear:N \l_stex_symdecl_definiens_tl
2176   \stex_get_symbol:n { #2 }
2177   \tl_set:Nn \l_stex_notation_after_do_tl {
2178     \__stex_notation_final:
2179     \IfBooleanTF#1{
2180       \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2181     }{}
2182     \stex_smsmode_do:\ignorespacesandpars
2183   }
2184   \stex_notation_do:nnnnn
2185   { \prop_item:cn {\l_stex_symdecl\_l_stex_get_symbol_uri_str _prop } { args } }
2186   { \prop_item:cn { \l_stex_symdecl\_l_stex_get_symbol_uri_str _prop } { arity } }
2187   { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2188   { \l__stex_notation_prec_str }
2189 }
2190 \stex_deactivate_macro:Nn \notation {module~environments}

```

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

`\stex_notation_do:nnnnn`

```

2191 \seq_new:N \l__stex_notation_precedences_seq
2192 \tl_new:N \l__stex_notation_opprec_tl
2193 \int_new:N \l__stex_notation_currarg_int

```

```

2194 \tl_new:N \stex_symbol_after_invokation_tl
2195
2196 \cs_new_protected:Nn \stex_notation_do:nnnnn {
2197   \let\l_stex_current_symbol_str\relax
2198   \seq_clear:N \l__stex_notation_precedences_seq
2199   \tl_clear:N \l__stex_notation_opprec_tl
2200   \str_set:Nx \l__stex_notation_args_str { #1 }
2201   \str_set:Nx \l__stex_notation_arity_str { #2 }
2202   \str_set:Nx \l__stex_notation_suffix_str { #3 }
2203   \str_set:Nx \l__stex_notation_prec_str { #4 }
2204
2205   % precedences
2206   \str_if_empty:NTF \l__stex_notation_prec_str {
2207     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2208       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2209     }{
2210       \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2211     }
2212   } {
2213     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2214       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2215       \int_step_inline:nn { \l__stex_notation_arity_str } {
2216         \exp_args:NNo
2217         \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2218       }
2219     }{
2220       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2221       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2222         \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2223         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2224           \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2225             \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2226           \seq_map_inline:Nn \l_tmpa_seq {
2227             \seq_put_right:Nn \l_tmpb_seq { ##1 }
2228           }
2229         }
2230       }{
2231         \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2232           \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2233         }{
2234           \tl_set:No \l__stex_notation_opprec_tl { 0 }
2235         }
2236       }
2237     }
2238   }
2239
2240   \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2241   \int_step_inline:nn { \l__stex_notation_arity_str } {
2242     \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
2243       \exp_args:NNo
2244       \seq_put_right:No \l__stex_notation_precedences_seq {
2245         \l__stex_notation_opprec_tl
2246       }
2247     }

```

```

2248 }
2249 \tl_clear:N \l_stex_notation_dummyargs_tl
2250
2251 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2252   \exp_args:NNe
2253   \cs_set:Npn \l_stex_notation_macrocode_cs {
2254     \_stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2255     { \l__stex_notation_suffix_str }
2256     { \l__stex_notation_opprec_tl }
2257     { \exp_not:n { #5 } }
2258   }
2259   \l_stex_notation_after_do_tl
2260 }{
2261   \str_if_in:NnTF \l__stex_notation_args_str b {
2262     \exp_args:Nne \use:nn
2263     {
2264       \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2265       \cs_set:Npn \l__stex_notation_arity_str } { {
2266         \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2267         { \l__stex_notation_suffix_str }
2268         { \l__stex_notation_opprec_tl }
2269         { \exp_not:n { #5 } }
2270       }}
2271   }{
2272     \str_if_in:NnTF \l__stex_notation_args_str B {
2273       \exp_args:Nne \use:nn
2274       {
2275         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2276         \cs_set:Npn \l__stex_notation_arity_str } { {
2277           \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2278           { \l__stex_notation_suffix_str }
2279           { \l__stex_notation_opprec_tl }
2280           { \exp_not:n { #5 } }
2281         } }
2282     }{
2283       \exp_args:Nne \use:nn
2284       {
2285         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2286         \cs_set:Npn \l__stex_notation_arity_str } { {
2287           \_stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2288           { \l__stex_notation_suffix_str }
2289           { \l__stex_notation_opprec_tl }
2290           { \exp_not:n { #5 } }
2291         } }
2292     }
2293   }
2294
2295   \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2296   \int_zero:N \l__stex_notation_currarg_int
2297   \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2298   \__stex_notation_arguments:
2299 }
2300 }

```

(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

```

2301 \cs_new_protected:Nn \_stex_notation_arguments: {
2302   \int_incr:N \l__stex_notation_currarg_int
2303   \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2304     \l_stex_notation_after_do_tl
2305   }{
2306     \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2307     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2308     \str_if_eq:VnTF \l_tmpa_str a {
2309       \_stex_notation_argument_assoc:n
2310     }{
2311       \str_if_eq:VnTF \l_tmpa_str B {
2312         \_stex_notation_argument_assoc:n
2313       }{
2314         \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2315         \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2316           { \_stex_term_math_arg:nnn
2317             { \int_use:N \l__stex_notation_currarg_int }
2318             { \l_tmpa_str }
2319             { ####\int_use:N \l__stex_notation_currarg_int }
2320           }
2321         }
2322         \_stex_notation_arguments:
2323       }
2324     }
2325   }
2326 }

```

(End definition for `\_stex_notation_arguments:.`)

`\_stex_notation_argument_assoc:n`

```

2327 \cs_new_protected:Nn \_stex_notation_argument_assoc:n {
2328
2329   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2330     {\l__stex_notation_arity_str}{
2331       #1
2332     }
2333   \int_zero:N \l_tmpa_int
2334   \tl_clear:N \l_tmpa_tl
2335   \str_map_inline:Nn \l__stex_notation_args_str {
2336     \int_incr:N \l_tmpa_int
2337     \tl_put_right:Nx \l_tmpa_tl {
2338       \str_if_eq:nnTF {##1}{a}{ {} }{
2339         \str_if_eq:nnTF {##1}{B}{ {} }{
2340           {\_stex_term_arg:nn{\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa_in
2341         }
2342       }
2343     }
2344   }
2345   \exp_after:wN\exp_after:wN\exp_after:wN \def
2346   \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2347   \exp_after:wN\exp_after:wN\exp_after:wN ##
2348   \exp_after:wN\exp_after:wN\exp_after:wN 1
2349   \exp_after:wN\exp_after:wN\exp_after:wN ##

```

```

2350 \exp_after:wN\exp_after:wN\exp_after:wN 2
2351 \exp_after:wN\exp_after:wN\exp_after:wN {
2352   \exp_after:wN \exp_after:wN \exp_after:wN
2353   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2354     \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2355   }
2356 }
2357
2358 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2359 \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
2360   \stex_term_math_assoc_arg:nnnn
2361   { \int_use:N \l__stex_notation_currarg_int }
2362   { \l_tmpa_str }
2363   { ####\int_use:N \l__stex_notation_currarg_int }
2364   { \l_tmpa_cs {####1} {####2} }
2365 } }
2366 \__stex_notation_arguments:
2367 }

```

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

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

```

2368 \cs_new_protected:Nn \__stex_notation_final: {
2369 % \exp_args:Nne \use:nn
2370 % {
2371 % \cs_generate_from_arg_count:cNnn {
2372 %   stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2373 %   \l__stex_notation_suffix_str
2374 %   _cs
2375 % }
2376 % \cs_set:Npn \l__stex_notation_arity_str } { {
2377 %   \exp_after:wN \exp_after:wN \exp_after:wN
2378 %   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2379 %   { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2380 % } }
2381
2382 % \tl_if_empty:NF \l__stex_notation_op_tl {
2383 %   \cs_set:cpx {
2384 %     stex_op_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2385 %     \l__stex_notation_suffix_str
2386 %     _cs
2387 %   } { \exp_not:N \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2388 % }
2389
2390 \exp_args:Nx \stex_do_up_to_module:n {
2391   \cs_generate_from_arg_count:cNnn {
2392     stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2393     \l__stex_notation_suffix_str
2394     _cs
2395   } \cs_set:Npn { \l__stex_notation_arity_str } {
2396     \exp_after:wN \exp_after:wN \exp_after:wN
2397     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2398     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2399   }

```

```

2400 \tl_if_empty:NF \l__stex_notation_op_tl {
2401   \cs_set:cpn {
2402     stex_op_notation_\l_stex_get_symbol_uri_str \c_hash_str
2403     \l__stex_notation_suffix_str
2404     _cs
2405   } { \exp_not:N \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2406 }
2407 }
2408
2409 \exp_args:Ne
2410 \stex_add_to_current_module:n {
2411   \cs_generate_from_arg_count:cNnn {
2412     stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2413     \l__stex_notation_suffix_str
2414     _cs
2415   } \cs_set:Npn {\l__stex_notation_arity_str} {
2416     \exp_after:wN \exp_after:wN \exp_after:wN
2417     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2418     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2419   }
2420 \tl_if_empty:NF \l__stex_notation_op_tl {
2421   \cs_set:cpn {
2422     stex_op_notation_\l_stex_get_symbol_uri_str \c_hash_str
2423     \l__stex_notation_suffix_str
2424     _cs
2425   } { \exp_not:N \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2426 }
2427 }
2428
2429 \stex_debug:nn{symbols}{
2430   Notation~\l__stex_notation_suffix_str
2431   ~for~\l_stex_get_symbol_uri_str^^J
2432   Operator~precedence:~\l__stex_notation_opprec_tl^^J
2433   Argument~precedences:~
2434   \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2435   Notation: \cs_meaning:c {
2436     stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2437     \l__stex_notation_suffix_str
2438     _cs
2439   }
2440 }
2441
2442 \exp_args:Nx
2443 \stex_do_up_to_module:n {
2444   \seq_put_right:cx {
2445     l_stex_symdecl_ \l_stex_get_symbol_uri_str
2446     _notations
2447   } {
2448     \l__stex_notation_suffix_str
2449   }
2450 }
2451 \exp_args:Ne
2452 \stex_add_to_current_module:n {
2453   \seq_put_right:cn {

```



```

2454     \l_stex_symdecl \l_stex_get_symbol_uri_str
2455     _notations
2456   } { \l__stex_notation_suffix_str }
2457 }
2458
2459 \stex_if_smsmode:F {
2460
2461   % HTML annotations
2462   \stex_if_do_html:T {
2463     \stex_annotate_invisible:nnn { notation }
2464     { \l_stex_get_symbol_uri_str } {
2465       \stex_annotate_invisible:nnn { notationfragment }
2466       { \l__stex_notation_suffix_str }{}
2467       \stex_annotate_invisible:nnn { precedence }
2468       { \l__stex_notation_prec_str }{}
2469
2470       \int_zero:N \l_tmpa_int
2471       \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2472       \tl_clear:N \l_tmpa_tl
2473       \int_step_inline:nn { \l__stex_notation_arity_str }{
2474         \int_incr:N \l_tmpa_int
2475         \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2476         \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_r
2477         \str_if_eq:VnTF \l_tmpb_str a {
2478           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2479             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2480             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2481           } }
2482         }{
2483           \str_if_eq:VnTF \l_tmpb_str B {
2484             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2485               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2486               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2487             } }
2488           }{
2489             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2490               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2491             } }
2492           }
2493         }
2494       }
2495       \stex_annotate_invisible:nnn { notationcomp }{}{
2496         \str_set:Nx \l_stex_current_symbol_str {\l_stex_get_symbol_uri_str }
2497         $ \exp_args:Nno \use:nn { \use:c {
2498           stex_notation_ \l_stex_current_symbol_str
2499           \c_hash_str \l__stex_notation_suffix_str _cs
2500         } } { \l_tmpa_tl } $
2501       }
2502     }
2503   }
2504 }
2505 }

```

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

`\setnotation`

```

2506 \keys_define:nn { stex / setnotation } {
2507   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2508   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2509   unknown .code:n = \str_set:Nx
2510     \l__stex_notation_variant_str \l_keys_key_str
2511 }
2512
2513 \cs_new_protected:Nn \stex_setnotation_args:n {
2514   \str_clear:N \l__stex_notation_lang_str
2515   \str_clear:N \l__stex_notation_variant_str
2516   \keys_set:nn { stex / setnotation } { #1 }
2517 }
2518
2519 \cs_new_protected:Nn \stex_setnotation:n {
2520   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1 _notations }
2521     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2522     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2523       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2524     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2525       { \c_hash_str }
2526     \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_#1 _notations }
2527       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2528     \exp_args:Nx \stex_add_to_current_module:n {
2529       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2530         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2531       \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_#1 _notations }
2532         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2533       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2534         { \c_hash_str }
2535     }
2536     \stex_debug:nn {notations}{
2537       Setting~default~notation~
2538       {\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str}~for~
2539       #1 \\
2540       \expandafter\meaning\csname
2541         l_stex_symdecl_#1 _notations\endcsname
2542     }
2543   }{
2544     % todo throw error
2545   }
2546 }
2547
2548 \NewDocumentCommand \setnotation {m m} {
2549   \stex_get_symbol:n { #1 }
2550   \stex_setnotation_args:n { #2 }
2551   \stex_setnotation:n{\l_stex_get_symbol_uri_str}
2552   \stex_smsmode_do:\ignorespacesandpars
2553 }
2554
2555 \cs_new_protected:Nn \stex_copy_notations:nn {
2556   \stex_debug:nn {notations}{
2557     Copying~notations~from~#2~to~#1\\
2558     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}

```

```

2559 }
2560 \tl_clear:N \l_tmpa_tl
2561 \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2562   \tl_put_right:Nn \l_tmpa_tl { {## ##1} }
2563 }
2564 \seq_map_inline:cn {l_stex_symdecl_#2_notations}{
2565   \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2566   \edef \l_tmpa_tl {
2567     \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2568     \exp_after:wN\exp_after:wN\exp_after:wN {
2569       \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2570     }
2571   }
2572   \exp_args:Nx
2573   \stex_do_up_to_module:n {
2574     \seq_put_right:cn{l_stex_symdecl_#1_notations}{##1}
2575     \cs_generate_from_arg_count:cNnn {
2576       stex_notation_ #1 \c_hash_str ##1 _cs
2577     } \cs_set:Npn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }{
2578       \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl}
2579     }
2580   }
2581 }
2582 }
2583
2584 \NewDocumentCommand \copynotation {m m} {
2585   \stex_get_symbol:n { #1 }
2586   \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2587   \stex_get_symbol:n { #2 }
2588   \exp_args:Noo
2589   \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2590   \exp_args:Nx \stex_add_import_to_current_module:n{
2591     \stex_copy_notations:nn {\l_tmpa_str} {\l_stex_get_symbol_uri_str}
2592   }
2593   \stex_smsmode_do:\ignorespacesandpars
2594 }
2595

```

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

**\symdef**

```

2596 \keys_define:nn { stex / symdef } {
2597   name .str_set_x:N = \l_stex_symdecl_name_str ,
2598   local .bool_set:N = \l_stex_symdecl_local_bool ,
2599   args .str_set_x:N = \l_stex_symdecl_args_str ,
2600   type .tl_set:N = \l_stex_symdecl_type_tl ,
2601   def .tl_set:N = \l_stex_symdecl_definiens_tl ,
2602   op .tl_set:N = \l_stex_notation_op_tl ,
2603   lang .str_set_x:N = \l_stex_notation_lang_str ,
2604   variant .str_set_x:N = \l_stex_notation_variant_str ,
2605   prec .str_set_x:N = \l_stex_notation_prec_str ,
2606   assoc .choices:nn =
2607     {bin,binl,binr,pre,conj,pwconj}
2608     {\str_set:Nx \l_stex_symdecl_assoc_type_str {\l_keys_choice_tl}},

```

```

2609   unknown .code:n      = \str_set:Nx
2610         \l__stex_notation_variant_str \l_keys_key_str
2611   }
2612
2613   \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2614     \str_clear:N \l_stex_symdecl_name_str
2615     \str_clear:N \l_stex_symdecl_args_str
2616     \str_clear:N \l_stex_symdecl_assoctype_str
2617     \bool_set_false:N \l_stex_symdecl_local_bool
2618     \tl_clear:N \l_stex_symdecl_type_tl
2619     \tl_clear:N \l_stex_symdecl_definiens_tl
2620     \str_clear:N \l__stex_notation_lang_str
2621     \str_clear:N \l__stex_notation_variant_str
2622     \str_clear:N \l__stex_notation_prec_str
2623     \tl_clear:N \l__stex_notation_op_tl
2624
2625     \keys_set:nn { stex / symdef } { #1 }
2626   }
2627
2628   \NewDocumentCommand \symdef { m O{} } {
2629     \__stex_notation_symdef_args:n { #2 }
2630     \bool_set_true:N \l_stex_symdecl_make_macro_bool
2631     \stex_symdecl_do:n { #1 }
2632     \tl_set:Nn \l_stex_notation_after_do_tl {
2633       \__stex_notation_final:
2634       \stex_smsmode_do:\ignorespacesandpars
2635     }
2636     \str_set:Nx \l_stex_get_symbol_uri_str {
2637       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2638     }
2639     \exp_args:Nx \stex_notation_do:nnnnn
2640       { \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str_prop} { args } }
2641       { \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str_prop} { arity } }
2642       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2643       { \l__stex_notation_prec_str }
2644   }
2645   \stex_deactivate_macro:Nn \symdef {module~environments}

```

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

## 29.3 Variables

```

2646   <@@=stex_variables>
2647
2648   \keys_define:nn { stex / vardef } {
2649     name .str_set_x:N = \l__stex_variables_name_str ,
2650     args .str_set_x:N = \l__stex_variables_args_str ,
2651     type .tl_set:N    = \l__stex_variables_type_tl ,
2652     def  .tl_set:N    = \l__stex_variables_def_tl ,
2653     op   .tl_set:N    = \l__stex_variables_op_tl ,
2654     prec .str_set_x:N = \l__stex_variables_prec_str ,
2655     assoc .choices:nn =
2656       {bin,binl,binr,pre,conj,pwconj}
2657       {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},

```

```

2658 bind .choices:nn =
2659 {forall,exists}
2660 {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2661 }
2662
2663 \cs_new_protected:Nn \__stex_variables_args:n {
2664 \str_clear:N \l__stex_variables_name_str
2665 \str_clear:N \l__stex_variables_args_str
2666 \str_clear:N \l__stex_variables_prec_str
2667 \str_clear:N \l__stex_variables_assoctype_str
2668 \str_clear:N \l__stex_variables_bind_str
2669 \tl_clear:N \l__stex_variables_type_tl
2670 \tl_clear:N \l__stex_variables_def_tl
2671 \tl_clear:N \l__stex_variables_op_tl
2672
2673 \keys_set:nn { stex / vardef } { #1 }
2674 }
2675
2676 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{} } {
2677 \__stex_variables_args:n {#2}
2678 \str_if_empty:NT \l__stex_variables_name_str {
2679 \str_set:Nx \l__stex_variables_name_str { #1 }
2680 }
2681 \prop_clear:N \l_tmpa_prop
2682 \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
2683
2684 \int_zero:N \l_tmpb_int
2685 \bool_set_true:N \l_tmpa_bool
2686 \str_map_inline:Nn \l__stex_variables_args_str {
2687 \token_case_meaning:NnF ##1 {
2688 0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2689 {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2690 {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2691 {\tl_to_str:n a} {
2692 \bool_set_false:N \l_tmpa_bool
2693 \int_incr:N \l_tmpb_int
2694 }
2695 {\tl_to_str:n B} {
2696 \bool_set_false:N \l_tmpa_bool
2697 \int_incr:N \l_tmpb_int
2698 }
2699 }{
2700 \msg_error:nnxx{stex}{error/wrongargs}{
2701 variable~\l__stex_variables_name_str
2702 }{##1}
2703 }
2704 }
2705 \bool_if:NTF \l_tmpa_bool {
2706 % possibly numeric
2707 \str_if_empty:NTF \l__stex_variables_args_str {
2708 \prop_put:Nnn \l_tmpa_prop { args } {}
2709 \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2710 }{
2711 \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }

```

```

2712     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2713     \str_clear:N \l_tmpa_str
2714     \int_step_inline:nn \l_tmpa_int {
2715       \str_put_right:Nn \l_tmpa_str i
2716     }
2717     \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
2718     \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2719   }
2720 } {
2721   \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2722   \prop_put:Nnx \l_tmpa_prop { arity }
2723   { \str_count:N \l__stex_variables_args_str }
2724 }
2725 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
2726 \tl_set:cx { #1 } { \stex_invoke_variable:n { \l__stex_variables_name_str } }
2727
2728 \prop_set_eq:cN { l_stex_variable_\l__stex_variables_name_str _prop } \l_tmpa_prop
2729
2730 \tl_if_empty:NF \l__stex_variables_op_tl {
2731   \cs_set:cpx {
2732     stex_var_op_notation_\l__stex_variables_name_str_cs
2733   } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
2734 }
2735
2736 \tl_set:Nn \l_stex_notation_after_do_tl {
2737   \exp_args:Nne \use:nn {
2738     \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str_cs }
2739     \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
2740   } {{
2741     \exp_after:wN \exp_after:wN \exp_after:wN
2742     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2743     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_symbol
2744     }}
2745   \stex_if_do_html:T {
2746     \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
2747       \stex_annotate_invisible:nnn { precedence }
2748       { \l__stex_variables_prec_str }{}
2749     \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{}}{\l
2750     \stex_annotate_invisible:nnn{args}{}}{ \l__stex_variables_args_str }
2751     \stex_annotate_invisible:nnn{macroname}{#1}{}}
2752     \tl_if_empty:NF \l__stex_variables_def_tl {
2753       \stex_annotate_invisible:nnn{definiens}{}}
2754       { $\l__stex_variables_def_tl$ }
2755     }
2756     \str_if_empty:NF \l__stex_variables_assoc_type_str {
2757       \stex_annotate_invisible:nnn{assoc_type}{\l__stex_variables_assoc_type_str}{}
2758     }
2759     \int_zero:N \l_tmpa_int
2760     \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
2761     \tl_clear:N \l_tmpa_tl
2762     \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } } {{
2763       \int_incr:N \l_tmpa_int
2764       \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
2765       \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables

```

```

2766 \str_if_eq:VnTF \l_tmpb_str a {
2767 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2768 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2769 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2770 } }
2771 }{
2772 \str_if_eq:VnTF \l_tmpb_str B {
2773 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2774 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2775 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2776 } }
2777 }{
2778 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2779 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2780 } }
2781 }
2782 }
2783 }
2784 \stex_annotate_invisible:nnn { notationcomp }{}{
2785 \str_set:Nx \l_stex_current_symbol_str {var://\l__stex_variables_name_str }
2786 $ \exp_args:Nno \use:nn { \use:c {
2787 stex_var_notation_\l__stex_variables_name_str _cs
2788 } } { \l_tmpa_tl } $
2789 }
2790 }
2791 }\ignorespacesandpars
2792 }
2793
2794 \stex_notation_do:nnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
2795 }
2796
2797 \cs_new:Nn \_stex_reset:N {
2798 \tl_if_exist:NTF #1 {
2799 \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
2800 }{
2801 \let \exp_not:N #1 \exp_not:N \undefined
2802 }
2803 }
2804
2805 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
2806 \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
2807 \exp_args:Nnx \use:nn {
2808 % TODO
2809 \stex_annotate_invisible:nnn {vardecls}{\clist_use:Nn\l__stex_variables_names,}{
2810 #2
2811 }
2812 }{
2813 \_stex_reset:N \varnot
2814 \_stex_reset:N \vartype
2815 \_stex_reset:N \vardefi
2816 }
2817 }
2818
2819 \NewDocumentCommand \vardef { s } {

```

```

2820 \IfBooleanTF#1 {
2821   \__stex_variables_do_complex:nn
2822 }{
2823   \__stex_variables_do_simple:nnn
2824 }
2825 }
2826
2827 \NewDocumentCommand \svar { 0{} m }{
2828   \tl_if_empty:nTF {#1}{
2829     \str_set:Nn \l_tmpa_str { #2 }
2830   }{
2831     \str_set:Nn \l_tmpa_str { #1 }
2832   }
2833   \_stex_term_omv:nn {
2834     var://\l_tmpa_str
2835   }{
2836     \exp_args:Nnx \use:nn {
2837       \def\comp{\_varcomp}
2838       \str_set:Nx \l_stex_current_symbol_str { var://\l_tmpa_str }
2839       \comp{ #2 }
2840     }{
2841       \_stex_reset:N \comp
2842       \_stex_reset:N \l_stex_current_symbol_str
2843     }
2844   }
2845 }
2846
2847
2848
2849 \keys_define:nn { stex / varseq } {
2850   name .str_set_x:N = \l__stex_variables_name_str ,
2851   args .int_set:N   = \l__stex_variables_args_int ,
2852   type .tl_set:N    = \l__stex_variables_type_tl ,
2853   mid .tl_set:N     = \l__stex_variables_mid_tl ,
2854   bind .choices:nn =
2855     {forall,exists}
2856     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2857 }
2858
2859 \cs_new_protected:Nn \__stex_variables_seq_args:n {
2860   \str_clear:N \l__stex_variables_name_str
2861   \int_set:Nn \l__stex_variables_args_int 1
2862   \tl_clear:N \l__stex_variables_type_tl
2863   \str_clear:N \l__stex_variables_bind_str
2864
2865   \keys_set:nn { stex / varseq } { #1 }
2866 }
2867
2868 \NewDocumentCommand \varseq {m 0{} m m m}{
2869   \__stex_variables_seq_args:n { #2 }
2870   \str_if_empty:NT \l__stex_variables_name_str {
2871     \str_set:Nx \l__stex_variables_name_str { #1 }
2872   }
2873   \prop_clear:N \l_tmpa_prop

```



```

2874 \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
2875
2876 \seq_set_from_clist:Nn \l_tmpa_seq {#3}
2877 \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
2878   \msg_error:nnxx{stex}{error/seqlength}
2879   {\int_use:N \l__stex_variables_args_int}
2880   {\seq_count:N \l_tmpa_seq}
2881 }
2882 \seq_set_from_clist:Nn \l_tmpb_seq {#4}
2883 \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
2884   \msg_error:nnxx{stex}{error/seqlength}
2885   {\int_use:N \l__stex_variables_args_int}
2886   {\seq_count:N \l_tmpb_seq}
2887 }
2888 \prop_put:Nnn \l_tmpa_prop {starts} {#3}
2889 \prop_put:Nnn \l_tmpa_prop {ends} {#4}
2890
2891 \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
2892   \cs_set:Npn { \int_use:N \l__stex_variables_args_int } { #5 }
2893
2894 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
2895 \int_step_inline:nn \l__stex_variables_args_int {
2896   \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
2897 }
2898 \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
2899 \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
2900 \tl_if_empty:NF \l__stex_variables_mid_tl {
2901   \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
2902   \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
2903 }
2904 \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
2905 \int_step_inline:nn \l__stex_variables_args_int {
2906   \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
2907 }
2908 \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
2909 \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
2910
2911
2912 \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl
2913
2914 \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}
2915
2916 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
2917
2918 \int_step_inline:nn \l__stex_variables_args_int {
2919   \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
2920     \stex_term_math_arg:nnn{##1}{0}{\exp_not:n{####}##1}
2921   }}
2922 }
2923
2924 \tl_set:Nx \l_tmpa_tl {
2925   \stex_term_math_oma:nnnn { varseq://\l__stex_variables_name_str }{}{0}{
2926     \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
2927   }

```

```

2928 }
2929
2930 \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \stex_symbol_after_invokation_tl} }
2931
2932 \exp_args:Nno \use:nn {
2933   \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
2934   \cs_set:Npn {\int_use:N \l__stex_variables_args_int}}{\l_tmpa_tl}
2935
2936 \stex_debug:nn{sequences}{New~Sequence:~
2937   \expandafter\meaning\csname stex_varseq_\l__stex_variables_name_str _cs\endcsname\\~\\
2938   \prop_to_keyval:N \l_tmpa_prop
2939 }
2940
2941 \prop_set_eq:cN {stex_varseq_\l__stex_variables_name_str _prop}\l_tmpa_prop
2942 \ignorespacesandpars
2943 }
2944
2945 </package>

```

## Chapter 30

# STEX -Terms Implementation

```
2946 <*package>
2947
2948 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2949
2950 <@@=stex_terms>
2951
2952 Warnings and error messages
2953 \msg_new:nnn{stex}{error/nonotation}{
2954   Symbol~#1~invoked,~but~has~no~notation~#2!
2955 }
2956 \msg_new:nnn{stex}{error/notationarg}{
2957   Error~in~parsing~notation~#1
2958 }
2959 \msg_new:nnn{stex}{error/noop}{
2960   Symbol~#1~has~no~operator~notation~for~notation~#2
2961 }
2962 \msg_new:nnn{stex}{error/notallowed}{
2963   Symbol~invocation~#1~not~allowed~in~notation~component~of~#2
2964 }
2965
```

### 30.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
2964
2965
2966 \bool_new:N \l_stex_allow_semantic_bool
2967 \bool_set_true:N \l_stex_allow_semantic_bool
2968
2969 \cs_new_protected:Nn \stex_invoke_symbol:n {
2970   \bool_if:NTF \l_stex_allow_semantic_bool {
2971     \str_if_eq:eeF {
2972       \prop_item:cn {
2973         l_stex_symdecl_#1_prop
2974       }{ deprecate }
2975     }
2976   }
2977 }
```

```

2975   }{}{
2976     \msg_warning:nxxx{stex}{warning/deprecated}{
2977       Symbol~#1
2978     }{
2979       \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
2980     }
2981   }
2982   \if_mode_math:
2983     \exp_after:wN \__stex_terms_invoke_math:n
2984   \else:
2985     \exp_after:wN \__stex_terms_invoke_text:n
2986   \fi: { #1 }
2987 }{
2988   \msg_error:nxxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
2989 }
2990 }
2991
2992 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
2993   \peek_charcode_remove:NTF ! {
2994     \__stex_terms_invoke_op_custom:nn {#1}
2995   }{
2996     \__stex_terms_invoke_custom:nn {#1}
2997   }
2998 }
2999
3000 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
3001   \peek_charcode_remove:NTF ! {
3002     % operator
3003     \peek_charcode_remove:NTF * {
3004       % custom op
3005       \__stex_terms_invoke_op_custom:nn {#1}
3006     }{
3007       % op notation
3008       \peek_charcode:NTF [ {
3009         \__stex_terms_invoke_op_notation:nw {#1}
3010       }{
3011         \__stex_terms_invoke_op_notation:nw {#1}[]
3012       }
3013     }
3014   }{
3015     \peek_charcode_remove:NTF * {
3016       \__stex_terms_invoke_custom:nn {#1}
3017       % custom
3018     }{
3019       % normal
3020       \peek_charcode:NTF [ {
3021         \__stex_terms_invoke_notation:nw {#1}
3022       }{
3023         \__stex_terms_invoke_notation:nw {#1}[]
3024       }
3025     }
3026   }
3027 }
3028

```

```

3029
3030 \cs_new_protected:Nn \__stex_terms_invoke_op_custom:nn {
3031   \exp_args:Nnx \use:nn {
3032     \def\comp{\_comp}
3033     \str_set:Nn \l_stex_current_symbol_str { #1 }
3034     \bool_set_false:N \l_stex_allow_semantic_bool
3035     \stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
3036       \comp{ #2 }
3037     }
3038   }{
3039     \stex_reset:N \comp
3040     \stex_reset:N \l_stex_current_symbol_str
3041     \bool_set_true:N \l_stex_allow_semantic_bool
3042   }
3043 }
3044
3045 \keys_define:nn { stex / terms } {
3046   lang .tl_set_x:N = \l_stex_notation_lang_str ,
3047   variant .tl_set_x:N = \l_stex_notation_variant_str ,
3048   unknown .code:n = \str_set:Nx
3049     \l_stex_notation_variant_str \l_keys_key_str
3050 }
3051
3052 \cs_new_protected:Nn \__stex_terms_args:n {
3053   \str_clear:N \l_stex_notation_lang_str
3054   \str_clear:N \l_stex_notation_variant_str
3055
3056   \keys_set:nn { stex / terms } { #1 }
3057 }
3058
3059 \cs_new_protected:Nn \stex_find_notation:nn {
3060   \__stex_terms_args:n { #2 }
3061   \seq_if_empty:cTF {
3062     l_stex_symdecl_ #1 _notations
3063   } {
3064     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
3065   } {
3066     \bool_lazy_all:nTF {
3067       {\str_if_empty_p:N \l_stex_notation_variant_str}
3068       {\str_if_empty_p:N \l_stex_notation_lang_str}
3069     }{
3070       \seq_get_left:cN {l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
3071     }{
3072       \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
3073         \l_stex_notation_variant_str \c_hash_str \l_stex_notation_lang_str
3074       }{
3075         \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3076       }{
3077         \msg_error:nnxx{stex}{error/nonotation}{#1}{
3078           ~\l_stex_notation_variant_str \c_hash_str \l_stex_notation_lang_str
3079         }
3080       }
3081     }
3082   }

```

```

3083 }
3084
3085 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3086   \exp_args:Nnx \use:nn {
3087     \def\comp{\_comp}
3088     \str_set:Nn \l_stex_current_symbol_str { #1 }
3089     \stex_find_notation:nn { #1 }{ #2 }
3090     \bool_set_false:N \l_stex_allow_semantic_bool
3091     \cs_if_exist:cTF {
3092       stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3093     }{
3094       \_stex_term_oms:nnn {
3095         #1 \c_hash_str \l_stex_notation_variant_str
3096       }{ #1 }{
3097         \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3098       }
3099     }{
3100       \int_compare:nNnTF {\prop_item:cn {l_stex_symdecl_#1_prop}{arity}} = 0{
3101         \cs_if_exist:cTF {
3102           stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3103         }{
3104           \tl_set:Nx \stex_symbol_after_invokation_tl {
3105             \_stex_reset:N \comp
3106             \_stex_reset:N \stex_symbol_after_invokation_tl
3107             \_stex_reset:N \l_stex_current_symbol_str
3108             \bool_set_true:N \l_stex_allow_semantic_bool
3109           }
3110           \def\comp{\_comp}
3111           \str_set:Nn \l_stex_current_symbol_str { #1 }
3112           \bool_set_false:N \l_stex_allow_semantic_bool
3113           \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3114         }{
3115           \msg_error:nnxx{stex}{error/nonotation}{#1}{
3116             ~\l_stex_notation_variant_str
3117           }
3118         }
3119       }{
3120         \msg_error:nnxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3121       }
3122     }
3123   }{
3124     \_stex_reset:N \comp
3125     \_stex_reset:N \l_stex_current_symbol_str
3126     \bool_set_true:N \l_stex_allow_semantic_bool
3127   }
3128 }
3129
3130 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {
3131   \stex_find_notation:nn { #1 }{ #2 }
3132   \cs_if_exist:cTF {
3133     stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3134   }{
3135     \tl_set:Nx \stex_symbol_after_invokation_tl {
3136       \_stex_reset:N \comp

```

```

3137     \stex_reset:N \stex_symbol_after_invokation_tl
3138     \stex_reset:N \l_stex_current_symbol_str
3139     \bool_set_true:N \l_stex_allow_semantic_bool
3140   }
3141   \def\comp{\_comp}
3142   \str_set:Nn \l_stex_current_symbol_str { #1 }
3143   \bool_set_false:N \l_stex_allow_semantic_bool
3144   \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3145   }{
3146     \msg_error:nnxx{stex}{error/nonotation}{#1}{
3147       ~\l_stex_notation_variant_str
3148     }
3149   }
3150 }
3151
3152 \prop_new:N \l__stex_terms_custom_args_prop
3153
3154 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3155   \exp_args:Nnx \use:nn {
3156     \bool_set_false:N \l_stex_allow_semantic_bool
3157     \def\comp{\_comp}
3158     \str_set:Nn \l_stex_current_symbol_str { #1 }
3159     \prop_clear:N \l__stex_terms_custom_args_prop
3160     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3161     \prop_get:cnN {
3162       l_stex_symdecl_#1 _prop
3163     }{ args } \l_tmpa_str
3164     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3165     \tl_set:Nn \arg { \__stex_terms_arg: }
3166     \str_if_empty:NTF \l_tmpa_str {
3167       \stex_term oms:nnn {#1}{#1}{#2}
3168     }{
3169       \str_if_in:NnTF \l_tmpa_str b {
3170         \stex_term ombind:nnn {#1}{#1}{#2}
3171       }{
3172         \str_if_in:NnTF \l_tmpa_str B {
3173           \stex_term ombind:nnn {#1}{#1}{#2}
3174         }{
3175           \stex_term oma:nnn {#1}{#1}{#2}
3176         }
3177       }
3178     }
3179     % TODO check that all arguments exist
3180   }{
3181     \stex_reset:N \l_stex_current_symbol_str
3182     \stex_reset:N \arg
3183     \stex_reset:N \comp
3184     \stex_reset:N \l__stex_terms_custom_args_prop
3185     \bool_set_true:N \l_stex_allow_semantic_bool
3186   }
3187 }
3188
3189 \NewDocumentCommand \__stex_terms_arg: { s O{} m }{
3190   \tl_if_empty:nTF {#2}{

```

```

3191 \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
3192 \bool_set_true:N \l_tmpa_bool
3193 \bool_do_while:Nn \l_tmpa_bool {
3194   \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int}
3195   \int_incr:N \l_tmpa_int
3196   }{
3197     \bool_set_false:N \l_tmpa_bool
3198   }
3199 }
3200 }{
3201   \int_set:Nn \l_tmpa_int { #2 }
3202   \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
3203     % TODO throw error
3204   }
3205 }
3206 \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
3207 \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3208   % TODO throw error
3209 }
3210 \bool_set_true:N \l_stex_allow_semantic_bool
3211 \IfBooleanTF#1{
3212   \stex_annotate_invisible:n {
3213     \exp_args:No \_stex_term_arg:nn {\l_stex_current_symbol_str}{#3}
3214   }
3215 }{
3216   \exp_args:No \_stex_term_arg:nn {\l_stex_current_symbol_str}{#3}
3217 }
3218 \bool_set_false:N \l_stex_allow_semantic_bool
3219 }
3220
3221
3222 \cs_new_protected:Nn \_stex_term_arg:nn {
3223   \bool_set_true:N \l_stex_allow_semantic_bool
3224   \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3225   \bool_set_false:N \l_stex_allow_semantic_bool
3226 }
3227
3228 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
3229   \exp_args:Nnx \use:nn
3230   { \int_set:Nn \l__stex_terms_downprec { #2 }
3231     \_stex_term_arg:nn { #1 }{ #3 }
3232   }
3233   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3234 }

```

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

`\_stex_term_math_assoc_arg:nnnn`

```

3235 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
3236   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3237   \tl_set:Nn \l_tmpb_tl {\_stex_term_math_arg:nnn{#1}{#2}}
3238   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 }}{
3239     \expandafter\if\expandafter\relax\noexpand#3
3240     \expandafter\_stex_terms_math_assoc_arg_maybe_sequence:N\expandafter#3

```



```

3241 \else\expandafter\__stex_terms_math_assoc_arg_simple:n\expandafter#3\fi
3242 }{
3243 \__stex_terms_math_assoc_arg_simple:n{#3}
3244 }
3245 }
3246
3247 \cs_new_protected:Nn \__stex_terms_math_assoc_arg_maybe_sequence:N {
3248 \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
3249 \str_if_empty:NTF \l_tmpa_str {
3250 \exp_args:Nx \cs_if_eq:NNTF {
3251 \tl_head:N #1
3252 } \stex_invoke_sequence:n {
3253 \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
3254 \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
3255 \tl_set:Nx \l_tmpa_tl {\prop_item:cn {stex_varseq \l_tmpa_str _prop}{notation}}
3256 \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
3257 \tl_set:Nx \l_tmpa_tl {\exp_not:N \exp_not:n{
3258 \exp_not:n{\exp_args:Nnx \use:nn} {
3259 \exp_not:n {
3260 \def\comp{\_varcomp}
3261 \str_set:Nn \l_stex_current_symbol_str
3262 } {varseq://\l_tmpa_str}
3263 \exp_not:n{ ##1 }
3264 }{
3265 \exp_not:n {
3266 \_stex_reset:N \comp
3267 \_stex_reset:N \l_stex_current_symbol_str
3268 }
3269 }
3270 }}}
3271 \exp_args:Nno \use:nn {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
3272 \seq_reverse:N \l_tmpa_seq
3273 \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
3274 \seq_map_inline:Nn \l_tmpa_seq {
3275 \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3276 \exp_args:Nno
3277 \l_tmpa_cs { ##1 } \l_tmpa_tl
3278 }
3279 }
3280 \tl_set:Nx \l_tmpa_tl {
3281 \_stex_term_omv:nn {varseq://\l_tmpa_str}{
3282 \exp_args:No \exp_not:n \l_tmpa_tl
3283 }
3284 }
3285 \exp_args:No\l_tmpb_tl\l_tmpa_tl
3286 }{
3287 \__stex_terms_math_assoc_arg_simple:n { #1 }
3288 }
3289 } {
3290 \__stex_terms_math_assoc_arg_simple:n { #1 }
3291 }
3292 }
3293 }
3294

```

```

3295 \cs_new_protected:Nn \__stex_terms_math_assoc_arg_simple:n {
3296   \clist_set:Nn \l_tmpa_clist{ #1 }
3297   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3298     \tl_set:Nn \l_tmpa_tl { #1 }
3299   }{
3300     \clist_reverse:N \l_tmpa_clist
3301     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3302
3303     \clist_map_inline:Nn \l_tmpa_clist {
3304       \exp_args:NNNo \exp_args:NNNo \tl_set:No \l_tmpa_tl {
3305         \exp_args:Nno
3306         \l_tmpa_cs { ##1 } \l_tmpa_tl
3307       }
3308     }
3309   }
3310   \exp_args:No\l_tmpb_tl\l_tmpa_tl
3311 }

```

(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
3312 \tl_const:Nx \infprec {\int_use:N \c_max_int}
3313 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
3314 \int_new:N \l__stex_terms_downprec
3315 \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
3316 \tl_set:Nn \l__stex_terms_left_bracket_str (
3317 \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

```

3318 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
3319   \bool_if:NTF \l__stex_terms_brackets_done_bool {
3320     \bool_set_false:N \l__stex_terms_brackets_done_bool
3321     #2
3322   } {
3323     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
3324       \bool_if:NTF \l_stex_inarray_bool { #2 }{
3325         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
3326         \dobrackets { #2 }
3327       }
3328     }{ #2 }
3329   }
3330 }

```

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

### `\dobrackets`

```

3331 \bool_new:N \l__stex_terms_brackets_done_bool
3332 %\RequirePackage{scalerel}
3333 \cs_new_protected:Npn \dobrackets #1 {
3334   %\ThisStyle{\if D\m@switch
3335   %   \exp_args:Nnx \use:nn
3336   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
3337   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
3338   % \else
3339   \exp_args:Nnx \use:nn
3340   {
3341     \bool_set_true:N \l__stex_terms_brackets_done_bool
3342     \int_set:Nn \l__stex_terms_downprec \infpref
3343     \l__stex_terms_left_bracket_str
3344     #1
3345   }
3346   {
3347     \bool_set_false:N \l__stex_terms_brackets_done_bool
3348     \l__stex_terms_right_bracket_str
3349     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3350   }
3351   %\fi}
3352 }
```

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

### `\withbrackets`

```

3353 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
3354   \exp_args:Nnx \use:nn
3355   {
3356     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
3357     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
3358     #3
3359   }
3360   {
3361     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
3362     {\l__stex_terms_left_bracket_str}
3363     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
3364     {\l__stex_terms_right_bracket_str}
3365   }
3366 }
```

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

### `\STEXinvisible`

```

3367 \cs_new_protected:Npn \STEXinvisible #1 {
3368   \stex_annotate_invisible:n { #1 }
3369 }
```

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

OMDoc terms:

`\_stex_term_math_oms:nnnn`

```
3370 \cs_new_protected:Nn \_stex_term_oms:nnn {
3371   \stex_annotate:nnn{ OMID }{ #2 }{
3372     \stex_highlight_term:nn { #1 } { #3 }
3373   }
3374 }
3375
3376 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
3377   \__stex_terms_maybe_brackets:nn { #3 }{
3378     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3379   }
3380 }
```

*(End definition for \\_stex\_term\_math\_oms:nnnn. This function is documented on page 62.)*

`\_stex_term_math_omv:nn`

```
3381 \cs_new_protected:Nn \_stex_term_omv:nn {
3382   \stex_annotate:nnn{ OMV }{ #1 }{
3383     \stex_highlight_term:nn { #1 } { #2 }
3384   }
3385 }
```

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

`\_stex_term_math_oma:nnnn`

```
3386 \cs_new_protected:Nn \_stex_term_oma:nnn {
3387   \stex_annotate:nnn{ OMA }{ #2 }{
3388     \stex_highlight_term:nn { #1 } { #3 }
3389   }
3390 }
3391
3392 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
3393   \__stex_terms_maybe_brackets:nn { #3 }{
3394     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3395   }
3396 }
```

*(End definition for \\_stex\_term\_math\_oma:nnnn. This function is documented on page 62.)*

`\_stex_term_math_omb:nnnn`

```
3397 \cs_new_protected:Nn \_stex_term_ombind:nnn {
3398   \stex_annotate:nnn{ OMBIND }{ #2 }{
3399     \stex_highlight_term:nn { #1 } { #3 }
3400   }
3401 }
3402
3403 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
3404   \__stex_terms_maybe_brackets:nn { #3 }{
3405     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3406   }
3407 }
```

*(End definition for \\_stex\_term\_math\_omb:nnnn. This function is documented on page 62.)*

**\symref**  
**\symname**

```

3408 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3409
3410 \keys_define:nn { stex / symname } {
3411   pre      .tl_set_x:N = \l__stex_terms_pre_tl ,
3412   post     .tl_set_x:N = \l__stex_terms_post_tl ,
3413   root     .tl_set_x:N = \l__stex_terms_root_tl
3414 }
3415
3416 \cs_new_protected:Nn \stex_symname_args:n {
3417   \tl_clear:N \l__stex_terms_post_tl
3418   \tl_clear:N \l__stex_terms_pre_tl
3419   \tl_clear:N \l__stex_terms_root_str
3420   \keys_set:nn { stex / symname } { #1 }
3421 }
3422
3423 \NewDocumentCommand \symref { m m }{
3424   \let\compemph_uri_prev:\compemph@uri
3425   \let\compemph@uri\symrefemph@uri
3426   \STEXsymbol{#1}!\{ #2 }
3427   \let\compemph@uri\compemph_uri_prev:
3428 }
3429
3430 \NewDocumentCommand \synonym { 0{} m m }{
3431   \stex_symname_args:n { #1 }
3432   \let\compemph_uri_prev:\compemph@uri
3433   \let\compemph@uri\symrefemph@uri
3434   % TODO
3435   \STEXsymbol{#2}!\{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
3436   \let\compemph@uri\compemph_uri_prev:
3437 }
3438
3439 \NewDocumentCommand \symname { 0{} m }{
3440   \stex_symname_args:n { #1 }
3441   \stex_get_symbol:n { #2 }
3442   \str_set:Nx \l_tmpa_str {
3443     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3444   }
3445   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3446
3447   \let\compemph_uri_prev:\compemph@uri
3448   \let\compemph@uri\symrefemph@uri
3449   \exp_args:NNx \use:nn
3450   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\{
3451     \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
3452   } }
3453   \let\compemph@uri\compemph_uri_prev:
3454 }
3455
3456 \NewDocumentCommand \Symname { 0{} m }{
3457   \stex_symname_args:n { #1 }
3458   \stex_get_symbol:n { #2 }
3459   \str_set:Nx \l_tmpa_str {
3460     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }

```

```

3461 }
3462 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {-}
3463 \let\compemph_uri_prev:\compemph@uri
3464 \let\compemph@uri\symrefemph@uri
3465 \exp_args:NNx \use:nn
3466 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!{
3467   \exp_after:wN \stex_capitalize:n \l_tmpa_str
3468   \l__stex_terms_post_tl
3469 } }
3470 \let\compemph@uri\compemph_uri_prev:
3471 }

```

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

### 30.3 Notation Components

```

3472 <@@=stex_notationcomps>

```

`\stex_highlight_term:nn`

```

3473 \cs_new_protected:Nn \stex_highlight_term:nn {
3474   #2
3475 }
3476
3477 \cs_new_protected:Nn \stex_unhighlight_term:n {
3478   % \latexml_if:TF {
3479   %   #1
3480   % } {
3481   %   \rustex_if:TF {
3482   %     #1
3483   %   } {
3484   %     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
3485   %   }
3486   % }
3487 }

```

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

```

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

3488 \cs_new_protected:Npn \_comp #1 {
3489   \str_if_empty:NF \l_stex_current_symbol_str {
3490     \rustex_if:TF {
3491       \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
3492     }{
3493       \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3494     }
3495   }
3496 }
3497
3498 \cs_new_protected:Npn \_varcomp #1 {
3499   \str_if_empty:NF \l_stex_current_symbol_str {
3500     \rustex_if:TF {
3501       \stex_annotate:nnn { varcomp }{ \l_stex_current_symbol_str }{ #1 }
3502     }{
3503       \exp_args:Nnx \varemp@uri { #1 } { \l_stex_current_symbol_str }

```

```

3504     }
3505   }
3506 }
3507
3508 \def\comp{\_comp}
3509
3510 \cs_new_protected:Npn \compemph@uri #1 #2 {
3511   \compemph{ #1 }
3512 }
3513
3514
3515 \cs_new_protected:Npn \compemph #1 {
3516   #1
3517 }
3518
3519 \cs_new_protected:Npn \defemph@uri #1 #2 {
3520   \defemph{#1}
3521 }
3522
3523 \cs_new_protected:Npn \defemph #1 {
3524   \textbf{#1}
3525 }
3526
3527 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3528   \symrefemph{#1}
3529 }
3530
3531 \cs_new_protected:Npn \symrefemph #1 {
3532   \textbf{#1}
3533 }
3534
3535 \cs_new_protected:Npn \varemp@uri #1 #2 {
3536   \varemp{#1}
3537 }
3538
3539 \cs_new_protected:Npn \varemp #1 {
3540   #1
3541 }

```

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

## **\ellipses**

```

3542 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix 3543 \bool_new:N \l_stex_inarray_bool
\parrayline 3544 \bool_set_false:N \l_stex_inarray_bool
\parraylineh 3545 \NewDocumentCommand \parray { m m } {
\parraycell 3546   \begingroup
3547   \bool_set_true:N \l_stex_inarray_bool
3548   \begin{array}{#1}
3549     #2
3550   \end{array}

```

```

3551 \endgroup
3552 }
3553
3554 \NewDocumentCommand \prmatrix { m } {
3555   \begingroup
3556   \bool_set_true:N \l_stex_inarray_bool
3557   \begin{matrix}
3558     #1
3559   \end{matrix}
3560 \endgroup
3561 }
3562
3563 \def \maybepline {
3564   \bool_if:NT \l_stex_inarray_bool {\hline}
3565 }
3566
3567 \def \parrayline #1 #2 {
3568   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3569 }
3570
3571 \def \pmrow #1 { \parrayline{}{ #1 } }
3572
3573 \def \parraylineh #1 #2 {
3574   #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3575 }
3576
3577 \def \parraycell #1 {
3578   #1 \bool_if:NT \l_stex_inarray_bool {&}
3579 }

```

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

## 30.4 Variables

```

3580 <@@=stex_variables>

```

\stex\_invoke\_variable:n Invokes a variable

```

3581 \cs_new_protected:Nn \stex_invoke_variable:n {
3582   \if_mode_math:
3583     \exp_after:wN \__stex_variables_invoke_math:n
3584   \else:
3585     \exp_after:wN \__stex_variables_invoke_text:n
3586   \fi: {#1}
3587 }
3588
3589 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
3590   %TODO
3591 }
3592
3593
3594 \cs_new_protected:Nn \__stex_variables_invoke_math:n {
3595   \peek_charcode_remove:NTF ! {
3596     \peek_charcode_remove:NTF ! {
3597       \peek_charcode:NTF [ {

```



```

3598     \__stex_variables_invoke_op_custom:nw
3599   }{
3600     % TODO throw error
3601   }
3602 }{
3603   \__stex_variables_invoke_op:n { #1 }
3604 }
3605 }{
3606   \peek_charcode_remove:NTF * {
3607     \__stex_variables_invoke_text:n { #1 }
3608   }{
3609     \__stex_variables_invoke_math_ii:n { #1 }
3610   }
3611 }
3612 }
3613
3614 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
3615   \cs_if_exist:cTF {
3616     stex_var_op_notation_ #1 _cs
3617   }{
3618     \exp_args:Nnx \use:nn {
3619       \def\comp{\_varcomp}
3620       \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3621       \_stex_term_omv:nn { var://#1 }{
3622         \use:c{stex_var_op_notation_ #1 _cs }
3623       }
3624     }{
3625       \_stex_reset:N \comp
3626       \_stex_reset:N \l_stex_current_symbol_str
3627     }
3628   }{
3629     \int_compare:nNnTF {\prop_item:cn {\l_stex_variable_#1_prop}{arity}} = 0{
3630       \__stex_variables_invoke_math_ii:n {#1}
3631     }{
3632       \msg_error:nnxx{stex}{error/noop}{variable~#1}{-}
3633     }
3634   }
3635 }
3636
3637 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n #1 {
3638   \cs_if_exist:cTF {
3639     stex_var_notation_#1_cs
3640   }{
3641     \tl_set:Nx \stex_symbol_after_invokation_tl {
3642       \_stex_reset:N \comp
3643       \_stex_reset:N \stex_symbol_after_invokation_tl
3644       \_stex_reset:N \l_stex_current_symbol_str
3645       \bool_set_true:N \l_stex_allow_semantic_bool
3646     }
3647     \def\comp{\_varcomp}
3648     \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3649     \bool_set_false:N \l_stex_allow_semantic_bool
3650     \use:c{stex_var_notation_#1_cs}
3651   }{

```

```

3652 \msg_error:nxxx{stex}{error/nonotation}{variable~#1}{s}
3653 }
3654 }

```

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

## 30.5 Sequences

```

3655 <@@=stex_sequences>
3656
3657 \cs_new_protected:Nn \stex_invoke_sequence:n {
3658   \peek_charcode_remove:NTF ! {
3659     \stex_term_omv:nn {varseq://#1}{
3660       \exp_args:Nnx \use:nn {
3661         \def\comp{\_varcomp}
3662         \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3663         \prop_item:cn{stex_varseq_#1_prop}{notation}
3664       }{
3665         \stex_reset:N \comp
3666         \stex_reset:N \l_stex_current_symbol_str
3667       }
3668     }
3669   }{
3670     \bool_set_false:N \l_stex_allow_semantic_bool
3671     \def\comp{\_varcomp}
3672     \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3673     \tl_set:Nx \stex_symbol_after_invokation_tl {
3674       \stex_reset:N \comp
3675       \stex_reset:N \stex_symbol_after_invokation_tl
3676       \stex_reset:N \l_stex_current_symbol_str
3677       \bool_set_true:N \l_stex_allow_semantic_bool
3678     }
3679     \use:c { stex_varseq_#1_cs }
3680   }
3681 }
3682 </package>

```

## Chapter 31

# STEX -Structural Features Implementation

```
3683 ⟨*package⟩
3684
3685 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3686
3687
3688 Warnings and error messages
3689 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3690   Symbol~#1~can~not~be~assigned~in~copymodule~#2
3691 }
3692 \msg_new:nnn{stex}{error/interpretmodule/noddefinens}{
3693   Symbol~#1~not~assigned~in~interpretmodule~#2
3694 }
3695 \msg_new:nnn{stex}{error/unknownstructure}{
3696   No~structure~#1~found!
3697 }
3698 \msg_new:nnn{stex}{error/unknownfield}{
3699   No~field~#1~in~instance~#2~found!~\#3
3700 }
3701 \msg_new:nnn{stex}{error/keyval}{
3702   Invalid~key=value~pair~#1
3703 }
3704 \msg_new:nnn{stex}{error/instantiate/missing}{
3705   Assignments~missing~in~instantiate:~#1
3706 }
3707 \msg_new:nnn{stex}{error/incompatible}{
3708   Incompatible~signature:~#1~(#2)~and~#3~(#4)
3709 }
3710
3711
```

## 31.1 Imports with modification

```

3712 <@@=stex_copymodule>
3713 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
3714   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3715     \tl_set:Nn \l_tmpa_tl { #1 }
3716     \__stex_copymodule_get_symbol_from_cs:
3717   }{
3718     % argument is a string
3719     % is it a command name?
3720     \cs_if_exist:cTF { #1 }{
3721       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3722       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3723       \str_if_empty:NNTF \l_tmpa_str {
3724         \exp_args:Nx \cs_if_eq:NNTF {
3725           \tl_head:N \l_tmpa_tl
3726         } \stex_invoke_symbol:n {
3727           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
3728         }{
3729           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3730         }
3731       } {
3732         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3733       }
3734     }{
3735       % argument is not a command name
3736       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3737       % \l_stex_all_symbols_seq
3738     }
3739   }
3740 }
3741
3742 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
3743   \str_set:Nn \l_tmpa_str { #1 }
3744   \bool_set_false:N \l_tmpa_bool
3745   \bool_if:NF \l_tmpa_bool {
3746     \tl_set:Nn \l_tmpa_tl {
3747       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
3748     }
3749     \str_set:Nn \l_tmpa_str { #1 }
3750     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3751     \seq_map_inline:Nn #2 {
3752       \str_set:Nn \l_tmpb_str { ##1 }
3753       \str_if_eq:eeT { \l_tmpa_str } {
3754         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3755       } {
3756         \seq_map_break:n {
3757           \tl_set:Nn \l_tmpa_tl {
3758             \str_set:Nn \l_stex_get_symbol_uri_str {
3759               ##1
3760             }
3761           }
3762         }
3763       }

```

```

3764     }
3765     \l_tmpa_tl
3766   }
3767 }
3768
3769 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
3770   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3771     { \tl_tail:N \l_tmpa_tl }
3772   \tl_if_single:NTF \l_tmpa_tl {
3773     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3774       \exp_after:wN \str_set:Nn \exp_after:wN
3775         \l_stex_get_symbol_uri_str \l_tmpa_tl
3776       \__stex_copymodule_get_symbol_check:n { #1 }
3777     }{
3778       % TODO
3779       % tail is not a single group
3780     }
3781   }{
3782     % TODO
3783     % tail is not a single group
3784   }
3785 }
3786
3787 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
3788   \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
3789     \msg_error:nxxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3790       :~\seq_use:Nn #1 {,~}
3791     }
3792   }
3793 }
3794
3795 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
3796   \stex_import_module_uri:nn { #1 } { #2 }
3797   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3798   \stex_import_require_module:nnnn
3799     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3800     { \l_stex_import_path_str } { \l_stex_import_name_str }
3801   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3802   \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
3803   \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
3804   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
3805     \seq_map_inline:cn {c_stex_module_###_constants}{
3806       \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
3807         ##1 ? ####1
3808       }
3809     }
3810   }
3811   \seq_clear:N \l_tmpa_seq
3812   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3813     name      = \l_stex_current_copymodule_name_str ,
3814     module    = \l_stex_current_module_str ,
3815     from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3816     includes  = \l_tmpa_seq ,
3817     fields    = \l_tmpa_seq

```

```

3818 }
3819 \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3820 as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3821 \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_seq {,
3822 \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
3823 \stex_if_smsmode:F {
3824 \begin{stex_annotate_env} {#4} {
3825 \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3826 }
3827 \stex_annotate_invisible:nnn{from}{\l_stex_import_ns_str ?\l_stex_import_name_str}{}
3828 }
3829 \bool_set_eq:NN \l__stex_copymodule_oldhtml_bool \_stex_html_do_output_bool
3830 \bool_set_false:N \_stex_html_do_output_bool
3831 }
3832 \cs_new_protected:Nn \stex_copymodule_end:n {
3833 \def \l_tmpa_cs ##1 ##2 {#1}
3834 \bool_set_eq:NN \_stex_html_do_output_bool \l__stex_copymodule_oldhtml_bool
3835 \tl_clear:N \l_tmpa_tl
3836 \tl_clear:N \l_tmpb_tl
3837 \prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3838 \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
3839 \seq_map_inline:cn {c_stex_module_##1_constants}{
3840 \tl_clear:N \l_tmpc_tl
3841 \l_tmpa_cs{##1}{####1}
3842 \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
3843 \tl_put_right:Nx \l_tmpa_tl {
3844 \prop_set_from_keyval:cn {
3845 l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_name_str}
3846 }{
3847 \exp_after:wN \prop_to_keyval:N \csname
3848 l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_name_str}
3849 \endcsname
3850 }
3851 \seq_clear:c {
3852 l_stex_symdecl_
3853 \l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_name_str}
3854 _notations
3855 }
3856 }
3857 \tl_put_right:Nx \l_tmpc_tl {
3858 \stex_copy_notations:nn {\l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_name_str}
3859 \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?####1_name_str}
3860 }
3861 \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_name_str}
3862 \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
3863 \tl_put_right:Nx \l_tmpc_tl {
3864 \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1?####1_name_str}
3865 }
3866 \tl_put_right:Nx \l_tmpa_tl {
3867 \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
3868 \stex_invoke_symbol:n {
3869 \l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_name_str}
3870 }
3871 }

```

```

3872     }
3873   }
3874 }{
3875   \tl_put_right:Nx \l_tmpc_tl {
3876     \stex_copy_notations:nn {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str}
3877   }
3878   \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
3879   \prop_put:Nnx \l_tmpa_prop { name }{ \l_stex_current_copymodule_name_str / ####1 }
3880   \prop_put:Nnx \l_tmpa_prop { module }{ \l_stex_current_module_str }
3881   \tl_put_right:Nx \l_tmpa_tl {
3882     \prop_set_from_keyval:cn {
3883       l_stex_symdecl_ \l_stex_current_module_str ? \l_stex_current_copymodule_name_str
3884     }{
3885       \prop_to_keyval:N \l_tmpa_prop
3886     }
3887     \seq_clear:c {
3888       l_stex_symdecl_
3889       \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3890       _notations
3891     }
3892   }
3893   \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str}
3894   \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
3895     \tl_put_right:Nx \l_tmpc_tl {
3896       \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}
3897     }
3898     \tl_put_right:Nx \l_tmpa_tl {
3899       \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
3900         \stex_invoke_symbol:n {
3901           \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3902         }
3903       }
3904     }
3905   }
3906 }
3907 \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
3908   \tl_put_right:Nx \l_tmpc_tl {
3909     \stex_annotate_invisible:nnn{definiens}{\use:c{l__stex_copymodule_copymodule_##1?####1_def_tl}}
3910   }
3911 }
3912 \tl_put_right:Nx \l_tmpb_tl {
3913   \stex_annotate:nnn{assignment} {##1?####1} { \l_tmpc_tl }
3914 }
3915 }
3916 }
3917 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3918 \tl_put_left:Nx \l_tmpa_tl {
3919   \prop_set_from_keyval:cn {
3920     l_stex_copymodule_ \l_stex_current_module_str? \l_stex_current_copymodule_name_str _prop
3921   }{
3922     \prop_to_keyval:N \l_stex_current_copymodule_prop
3923   }
3924 }
3925 \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl

```

```

3926 \stex_debug:nn{copymodule}{result:\meaning \l_tmpa_tl}
3927 \exp_args:Nx \stex_do_up_to_module:n {
3928   \exp_args:No \exp_not:n \l_tmpa_tl
3929 }
3930 \l_tmpb_tl
3931 \stex_if_smsmode:F {
3932   \end{stex_annotate_env}
3933 }
3934 }
3935
3936 \NewDocumentEnvironment {copymodule} { 0{} m m}{
3937   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ structure }
3938   \stex_deactivate_macro:Nn \symdecl {module~environments}
3939   \stex_deactivate_macro:Nn \symdef {module~environments}
3940   \stex_deactivate_macro:Nn \notation {module~environments}
3941   \stex_reactivate_macro:N \assign
3942   \stex_reactivate_macro:N \renamedekl
3943   \stex_reactivate_macro:N \donotcopy
3944   \stex_smsmode_do:
3945 }{
3946   \stex_copymodule_end:n {}
3947 }
3948
3949 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
3950   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ realization }
3951   \stex_deactivate_macro:Nn \symdecl {module~environments}
3952   \stex_deactivate_macro:Nn \symdef {module~environments}
3953   \stex_deactivate_macro:Nn \notation {module~environments}
3954   \stex_reactivate_macro:N \assign
3955   \stex_reactivate_macro:N \renamedekl
3956   \stex_reactivate_macro:N \donotcopy
3957   \stex_smsmode_do:
3958 }{
3959   \stex_copymodule_end:n {
3960     \tl_if_exist:cF {
3961       l__stex_copymodule_copymodule_##1?##2_def_tl
3962     }{
3963       \str_if_eq:eeF {
3964         \prop_item:cn{
3965           l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
3966         }{ true }{
3967           \msg_error:nnxx{stex}{error/interpretmodule/nodéfiniens}{
3968             ##1?##2
3969           }{\l_stex_current_copymodule_name_str}
3970         }
3971       }
3972     }
3973   }
3974
3975 \NewDocumentCommand \donotcopy { 0{} m}{
3976   \stex_import_module_uri:nn { #1 } { #2 }
3977   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3978   \seq_map_inline:Nn \l_stex_collect_imports_seq {
3979     \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }

```



```

3980 \seq_map_inline:cn {c_stex_module_##1_constants}{
3981 \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? #####1 }
3982 \bool_lazy_any_p:nT {
3983   { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_name_str}}
3984   { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_macroname_str}}
3985   { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_def_tl}}
3986 }{
3987   % TODO throw error
3988 }
3989 }
3990 }
3991
3992 \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
3993 \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_ns_str ?\l_stex_import_name_str }
3994 \prop_put:Nnx \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
3995 }
3996
3997 \NewDocumentCommand \assign { m m }{
3998 \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
3999 \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
4000 \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _def_tl}{#2}
4001 }
4002
4003 \keys_define:nn { stex / renamedec1 } {
4004   name .str_set_x:N = \l_stex_renamedec1_name_str
4005 }
4006 \cs_new_protected:Nn \__stex_copymodule_renamedec1_args:n {
4007 \str_clear:N \l_stex_renamedec1_name_str
4008 \keys_set:nn { stex / renamedec1 } { #1 }
4009 }
4010
4011 \NewDocumentCommand \renamedec1 { 0{} m m }{
4012 \__stex_copymodule_renamedec1_args:n { #1 }
4013 \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
4014 \stex_debug:nn{renamedec1}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
4015 \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
4016 \str_if_empty:NTF \l_stex_renamedec1_name_str {
4017   \tl_set:cx { #3 }{\stex_invoke_symbol:n {
4018     \l_stex_get_symbol_uri_str
4019   } }
4020 } {
4021   \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex
4022     \stex_debug:nn{renamedec1}{@~\l_stex_current_module_str ? \l_stex_renamedec1_name_str}
4023     \prop_set_eq:cc {l_stex_symdec1_
4024       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
4025       _prop
4026     }{l_stex_symdec1_ \l_stex_get_symbol_uri_str _prop}
4027     \seq_set_eq:cc {l_stex_symdec1_
4028       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
4029       _notations
4030     }{l_stex_symdec1_ \l_stex_get_symbol_uri_str _notations}
4031     \prop_put:cnx {l_stex_symdec1_
4032       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
4033       _prop

```

```

4034     }{ name }{ \l_stex_renamedekl_name_str }
4035     \prop_put:cnx {l_stex_symdecl_
4036       \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4037       _prop
4038     }{ module }{ \l_stex_current_module_str }
4039     \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
4040       \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4041     }
4042     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4043       \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4044     } }
4045   }
4046 }
4047
4048 \stex_deactivate_macro:Nn \assign {copymodules}
4049 \stex_deactivate_macro:Nn \renamedekl {copymodules}
4050 \stex_deactivate_macro:Nn \donotcopy {copymodules}
4051
4052
4053 \seq_new:N \l_stex_implicit_morphisms_seq
4054 \NewDocumentCommand \implicitmorphism { 0{} m m }{
4055   \stex_import_module_uri:nn { #1 } { #2 }
4056   \stex_debug:nn{implicits}{
4057     Implicit~morphism:~
4058     \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4059   }
4060   \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
4061     \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4062   }{
4063     \msg_error:nnn{stex}{error/conflictingmodules}{
4064       \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4065     }
4066   }
4067 }
4068 % TODO
4069
4070
4071
4072 \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
4073   \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4074 }
4075 }
4076

```

## 31.2 The feature environment

structural@feature

```

4077 <@@=stex_features>
4078
4079 \NewDocumentEnvironment{structural_feature_module}{ m m m }{
4080   \stex_if_in_module:F {
4081     \msg_set:nnn{stex}{error/nomodule}{
4082       Structural~Feature~has~to~occur~in~a~module:\\

```

```

4083     Feature~#2~of~type~#1\\
4084     In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
4085   }
4086   \msg_error:nn{stex}{error/nomodule}
4087 }
4088
4089 \stex_module_setup:nn{meta=NONE}{#2 - #1}
4090
4091 \stex_if_smsmode:F {
4092   \begin{stex_annotate_env}{ feature:#1 }{}
4093   \stex_annotate_invisible:nnn{header}{}{ #3 }
4094 }
4095 }{
4096   \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
4097   \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
4098   \stex_debug:nn{features}{
4099     Feature: \l_stex_last_feature_str
4100   }
4101   \stex_if_smsmode:F {
4102     \end{stex_annotate_env}
4103   }
4104 }

```

### 31.3 Structure

structure

```

4105 <@@=stex_structures>
4106 \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
4107   \prop_if_exist:cF {c_stex_module_ \l_stex_current_module_str _structures}{
4108     \prop_new:c {c_stex_module_ \l_stex_current_module_str _structures}
4109   }
4110   \prop_gput:cxn{c_stex_module_ \l_stex_current_module_str _structures}
4111   {#1}{#2}
4112 }
4113
4114 \keys_define:nn { stex / features / structure } {
4115   name .str_set_x:N = \l__stex_structures_name_str ,
4116 }
4117
4118 \cs_new_protected:Nn \__stex_structures_structure_args:n {
4119   \str_clear:N \l__stex_structures_name_str
4120   \keys_set:nn { stex / features / structure } { #1 }
4121 }
4122
4123 \NewDocumentEnvironment{mathstructure}{m O{}}{
4124   \__stex_structures_structure_args:n { #2 }
4125   \str_if_empty:NT \l__stex_structures_name_str {
4126     \str_set:Nx \l__stex_structures_name_str { #1 }
4127   }
4128   \exp_args:Nx \stex_symdecl_do:nn {
4129     name = \l__stex_structures_name_str ,
4130     type = \metacollection ,
4131     def = {\STEXsymbol{module-type}}{

```

```

4132     \stex_term_math_oms:nmmm {
4133       \prop_get:cn {c_stex_module_\l_stex_current_module_str _prop}
4134       { ns } ?
4135       \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4136       { name } / \l__stex_structures_name_str - structure
4137     }{}{}{}
4138   }}
4139   }{ #1 }
4140   \exp_args:Nnnx
4141   \begin{structural_feature_module}{ structure }
4142     { \l__stex_structures_name_str }{}
4143   \stex_smsmode_do:
4144   }{
4145     \end{structural_feature_module}
4146     \stex_reset_up_to_module:
4147     \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
4148     \seq_clear:N \l_tmpa_seq
4149     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4150       \seq_map_inline:cn{c_stex_module_##1_constants}{
4151         \seq_put_right:Nn \l_tmpa_seq { ##1 ? ####1 }
4152       }
4153     }
4154     \exp_args:Nnno
4155     \prop_gput:cn {c_stex_module_\l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
4156     \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
4157     \stex_add_structure_to_current_module:nn
4158       \l__stex_structures_name_str
4159       \l_stex_last_feature_str
4160     \exp_args:Nx
4161     \stex_add_to_current_module:n {
4162       \tl_set:cn { #1 }{
4163         \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures
4164       }
4165     }
4166     \exp_args:Nx
4167     \stex_do_up_to_module:n {
4168       \tl_set:cn { #1 }{
4169         \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures
4170       }
4171     }
4172   }
4173   \seq_put_right:Nx \g_stex_smsmode_allowedenvs_seq { \tl_to_str:n {mathstructure}}
4174
4175   \cs_new:Nn \stex_invoke_structure:nn {
4176     \stex_invoke_symbol:n { #1?#2 }
4177   }
4178
4179   \cs_new_protected:Nn \stex_get_structure:n {
4180     \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4181       \tl_set:Nn \l_tmpa_tl { #1 }
4182       \__stex_structures_get_from_cs:
4183     }{
4184       \cs_if_exist:cTF { #1 }{
4185         \cs_set_eq:Nc \l_tmpa_cs { #1 }

```

```

4186     \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
4187     \str_if_empty:NTF \l_tmpa_str {
4188         \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs} \stex_invoke_structure:nn {
4189             \__stex_structures_get_from_cs:
4190         }{
4191             \__stex_structures_get_from_string:n { #1 }
4192         }
4193     }{
4194         \__stex_structures_get_from_string:n { #1 }
4195     }
4196 }{
4197     \__stex_structures_get_from_string:n { #1 }
4198 }
4199 }
4200 }
4201
4202 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
4203     \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4204     { \tl_tail:N \l_tmpa_tl }
4205     \str_set:Nx \l_tmpa_str {
4206         \exp_after:wN \use_i:nn \l_tmpa_tl
4207     }
4208     \str_set:Nx \l_tmpb_str {
4209         \exp_after:wN \use_ii:nn \l_tmpa_tl
4210     }
4211     \str_set:Nx \l_stex_get_structure_str {
4212         \l_tmpa_str ? \l_tmpb_str
4213     }
4214     \str_set:Nx \l_stex_get_structure_module_str {
4215         \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
4216     }
4217 }
4218
4219 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
4220     \tl_set:Nn \l_tmpa_tl {
4221         \msg_error:nnn{stex}{error/unknownstructure}{#1}
4222     }
4223     \str_set:Nn \l_tmpa_str { #1 }
4224     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4225
4226     \seq_map_inline:Nn \l_stex_all_modules_seq {
4227         \prop_if_exist:cT {c_stex_module_##1_structures} {
4228             \prop_map_inline:cn {c_stex_module_##1_structures} {
4229                 \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
4230                     \prop_map_break:n{\seq_map_break:n{
4231                         \tl_set:Nn \l_tmpa_tl {
4232                             \str_set:Nn \l_stex_get_structure_str {##1?####1}
4233                             \str_set:Nn \l_stex_get_structure_module_str {####2}
4234                         }
4235                     }}
4236                 }
4237             }
4238         }
4239     }

```

```

4240 \l_tmpa_tl
4241 }

\instantiate

4242
4243 \keys_define:nn { stex / instantiate } {
4244   name .str_set_x:N = \l__stex_structures_name_str
4245 }
4246 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
4247   \str_clear:N \l__stex_structures_name_str
4248   \keys_set:nn { stex / instantiate } { #1 }
4249 }
4250
4251 \NewDocumentCommand \instantiate {m O{} m m m}{
4252   \begin_group
4253     \stex_get_structure:n {#4}
4254     \__stex_structures_instantiate_args:n { #2 }
4255     \str_if_empty:NT \l__stex_structures_name_str {
4256       \str_set:Nn \l__stex_structures_name_str { #1 }
4257     }
4258     \seq_clear:N \l__stex_structures_fields_seq
4259     \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4260     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4261       \seq_map_inline:cn {c_stex_module_##1_constants}{
4262         \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? #####1 }
4263       }
4264     }
4265     \seq_set_split:Nnn \l_tmpa_seq , {#3}
4266     \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4267     \prop_clear:N \l_tmpa_prop
4268     \seq_map_inline:Nn \l_tmpa_seq {
4269       \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4270       \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4271         \msg_error:nnn{stex}{error/keyval}{##1}
4272       }
4273       \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_structur
4274       \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4275       \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_uri
4276       \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}
4277       \exp_args:Nxx \str_if_eq:nnF
4278         {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4279         {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}{
4280         \msg_error:nnxxx{stex}{error/incompatible}
4281         {\l__stex_structures_dom_str
4282         {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4283         {\l_stex_get_symbol_uri_str
4284         {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}
4285       }
4286       \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
4287     }
4288     \seq_if_empty:NF \l__stex_structures_fields_seq {
4289       \msg_error:nnx{stex}{error/instantiate/missing}{\seq_use:Nn\l__stex_structures_fields_
4290     }
4291     \exp_args:Nx

```

```

4292 \stex_add_to_current_module:n {
4293   \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
4294   domain = \l_stex_get_structure_module_str ,
4295   \prop_to_keyval:N \l_tmpa_prop
4296 }
4297 \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
4298 }
4299 \exp_args:Nx
4300 \stex_do_up_to_module:n {
4301   \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
4302   domain = \l_stex_get_structure_module_str ,
4303   \prop_to_keyval:N \l_tmpa_prop
4304 }
4305 \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
4306 }
4307 \stex_debug:nn{instantiate}{
4308   Instance~\l_stex_current_module_str?\l__stex_structures_name_str \
4309   \prop_to_keyval:N \l_tmpa_prop
4310 }
4311 \exp_args:Nxx \stex_symdecl_do:nn {
4312   type={\STEXsymbol{module-type}}{
4313     \stex_term_math_oms:nnnn {
4314       \l_stex_get_structure_module_str
4315     }{}{0}{}
4316   }}
4317   {}{\l__stex_structures_name_str}
4318   \exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}
4319 \endgroup
4320 \stex_smsmode_do:\ignorespacesandpars
4321 }
4322 \tl_put_right:Nx \g_stex_smsmode_allowedmacros_escape_tl {\instantiate}
4323
4324 \cs_new_protected:Nn \stex_symbol_or_var:n {
4325   \cs_if_exist:cTF{#1}{
4326     \cs_set_eq:Nc \l_tmpa_tl { #1 }
4327     \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4328     \str_if_empty:NTF \l_tmpa_str {
4329       \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
4330       \stex_invoke_variable:n {
4331         \bool_set_true:N \l_stex_symbol_or_var_bool
4332         \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4333         \str_set:Nx \l_stex_get_symbol_uri_str {
4334           \exp_after:wN \use:n \l_tmpa_tl
4335         }
4336       }{
4337         \bool_set_false:N \l_stex_symbol_or_var_bool
4338         \stex_get_symbol:n{#1}
4339       }
4340     }{
4341       \__stex_structures_symbolorvar_from_string:n{ #1 }
4342     }
4343   }{
4344     \__stex_structures_symbolorvar_from_string:n{ #1 }
4345   }

```

```

4346 }
4347
4348 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
4349   \prop_if_exist:cTF {l_stex_variable_#1 _prop}{
4350     \bool_set_true:N \l_stex_symbol_or_var_bool
4351     \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
4352   }{
4353     \bool_set_false:N \l_stex_symbol_or_var_bool
4354     \stex_get_symbol:n{#1}
4355   }
4356 }
4357
4358
4359 \NewDocumentCommand \varinstantiate {m O{} m m m}{
4360   \beginingroup
4361     \stex_get_structure:n {#4}
4362     \__stex_structures_instantiate_args:n { #2 }
4363     \str_if_empty:NT \l__stex_structures_name_str {
4364       \str_set:Nn \l__stex_structures_name_str { #1 }
4365     }
4366     \seq_clear:N \l__stex_structures_fields_seq
4367     \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4368     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4369       \seq_map_inline:cn {c_stex_module_##1_constants}{
4370         \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? #####1 }
4371       }
4372     }
4373     \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4374     \prop_clear:N \l_tmpa_prop
4375     \tl_if_empty:nF {#3} {
4376       \seq_set_split:Nnn \l_tmpa_seq , {#3}
4377       \seq_map_inline:Nn \l_tmpa_seq {
4378         \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4379         \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4380           \msg_error:nnn{stex}{error/keyval}{##1}
4381         }
4382         \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
4383         \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4384         \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
4385         \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
4386         \bool_if:NTF \l_stex_symbol_or_var_bool {
4387           \exp_args:Nxx \str_if_eq:nnF
4388             {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4389             {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}{
4390             \msg_error:nnxxx{stex}{error/incompatible}
4391             {\l__stex_structures_dom_str}
4392             {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4393             {\l_stex_get_symbol_uri_str}
4394             {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}
4395           }
4396           \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n {
4397         }{
4398           \exp_args:Nxx \str_if_eq:nnF
4399             {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}

```



```

4400         {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
4401         \msg_error:nnxxxx{stex}{error/incompatible}
4402         {\l__stex_structures_dom_str}
4403         {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4404         {\l_stex_get_symbol_uri_str}
4405         {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
4406     }
4407     \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n {\l
4408 }
4409 }
4410 }
4411 \tl_gclear:N \g__stex_structures_aftergroup_tl
4412 \seq_map_inline:Nn \l__stex_structures_fields_seq {
4413     \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdecl_
4414     \stex_find_notation:nn{##1}{}
4415     \cs_gset_eq:cc{g__stex_structures_tmpa_\l_tmpa_str _cs}
4416     {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4417     \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
4418     \cs_gset_eq:cc {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
4419     {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4420 }
4421 }
4422 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4423     \prop_set_from_keyval:cn { l_stex_variable_ \l_tmpa_str _prop}{
4424         name = \l_tmpa_str ,
4425         args = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4426         arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4427         assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4428     }
4429     \cs_set_eq:cc {stex_var_notation_\l_tmpa_str _cs}
4430     {g__stex_structures_tmpa_\l_tmpa_str _cs}
4431     \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str _cs}
4432     {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
4433 }
4434 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_invoke
4435 }
4436 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4437     \prop_set_from_keyval:cn {l_stex_varinstance_\l__stex_structures_name_str _prop }{
4438         domain = \l_stex_get_structure_module_str ,
4439         \prop_to_keyval:N \l_tmpa_prop
4440     }
4441     \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
4442     \tl_set:cn {l_stex_varinstance_\l__stex_structures_name_str _op_tl}{
4443         \exp_args:Nnx \exp_not:N \use:nn {
4444             \str_set:Nn \exp_not:N \l_stex_current_symbol_str {var://\l__stex_structures_name_
4445             \_stex_term_omv:nn {var://\l__stex_structures_name_str}{
4446                 \exp_not:n{
4447                     \_varcomp{#5}
4448                 }
4449             }
4450         }{
4451             \exp_not:n{\_stex_reset:N \l_stex_current_symbol_str}
4452         }
4453     }

```

```

4454     }
4455     \aftergroup\g__stex_structures_aftergroup_tl
4456 \endgroup
4457 \stex_smsmode_do:\ignorespacesandpars
4458 }
4459
4460 \cs_new_protected:Nn \stex_invoke_instance:n {
4461   \peek_charcode_remove:NTF ! {
4462     \stex_invoke_symbol:n{#1}
4463   }{
4464     \stex_invoke_instance:nn {#1}
4465   }
4466 }
4467
4468
4469 \cs_new_protected:Nn \stex_invoke_varinstance:n {
4470   \peek_charcode_remove:NTF ! {
4471     \use:c{l_stex_varinstance_#1_op_tl}
4472   }{
4473     \stex_invoke_varinstance:nn {#1}
4474   }
4475 }
4476
4477 \cs_new_protected:Nn \stex_invoke_instance:nn {
4478   \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
4479     \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
4480   }{
4481     \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
4482     \msg_error:nnnn{stex}{error/unknownfield}{#2}{#1}{
4483       \prop_to_keyval:N \l_tmpa_prop
4484     }
4485   }
4486 }
4487
4488 \cs_new_protected:Nn \stex_invoke_varinstance:nn {
4489   \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
4490     \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
4491     \l_tmpa_tl
4492   }{
4493     \msg_error:nnnn{stex}{error/unknownfield}{#2}{#1}{
4494   }
4495 }

```

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

`\stex_invoke_structure:nnn`

```

4496 % #1: URI of the instance
4497 % #2: URI of the instantiated module
4498 \cs_new_protected:Nn \stex_invoke_structure:nnn {
4499   \tl_if_empty:nTF{ #3 }{
4500     \prop_set_eq:Nc \l__stex_structures_structure_prop {
4501       c_stex_feature_ #2 _prop
4502     }
4503     \tl_clear:N \l_tmpa_tl

```

```

4504 \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
4505 \seq_map_inline:Nn \l_tmpa_seq {
4506   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
4507   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
4508   \cs_if_exist:cT {
4509     stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
4510   }{
4511     \tl_if_empty:NF \l_tmpa_tl {
4512       \tl_put_right:Nn \l_tmpa_tl {,}
4513     }
4514     \tl_put_right:Nx \l_tmpa_tl {
4515       \stex_invoke_symbol:n {#1/\l_tmpa_str}!
4516     }
4517   }
4518 }
4519 \exp_args:No \mathstruct \l_tmpa_tl
4520 }{
4521   \stex_invoke_symbol:n{#1/#3}
4522 }
4523 }

```

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

```

4524 </package>

```

## Chapter 32

# STEX -Statements Implementation

```
4525 <*package>
4526
4527 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4528
4529 <@@=stex_statements>
    Warnings and error messages
4530
\titleemph
4531 \def\titleemph#1{\textbf{#1}}
    (End definition for \titleemph. This function is documented on page ??.)
```

### 32.1 Definitions

#### definiendum

```
4532 \keys_define:nn {stex / definiendum }{
4533   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
4534   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
4535   root     .str_set_x:N    = \l__stex_statements_definiendum_root_str,
4536   gfa      .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
4537 }
4538 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
4539   \str_clear:N \l__stex_statements_definiendum_root_str
4540   \tl_clear:N \l__stex_statements_definiendum_post_tl
4541   \str_clear:N \l__stex_statements_definiendum_gfa_str
4542   \keys_set:nn { stex / definiendum }{ #1 }
4543 }
4544 \NewDocumentCommand \definiendum { O{} m m } {
4545   \__stex_statements_definiendum_args:n { #1 }
4546   \stex_get_symbol:n { #2 }
4547   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4548   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
4549     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```

4550     \tl_set:Nn \l_tmpa_tl { #3 }
4551   } {
4552     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
4553     \tl_set:Nn \l_tmpa_tl {
4554       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
4555     }
4556   }
4557 } {
4558   \tl_set:Nn \l_tmpa_tl { #3 }
4559 }
4560
4561 % TODO root
4562 \rustex_if:TF {
4563   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
4564 } {
4565   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
4566 }
4567 }
4568 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

#### definame

```

4569
4570 \NewDocumentCommand \definame { 0{ } m } {
4571   \__stex_statements_definiendum_args:n { #1 }
4572   % TODO: root
4573   \stex_get_symbol:n { #2 }
4574   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4575   \str_set:Nx \l_tmpa_str {
4576     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4577   }
4578   \str_replace_all:Nnn \l_tmpa_str {-} {~}
4579   \rustex_if:TF {
4580     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4581       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4582     }
4583   } {
4584     \exp_args:Nnx \defemph@uri {
4585       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4586     } { \l_stex_get_symbol_uri_str }
4587   }
4588 }
4589 \stex_deactivate_macro:Nn \definame {definition~environments}
4590
4591 \NewDocumentCommand \Definame { 0{ } m } {
4592   \__stex_statements_definiendum_args:n { #1 }
4593   \stex_get_symbol:n { #2 }
4594   \str_set:Nx \l_tmpa_str {
4595     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4596   }
4597   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4598   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4599   \rustex_if:TF {

```

```

4600 \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4601 \l_tmpa_str\l__stex_statements_definiendum_post_tl
4602 }
4603 } {
4604 \exp_args:Nnx \defemph@uri {
4605 \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
4606 } { \l_stex_get_symbol_uri_str }
4607 }
4608 }
4609 \stex_deactivate_macro:Nn \Definame {definition~environments}
4610
4611 \NewDocumentCommand \premise { m }{
4612 \stex_annotate:nnn{ premise }{}{ #1 }
4613 }
4614 \NewDocumentCommand \conclusion { m }{
4615 \stex_annotate:nnn{ conclusion }{}{ #1 }
4616 }
4617 \NewDocumentCommand \definiens { O{} m }{
4618 \str_clear:N \l_stex_get_symbol_uri_str
4619 \tl_if_empty:nF {#1} {
4620 \stex_get_symbol:n { #1 }
4621 }
4622 \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
4623 }
4624
4625 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
4626 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
4627 \stex_deactivate_macro:Nn \definiens {definition~environments}
4628

```

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

## sdefinition

```

4629
4630 \keys_define:nn {stex / sdefinition }{
4631 type .str_set_x:N = \sdefinitiontype,
4632 id .str_set_x:N = \sdefinitionid,
4633 name .str_set_x:N = \sdefinitionname,
4634 for .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4635 title .tl_set:N = \sdefinitiontitle
4636 }
4637 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
4638 \str_clear:N \sdefinitiontype
4639 \str_clear:N \sdefinitionid
4640 \str_clear:N \sdefinitionname
4641 \clist_clear:N \l__stex_statements_sdefinition_for_clist
4642 \tl_clear:N \sdefinitiontitle
4643 \keys_set:nn { stex / sdefinition }{ #1 }
4644 }
4645
4646 \NewDocumentEnvironment{sdefinition}{O{}}{
4647 \__stex_statements_sdefinition_args:n{ #1 }
4648 \stex_reactivate_macro:N \definiendum
4649 \stex_reactivate_macro:N \definame

```

```

4650 \stex_reactivate_macro:N \Definame
4651 \stex_reactivate_macro:N \premise
4652 \stex_reactivate_macro:N \definiens
4653 \stex_if_smsmode:F{
4654   \seq_clear:N \l_tmpa_seq
4655   \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4656     \tl_if_empty:nF{ ##1 }{
4657       \stex_get_symbol:n { ##1 }
4658       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4659         \l_stex_get_symbol_uri_str
4660       }
4661     }
4662   }
4663   \exp_args:Nnnx
4664   \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {,}}
4665   \str_if_empty:NF \sdefinitiontype {
4666     \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{ }
4667   }
4668   \clist_set:N \l_tmpa_clist \sdefinitiontype
4669   \tl_clear:N \l_tmpa_tl
4670   \clist_map_inline:Nn \l_tmpa_clist {
4671     \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
4672       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
4673     }
4674   }
4675   \tl_if_empty:NTF \l_tmpa_tl {
4676     \__stex_statements_sdefinition_start:
4677   }{
4678     \l_tmpa_tl
4679   }
4680 }
4681 \stex_ref_new_doc_target:n \sdefinitionid
4682 \stex_smsmode_do:
4683 }{
4684   \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
4685   \stex_if_smsmode:F {
4686     \clist_set:N \l_tmpa_clist \sdefinitiontype
4687     \tl_clear:N \l_tmpa_tl
4688     \clist_map_inline:Nn \l_tmpa_clist {
4689       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
4690         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
4691       }
4692     }
4693     \tl_if_empty:NTF \l_tmpa_tl {
4694       \__stex_statements_sdefinition_end:
4695     }{
4696       \l_tmpa_tl
4697     }
4698     \end{stex_annotate_env}
4699   }
4700 }

```

**\stexpatchdefinition**

```

4701 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {

```

```

4702 \par\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
4703   ~(\sdefinitiontitle)
4704 }~}
4705 }
4706 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\par\medskip}
4707
4708 \newcommand\stexpatchdefinition[3][] {
4709   \str_set:Nx \l_tmpa_str{ #1 }
4710   \str_if_empty:NTF \l_tmpa_str {
4711     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
4712     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
4713   }{
4714     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
4715     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
4716   }
4717 }

```

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

\inlinedef inline:

```

4718 \keys_define:nn {stex / inlinedef }{
4719   type      .str_set_x:N = \sdefinitiontype,
4720   id        .str_set_x:N = \sdefinitionid,
4721   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4722   name      .str_set_x:N = \sdefinitionname
4723 }
4724 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
4725   \str_clear:N \sdefinitiontype
4726   \str_clear:N \sdefinitionid
4727   \str_clear:N \sdefinitionname
4728   \clist_clear:N \l__stex_statements_sdefinition_for_clist
4729   \keys_set:nn { stex / inlinedef }{ #1 }
4730 }
4731 \NewDocumentCommand \inlinedef { 0{} m } {
4732   \begingroup
4733   \__stex_statements_inlinedef_args:n{ #1 }
4734   \stex_reactivate_macro:N \definiendum
4735   \stex_reactivate_macro:N \definame
4736   \stex_reactivate_macro:N \Definame
4737   \stex_reactivate_macro:N \premise
4738   \stex_reactivate_macro:N \definiens
4739   \stex_ref_new_doc_target:n \sdefinitionid
4740   \stex_if_smsmode:TF{
4741     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
4742   }{
4743     \seq_clear:N \l_tmpa_seq
4744     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4745       \tl_if_empty:nF{ ##1 }{
4746         \stex_get_symbol:n { ##1 }
4747         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4748           \l_stex_get_symbol_uri_str
4749         }
4750       }
4751     }

```



```

4752 \exp_args:Nnx
4753 \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
4754   \str_if_empty:NF \sdefinitiontype {
4755     \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{}
4756   }
4757   #2
4758   \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}}{\sdefinitionname} }
4759 }
4760 }
4761 \endgroup
4762 \stex_smsmode_do:
4763 }

```

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

## 32.2 Assertions

**sassertion**

```

4764
4765 \keys_define:nn {stex / sassertion }{
4766   type      .str_set_x:N = \sassertiontype,
4767   id        .str_set_x:N = \sassertionid,
4768   title     .tl_set:N    = \sassertiontitle ,
4769   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4770   name      .str_set_x:N = \sassertionname
4771 }
4772 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
4773   \str_clear:N \sassertiontype
4774   \str_clear:N \sassertionid
4775   \str_clear:N \sassertionname
4776   \clist_clear:N \l__stex_statements_sassertion_for_clist
4777   \tl_clear:N \sassertiontitle
4778   \keys_set:nn { stex / sassertion }{ #1 }
4779 }
4780
4781 %\tl_new:N \g__stex_statements_aftergroup_tl
4782
4783 \NewDocumentEnvironment{sassertion}{0{}}{
4784   \__stex_statements_sassertion_args:n{ #1 }
4785   \stex_reactivate_macro:N \premise
4786   \stex_reactivate_macro:N \conclusion
4787   \stex_if_smsmode:F {
4788     \seq_clear:N \l_tmpa_seq
4789     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4790       \tl_if_empty:nF{ ##1 }{
4791         \stex_get_symbol:n { ##1 }
4792         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4793           \l_stex_get_symbol_uri_str
4794         }
4795       }
4796     }
4797   }
4798   \exp_args:Nnnx
4799   \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}

```

```

4799 \str_if_empty:NF \sassertiontype {
4800   \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
4801 }
4802 \clist_set:No \l_tmpa_clist \sassertiontype
4803 \tl_clear:N \l_tmpa_tl
4804 \clist_map_inline:Nn \l_tmpa_clist {
4805   \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
4806     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
4807   }
4808 }
4809 \tl_if_empty:NTF \l_tmpa_tl {
4810   \__stex_statements_sassertion_start:
4811 }{
4812   \l_tmpa_tl
4813 }
4814 }
4815 \str_if_empty:NTF \sassertionid {
4816   \str_if_empty:NF \sassertionname {
4817     \stex_ref_new_doc_target:n {}
4818   }
4819 } {
4820   \stex_ref_new_doc_target:n \sassertionid
4821 }
4822 \stex_smsmode_do:
4823 }{
4824   \str_if_empty:NF \sassertionname {
4825     \stex_symdecl_do:nn{ }\sassertionname}
4826     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4827   }
4828   \stex_if_smsmode:F {
4829     \clist_set:No \l_tmpa_clist \sassertiontype
4830     \tl_clear:N \l_tmpa_tl
4831     \clist_map_inline:Nn \l_tmpa_clist {
4832       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
4833         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
4834       }
4835     }
4836     \tl_if_empty:NTF \l_tmpa_tl {
4837       \__stex_statements_sassertion_end:
4838     }{
4839       \l_tmpa_tl
4840     }
4841     \end{stex_annotate_env}
4842   }
4843 }

```

### **\stexpatchassertion**

```

4844
4845 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
4846   \par\noindent\titllemph{Assertion~\tl_if_empty:NF \sassertiontitle {
4847     (\sassertiontitle)
4848   }~}
4849 }
4850 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}

```

```

4851
4852 \newcommand\stexpatchassertion[3] [] {
4853   \str_set:Nx \l_tmpa_str{ #1 }
4854   \str_if_empty:NTF \l_tmpa_str {
4855     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
4856     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
4857   }{
4858     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
4859     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
4860   }
4861 }

```

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

\inlineass inline:

```

4862 \keys_define:nn {stex / inlineass }{
4863   type      .str_set_x:N = \sassertiontype,
4864   id        .str_set_x:N = \sassertionid,
4865   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4866   name      .str_set_x:N = \sassertionname
4867 }
4868 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
4869   \str_clear:N \sassertiontype
4870   \str_clear:N \sassertionid
4871   \str_clear:N \sassertionname
4872   \clist_clear:N \l__stex_statements_sassertion_for_clist
4873   \keys_set:nn { stex / inlineass }{ #1 }
4874 }
4875 \NewDocumentCommand \inlineass { 0{} m } {
4876   \beginngroup
4877   \stex_reactivate_macro:N \premise
4878   \stex_reactivate_macro:N \conclusion
4879   \__stex_statements_inlineass_args:n{ #1 }
4880   \str_if_empty:NTF \sassertionid {
4881     \str_if_empty:NF \sassertionname {
4882       \stex_ref_new_doc_target:n {}
4883     }
4884   } {
4885     \stex_ref_new_doc_target:n \sassertionid
4886   }
4887
4888   \stex_if_smsmode:TF{
4889     \str_if_empty:NF \sassertionname {
4890       \stex_symdecl_do:nn{}{\sassertionname}
4891       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4892     }
4893   }{
4894     \seq_clear:N \l_tmpa_seq
4895     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4896       \tl_if_empty:nF{ ##1 }{
4897         \stex_get_symbol:n { ##1 }
4898         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4899           \l_stex_get_symbol_uri_str
4900         }

```

```

4901     }
4902   }
4903   \exp_args:Nnx
4904   \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{
4905     \str_if_empty:NF \sassertiontype {
4906       \stex_annotate_invisible:nnn{type}{\sassertiontype}{}}
4907   }
4908   #2
4909   \str_if_empty:NF \sassertionname {
4910     \stex_symdecl_do:nn{}{\sassertionname}
4911     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4912   }
4913 }
4914 }
4915 \endgroup
4916 \stex_smsmode_do:
4917 }

```

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

## 32.3 Examples

`sexample`

```

4918
4919 \keys_define:nn {stex / sexample }{
4920   type      .str_set_x:N = \exampletype,
4921   id        .str_set_x:N = \sexampleid,
4922   title     .tl_set:N     = \sexampletile,
4923   name      .str_set_x:N = \sexamplename ,
4924   for       .clist_set:N  = \l__stex_statements_sexample_for_clist,
4925 }
4926 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
4927   \str_clear:N \sexampletype
4928   \str_clear:N \sexampleid
4929   \str_clear:N \sexamplename
4930   \tl_clear:N \sexampletile
4931   \clist_clear:N \l__stex_statements_sexample_for_clist
4932   \keys_set:nn { stex / sexample }{ #1 }
4933 }
4934
4935 \NewDocumentEnvironment{sexample}{0{}}{
4936   \__stex_statements_sexample_args:n{ #1 }
4937   \stex_reactivate_macro:N \premise
4938   \stex_reactivate_macro:N \conclusion
4939   \stex_if_smsmode:F {
4940     \seq_clear:N \l_tmpa_seq
4941     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4942       \tl_if_empty:nF{ ##1 }{
4943         \stex_get_symbol:n { ##1 }
4944         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4945           \l_stex_get_symbol_uri_str
4946         }
4947       }

```

```

4948 }
4949 \exp_args:Nnnx
4950 \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
4951 \str_if_empty:NF \sexamplotype {
4952   \stex_annotate_invisible:nnn{type}{\sexamplotype}{}
4953 }
4954 \clist_set:No \l_tmpa_clist \sexamplotype
4955 \tl_clear:N \l_tmpa_tl
4956 \clist_map_inline:Nn \l_tmpa_clist {
4957   \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
4958     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
4959   }
4960 }
4961 \tl_if_empty:NTF \l_tmpa_tl {
4962   __stex_statements_sexample_start:
4963 }{
4964   \l_tmpa_tl
4965 }
4966 }
4967 \str_if_empty:NF \sexampleid {
4968   \stex_ref_new_doc_target:n \sexampleid
4969 }
4970 \stex_smsmode_do:
4971 }{
4972   \str_if_empty:NF \sexamplename { \stex_symdecl_do:nn{}{\sexamplename} }
4973   \stex_if_smsmode:F {
4974     \clist_set:No \l_tmpa_clist \sexamplotype
4975     \tl_clear:N \l_tmpa_tl
4976     \clist_map_inline:Nn \l_tmpa_clist {
4977       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
4978         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
4979       }
4980     }
4981     \tl_if_empty:NTF \l_tmpa_tl {
4982       __stex_statements_sexample_end:
4983     }{
4984       \l_tmpa_tl
4985     }
4986     \end{stex_annotate_env}
4987   }
4988 }

```

**\stexpatchexample**

```

4989
4990 \cs_new_protected:Nn \__stex_statements_sexample_start: {
4991   \par\noindent\titllemph{Example~\tl_if_empty:NF \sexampltitle {
4992     (\sexampltitle)
4993   }~}
4994 }
4995 \cs_new_protected:Nn \__stex_statements_sexample_end: {\par\medskip}
4996
4997 \newcommand\stexpatchexample[3] [] {
4998   \str_set:Nx \l_tmpa_str{ #1 }
4999   \str_if_empty:NTF \l_tmpa_str {

```

```

5000     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
5001     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
5002   }{
5003     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
5004     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
5005   }
5006 }

```

\inlineex inline:

```

5050 \stex_smsmode_do:
5051 }

```

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

## 32.4 Logical Paragraphs

sparagraph

```

5052 \keys_define:nn { stex / sparagraph} {
5053   id      .str_set_x:N = \sparagraphid ,
5054   title   .tl_set:N    = \l_stex_sparagraph_title_tl ,
5055   type    .str_set_x:N = \sparagraphtype ,
5056   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5057   from    .tl_set:N    = \sparagraphfrom ,
5058   to      .tl_set:N    = \sparagraphto ,
5059   start   .tl_set:N    = \l_stex_sparagraph_start_tl ,
5060   name    .str_set:N   = \sparagraphname
5061 }
5062
5063 \cs_new_protected:Nn \stex_sparagraph_args:n {
5064   \tl_clear:N \l_stex_sparagraph_title_tl
5065   \tl_clear:N \sparagraphfrom
5066   \tl_clear:N \sparagraphto
5067   \tl_clear:N \l_stex_sparagraph_start_tl
5068   \str_clear:N \sparagraphid
5069   \str_clear:N \sparagraphtype
5070   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5071   \str_clear:N \sparagraphname
5072   \keys_set:nn { stex / sparagraph }{ #1 }
5073 }
5074 \newif\if@in@omtext\@in@omtextfalse
5075
5076 \NewDocumentEnvironment {sparagraph} { 0{ } } {
5077   \stex_sparagraph_args:n { #1 }
5078   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5079     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
5080   }{
5081     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
5082   }
5083   \@in@omtexttrue
5084   \stex_if_smsmode:F {
5085     \seq_clear:N \l_tmpa_seq
5086     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5087       \tl_if_empty:nF{ ##1 }{
5088         \stex_get_symbol:n { ##1 }
5089         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5090           \l_stex_get_symbol_uri_str
5091         }
5092       }
5093     }
5094     \exp_args:Nnnx
5095     \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
5096     \str_if_empty:NF \sparagraphtype {

```

```

5097     \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}
5098   }
5099   \str_if_empty:NF \sparagraphfrom {
5100     \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5101   }
5102   \str_if_empty:NF \sparagraphto {
5103     \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5104   }
5105   \clist_set:No \l_tmpa_clist \sparagraphtype
5106   \tl_clear:N \l_tmpa_tl
5107   \clist_map_inline:Nn \sparagraphtype {
5108     \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
5109       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
5110     }
5111   }
5112   \tl_if_empty:NTF \l_tmpa_tl {
5113     \__stex_statements_sparagraph_start:
5114   }{
5115     \l_tmpa_tl
5116   }
5117 }
5118 \clist_set:No \l_tmpa_clist \sparagraphtype
5119 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}
5120 {
5121   \stex_reactivate_macro:N \definiendum
5122   \stex_reactivate_macro:N \definame
5123   \stex_reactivate_macro:N \Definame
5124   \stex_reactivate_macro:N \premise
5125   \stex_reactivate_macro:N \definiens
5126 }
5127 \str_if_empty:NTF \sparagraphid {
5128   \str_if_empty:NTF \sparagraphname {
5129     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5130       \stex_ref_new_doc_target:n {}
5131     }
5132   } {
5133     \stex_ref_new_doc_target:n {}
5134   }
5135 } {
5136   \stex_ref_new_doc_target:n \sparagraphid
5137 }
5138 \exp_args:NNx
5139 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5140   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5141     \tl_if_empty:nF{ ##1 }{
5142       \stex_get_symbol:n { ##1 }
5143       \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5144     }
5145   }
5146 }
5147 \stex_smsmode_do:
5148 \ignorespacesandpars
5149 }{
5150   \str_if_empty:NF \sparagraphname {

```



```

5151 \stex_symdecl_do:nn{}{\sparagraphname}
5152 \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5153 }
5154 \stex_if_smsmode:F {
5155 \clist_set:N \l_tmpa_clist \sparagraphtype
5156 \tl_clear:N \l_tmpa_tl
5157 \clist_map_inline:Nn \l_tmpa_clist {
5158 \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
5159 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
5160 }
5161 }
5162 \tl_if_empty:NTF \l_tmpa_tl {
5163 \__stex_statements_sparagraph_end:
5164 }{
5165 \l_tmpa_tl
5166 }
5167 \end{stex_annotate_env}
5168 }
5169 }

```

## \stexpatchparagraph

```

5170
5171 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
5172 \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5173 \tl_if_empty:NF \l_stex_sparagraph_title_tl {
5174 \titleemph{\l_stex_sparagraph_title_tl}:~
5175 }
5176 }{
5177 \titleemph{\l_stex_sparagraph_start_tl}~
5178 }
5179 }
5180 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
5181
5182 \newcommand\stexpatchparagraph[3] [] {
5183 \str_set:Nx \l_tmpa_str{ #1 }
5184 \str_if_empty:NTF \l_tmpa_str {
5185 \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
5186 \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
5187 }{
5188 \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2 }
5189 \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
5190 }
5191 }
5192
5193 \keys_define:nn { stex / inlinepara } {
5194 id .str_set_x:N = \sparagraphid ,
5195 type .str_set_x:N = \sparagraphtype ,
5196 for .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5197 from .tl_set:N = \sparagraphfrom ,
5198 to .tl_set:N = \sparagraphto ,
5199 name .str_set:N = \sparagraphname
5200 }
5201 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
5202 \tl_clear:N \sparagraphfrom

```

```

5203 \tl_clear:N \sparagraphto
5204 \str_clear:N \sparagraphid
5205 \str_clear:N \sparapgraphtype
5206 \clist_clear:N \l__stex_statements_sparagraph_for_clist
5207 \str_clear:N \sparagraphname
5208 \keys_set:nn { stex / inlinepara }{ #1 }
5209 }
5210 \NewDocumentCommand \inlinepara { 0{} m } {
5211   \begingroup
5212   \__stex_statements_inlinepara_args:n{ #1 }
5213   \clist_set:No \l_tmpa_clist \sparapgraphtype
5214   \str_if_empty:NTF \sparagraphid {
5215     \str_if_empty:NTF \sparagraphname {
5216       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5217         \stex_ref_new_doc_target:n {}
5218       }
5219     } {
5220       \stex_ref_new_doc_target:n {}
5221     }
5222   } {
5223     \stex_ref_new_doc_target:n \sparagraphid
5224   }
5225   \stex_if_smsmode:TF{
5226     \str_if_empty:NF \sparagraphname {
5227       \stex_symdecl_do:nn{}{\sparagraphname}
5228       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5229     }
5230   }{
5231     \seq_clear:N \l_tmpa_seq
5232     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5233       \tl_if_empty:nF{ ##1 }{
5234         \stex_get_symbol:n { ##1 }
5235         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5236           \l_stex_get_symbol_uri_str
5237         }
5238       }
5239     }
5240     \exp_args:NNx
5241     \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
5242       \str_if_empty:NF \sparapgraphtype {
5243         \stex_annotate_invisible:nnn{type}{\sparapgraphtype}{}
5244       }
5245       \str_if_empty:NF \sparagraphfrom {
5246         \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5247       }
5248       \str_if_empty:NF \sparagraphto {
5249         \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5250       }
5251       \str_if_empty:NF \sparagraphname {
5252         \stex_symdecl_do:nn{}{\sparagraphname}
5253         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5254       }
5255       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5256         \clist_map_inline:Nn \l_tmpa_seq {

```

```

5257         \stex_ref_new_sym_target:n {##1}
5258     }
5259 }
5260 #2
5261 }
5262 }
5263 \endgroup
5264 \stex_smsmode_do:
5265 }
5266

```

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

```

5267 </package>

```

## 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>

```
5268 <*package>
5269 <@@=stex_sproof>
5270
5271 %%%%%%%%%%% sproof.dtx %%%%%%%%%%%
5272
```

### 33.2 Proofs

We first define some keys for the proof environment.

```
5273 \keys_define:nn { stex / spf } {
5274   id          .str_set_x:N = \spfid,
5275   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,
5276   from        .tl_set:N    = \l__stex_sproof_spf_from_tl ,
5277   proofend    .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
5278   type        .str_set_x:N = \spftype,
5279   title       .tl_set:N    = \spftitle,
5280   continues   .tl_set:N    = \l__stex_sproof_spf_continues_tl,
5281   functions   .tl_set:N    = \l__stex_sproof_spf_functions_tl,
5282   method      .tl_set:N    = \l__stex_sproof_spf_method_tl
5283 }
5284 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
5285   \str_clear:N \spfid
5286   \tl_clear:N \l__stex_sproof_spf_for_tl
5287   \tl_clear:N \l__stex_sproof_spf_from_tl
5288   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
5289   \str_clear:N \spftype
5290   \tl_clear:N \spftitle
5291   \tl_clear:N \l__stex_sproof_spf_continues_tl
5292   \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

```

5293 \tl_clear:N \l__stex_sproof_spf_method_tl
5294 \bool_set_false:N \l__stex_sproof_inc_counter_bool
5295 \keys_set:nn { stex / spf }{ #1 }
5296 }

```

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

```

5297 \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.

```

5298 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
5299 \cs_new_protected:Npn \sproofnumber {
5300   \int_set:Nn \l_tmpa_int {1}
5301   \bool_while_do:nn {
5302     \int_compare_p:nNn {
5303       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5304     } > 0
5305   }{
5306     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
5307     \int_incr:N \l_tmpa_int
5308   }
5309 }
5310 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
5311   \int_set:Nn \l_tmpa_int {1}
5312   \bool_while_do:nn {
5313     \int_compare_p:nNn {
5314       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5315     } > 0
5316   }{
5317     \int_incr:N \l_tmpa_int
5318   }
5319   \int_compare:nNnF \l_tmpa_int = 1 {
5320     \int_decr:N \l_tmpa_int
5321   }
5322   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
5323     \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

```

5324 }
5325 }
5326
5327 \cs_new_protected:Npn \__stex_sproof_add_counter: {
5328   \int_set:Nn \l_tmpa_int {1}
5329   \bool_while_do:nn {
5330     \int_compare_p:nNn {
5331       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5332     } > 0
5333   }{
5334     \int_incr:N \l_tmpa_int
5335   }
5336   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 1 }
5337 }
5338
5339 \cs_new_protected:Npn \__stex_sproof_remove_counter: {
5340   \int_set:Nn \l_tmpa_int {1}
5341   \bool_while_do:nn {
5342     \int_compare_p:nNn {
5343       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5344     } > 0
5345   }{
5346     \int_incr:N \l_tmpa_int
5347   }
5348   \int_decr:N \l_tmpa_int
5349   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 0 }
5350 }

```

**\sproofend** This macro places a little box at the end of the line if there is space, or at the end of the next line if there isn't

```

5351 \def\sproof@box{
5352   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
5353 }
5354 \def\sproofend{
5355   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
5356     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
5357   }
5358 }

```

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

**spf@\*@kw**

```

5359 \def\spf@proofsketch@kw{Proof-Sketch}
5360 \def\spf@proof@kw{Proof}
5361 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

5362 \AddToHook{begindocument}{
5363   \ltx@ifpackageloaded{babel}{
5364     \makeatletter
5365     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5366     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5367       \input{sproof-ngerman.ldf}

```

```

5368     }
5369     \clist_if_in:NnT \l_tmpa_clist {finnish}{
5370       \input{sproof-finnish.ldf}
5371     }
5372     \clist_if_in:NnT \l_tmpa_clist {french}{
5373       \input{sproof-french.ldf}
5374     }
5375     \clist_if_in:NnT \l_tmpa_clist {russian}{
5376       \input{sproof-russian.ldf}
5377     }
5378     \makeatother
5379   }{}
5380 }

```

spfsketch

```

5381 \newcommand\spfsketch[2] [] {
5382   \beginingroup
5383   \let \premise \stex_proof_premise:
5384   \__stex_sproof_spf_args:n{#1}
5385   \stex_if_smsmode:TF {
5386     \str_if_empty:NF \spfid {
5387       \stex_ref_new_doc_target:n \spfid
5388     }
5389   }{
5390     \seq_clear:N \l_tmpa_seq
5391     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5392       \tl_if_empty:nF{ ##1 }{
5393         \stex_get_symbol:n { ##1 }
5394         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5395           \l_stex_get_symbol_uri_str
5396         }
5397       }
5398     }
5399     \exp_args:Nnx
5400     \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
5401       \str_if_empty:NF \spftype {
5402         \stex_annotate_invisible:nnn{type}{\spftype}{-}
5403       }
5404       \clist_set:No \l_tmpa_clist \spftype
5405       \tl_set:Nn \l_tmpa_tl {
5406         \titleemph{
5407           \tl_if_empty:NTF \spftitle {
5408             \spf@proofsketch@kw
5409           }{
5410             \spftitle
5411           }
5412         }::~
5413       }
5414       \clist_map_inline:Nn \l_tmpa_clist {
5415         \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5416           \tl_clear:N \l_tmpa_tl
5417         }
5418       }
5419       \str_if_empty:NF \spfid {

```

```

5420         \stex_ref_new_doc_target:n \spfid
5421     }
5422     \l_tmpa_tl #2 \sproofend
5423 }
5424 }
5425 \endgroup
5426 \stex_smsmode_do:
5427 }
5428

```

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

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

```

5429 \newenvironment{spfeq}[2][]{
5430   \__stex_sproof_spf_args:n{#1}
5431   \let \premise \stex_proof_premise:
5432   \stex_if_smsmode:TF {
5433     \str_if_empty:NF \spfid {
5434       \stex_ref_new_doc_target:n \spfid
5435     }
5436   }{
5437     \seq_clear:N \l_tmpa_seq
5438     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5439       \tl_if_empty:NF{ ##1 }{
5440         \stex_get_symbol:n { ##1 }
5441         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5442           \l_stex_get_symbol_uri_str
5443         }
5444       }
5445     }
5446     \exp_args:Nnnx
5447     \begin{stex_annotate_env}{spfeq}{\seq_use:Nn \l_tmpa_seq {,}}
5448     \str_if_empty:NF \spftype {
5449       \stex_annotate_invisible:nnn{type}{\spftype}{ }
5450     }
5451
5452     \clist_set:No \l_tmpa_clist \spftype
5453     \tl_clear:N \l_tmpa_tl
5454     \clist_map_inline:Nn \l_tmpa_clist {
5455       \tl_if_exist:cT {__stex_sproof_spfeq_##1_start:}{
5456         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_start:}}
5457       }
5458       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5459         \tl_set:Nn \l_tmpa_tl {\use:n{ }}
5460       }
5461     }
5462     \tl_if_empty:NTF \l_tmpa_tl {
5463       \__stex_sproof_spfeq_start:
5464     }{
5465       \l_tmpa_tl
5466     }{-#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



```

5467 \str_if_empty:NF \spfid {
5468 \stex_ref_new_doc_target:n \spfid
5469 }
5470 \begin{displaymath}\begin{array}{rcll}
5471 }
5472 \stex_smsmode_do:
5473 }{
5474 \stex_if_smsmode:F {
5475 \end{array}\end{displaymath}
5476 \clist_set:No \l_tmpa_clist \spftype
5477 \tl_clear:N \l_tmpa_tl
5478 \clist_map_inline:Nn \l_tmpa_clist {
5479 \tl_if_exist:cT {__stex_sproof_spfeq_##1_end:}{
5480 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_end:}}
5481 }
5482 }
5483 \tl_if_empty:NTF \l_tmpa_tl {
5484 \__stex_sproof_spfeq_end:
5485 }{
5486 \l_tmpa_tl
5487 }
5488 \end{stex_annotate_env}
5489 }
5490 }
5491
5492 \cs_new_protected:Nn \__stex_sproof_spfeq_start: {
5493 \titleemph{
5494 \tl_if_empty:NTF \spftitle {
5495 \spf@proof@kw
5496 }{
5497 \spftitle
5498 }
5499 }:
5500 }
5501 \cs_new_protected:Nn \__stex_sproof_spfeq_end: {\sproofend}
5502
5503 \newcommand\stexpatchspfeq[3] [] {
5504 \str_set:Nx \l_tmpa_str{ #1 }
5505 \str_if_empty:NTF \l_tmpa_str {
5506 \tl_set:Nn \__stex_sproof_spfeq_start: { #2 }
5507 \tl_set:Nn \__stex_sproof_spfeq_end: { #3 }
5508 }{
5509 \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_start:\endcsname{ #2 }
5510 \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_end:\endcsname{ #3 }
5511 }
5512 }
5513

```

(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.

```

5514 \newenvironment{sproof}[2] []{

```

```

5515 \let \premise \stex_proof_premise:
5516 \intarray_gzero:N \l__stex_sproof_counter_intarray
5517 \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
5518 \__stex_sproof_spf_args:n{#1}
5519 \stex_if_smsmode:TF {
5520   \str_if_empty:NF \spfid {
5521     \stex_ref_new_doc_target:n \spfid
5522   }
5523 }{
5524   \seq_clear:N \l_tmpa_seq
5525   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5526     \tl_if_empty:NF{ ##1 }{
5527       \stex_get_symbol:n { ##1 }
5528       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5529         \l_stex_get_symbol_uri_str
5530       }
5531     }
5532   }
5533   \exp_args:Nnnx
5534   \begin{stex_annotate_env}{sproof}{\seq_use:Nn \l_tmpa_seq {},}}
5535   \str_if_empty:NF \spftype {
5536     \stex_annotate_invisible:nnn{type}{\spftype}{}
5537   }
5538
5539   \clist_set:No \l_tmpa_clist \spftype
5540   \tl_clear:N \l_tmpa_tl
5541   \clist_map_inline:Nn \l_tmpa_clist {
5542     \tl_if_exist:cT {__stex_sproof_sproof_##1_start:}{
5543       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_start:}}
5544     }
5545     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5546       \tl_set:Nn \l_tmpa_tl {\use:n{}}
5547     }
5548   }
5549   \tl_if_empty:NTF \l_tmpa_tl {
5550     \__stex_sproof_sproof_start:
5551   }{
5552     \l_tmpa_tl
5553   }{~#2}
5554   \str_if_empty:NF \spfid {
5555     \stex_ref_new_doc_target:n \spfid
5556   }
5557   \begin{description}
5558 }
5559 \stex_smsmode_do:
5560 }{
5561   \stex_if_smsmode:F{
5562     \end{description}
5563     \clist_set:No \l_tmpa_clist \spftype
5564     \tl_clear:N \l_tmpa_tl
5565     \clist_map_inline:Nn \l_tmpa_clist {
5566       \tl_if_exist:cT {__stex_sproof_sproof_##1_end:}{
5567         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_end:}}
5568       }

```

```

5569     }
5570     \tl_if_empty:NTF \l_tmpa_tl {
5571       \__stex_sproof_sproof_end:
5572     }{
5573       \l_tmpa_tl
5574     }
5575     \end{stex_annotate_env}
5576   }
5577 }
5578
5579 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
5580   \par\noindent\titleemph{
5581     \tl_if_empty:NTF \spftype {
5582       \spf@proof@kw
5583     }{
5584       \spftype
5585     }
5586   }:
5587 }
5588 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
5589
5590 \newcommand\stexpatchproof[3] [] {
5591   \str_set:Nx \l_tmpa_str{ #1 }
5592   \str_if_empty:NTF \l_tmpa_str {
5593     \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
5594     \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
5595   }{
5596     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
5597     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
5598   }
5599 }

```

\spfidea

```

5600 \newcommand\spfidea[2] []{
5601   \__stex_sproof_spf_args:n{#1}
5602   \titleemph{
5603     \tl_if_empty:NTF \spftype {Proof~Idea}{
5604       \spftype
5605     }:
5606   }~#2
5607   \sproofend
5608 }

```

(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

```

5609 \newenvironment{spfstep}[1] []{
5610   \__stex_sproof_spf_args:n{#1}
5611   \stex_if_smsmode:TF {

```

```

5612 \str_if_empty:NF \spfid {
5613   \stex_ref_new_doc_target:n \spfid
5614 }
5615 }{
5616   \@in@omtexttrue
5617   \seq_clear:N \l_tmpa_seq
5618   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5619     \tl_if_empty:NF{ ##1 }{
5620       \stex_get_symbol:n { ##1 }
5621       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5622         \l_stex_get_symbol_uri_str
5623       }
5624     }
5625   }
5626   \exp_args:Nnnx
5627   \begin{stex_annotate_env}{spfstep}{\seq_use:Nn \l_tmpa_seq {,}}
5628   \str_if_empty:NF \spftype {
5629     \stex_annotate_invisible:nnn{type}{\spftype}{}
5630   }
5631   \clist_set:No \l_tmpa_clist \spftype
5632   \tl_set:Nn \l_tmpa_tl {
5633     \item[\sproofnumber]
5634     \bool_set_true:N \l__stex_sproof_inc_counter_bool
5635   }
5636   \clist_map_inline:Nn \l_tmpa_clist {
5637     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5638       \tl_clear:N \l_tmpa_tl
5639     }
5640   }
5641   \l_tmpa_tl
5642   \tl_if_empty:NF \spftitle {
5643     {(\titleemph{\spftitle})\enspace}
5644   }
5645   \str_if_empty:NF \spfid {
5646     \stex_ref_new_doc_target:n \spfid
5647   }
5648 }
5649 \stex_smsmode_do:
5650 \ignorespacesandpars
5651 }{
5652   \bool_if:NT \l__stex_sproof_inc_counter_bool {
5653     \__stex_sproof_inc_counter:
5654   }
5655   \stex_if_smsmode:F {
5656     \end{stex_annotate_env}
5657   }
5658 }

```

sproofcomment

```

5659 \newenvironment{sproofcomment}[1][]{
5660   \__stex_sproof_spf_args:n{#1}
5661   \clist_set:No \l_tmpa_clist \spftype
5662   \tl_set:Nn \l_tmpa_tl {
5663     \item[\sproofnumber]

```

```

5664 \bool_set_true:N \l__stex_sproof_inc_counter_bool
5665 }
5666 \clist_map_inline:Nn \l_tmpa_clist {
5667   \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5668     \tl_clear:N \l_tmpa_tl
5669   }
5670 }
5671 \l_tmpa_tl
5672 }{
5673   \bool_if:NT \l__stex_sproof_inc_counter_bool {
5674     \__stex_sproof_inc_counter:
5675   }
5676 }

```

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.

```

5677 \newenvironment{subproof}[2][]{
5678   \__stex_sproof_spf_args:n{#1}
5679   \stex_if_smsmode:TF{
5680     \str_if_empty:NF \spfid {
5681       \stex_ref_new_doc_target:n \spfid
5682     }
5683   }{
5684     \seq_clear:N \l_tmpa_seq
5685     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5686       \tl_if_empty:NF{ ##1 }{
5687         \stex_get_symbol:n { ##1 }
5688         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5689           \l_stex_get_symbol_uri_str
5690         }
5691       }
5692     }
5693     \exp_args:Nnnx
5694     \begin{stex_annotate_env}{subproof}{\seq_use:Nn \l_tmpa_seq {,}}
5695     \str_if_empty:NF \spftype {
5696       \stex_annotate_invisible:nnn{type}{\spftype}{ }
5697     }
5698
5699     \clist_set:No \l_tmpa_clist \spftype
5700     \tl_set:Nn \l_tmpa_tl {
5701       \item[\sproofnumber]
5702       \bool_set_true:N \l__stex_sproof_inc_counter_bool
5703     }
5704     \clist_map_inline:Nn \l_tmpa_clist {
5705       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5706         \tl_clear:N \l_tmpa_tl
5707       }
5708     }
5709     \l_tmpa_tl
5710     \tl_if_empty:NF \spftitle {
5711       {(\titleemph{\spftitle})\enspace}
5712     }

```

```

5713     {~#2}
5714     \str_if_empty:NF \spfid {
5715         \stex_ref_new_doc_target:n \spfid
5716     }
5717 }
5718 \__stex_sproof_add_counter:
5719 \stex_smsmode_do:
5720 }{
5721     \__stex_sproof_remove_counter:
5722     \bool_if:NT \l__stex_sproof_inc_counter_bool {
5723         \__stex_sproof_inc_counter:
5724     }
5725     \stex_if_smsmode:F{
5726         \end{stex_annotate_env}
5727     }
5728 }

```

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

```

5729 \newenvironment{spfcases}[2][]{
5730     \tl_if_empty:nTF{#1}{
5731         \begin{subproof}[method=by-cases]{#2}
5732     }{
5733         \begin{subproof}[#1,method=by-cases]{#2}
5734     }
5735 }{
5736     \end{subproof}
5737 }

```

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

```

5738 \newenvironment{spfcase}[2][]{
5739     \__stex_sproof_spf_args:n{#1}
5740     \stex_if_smsmode:TF {
5741         \str_if_empty:NF \spfid {
5742             \stex_ref_new_doc_target:n \spfid
5743         }
5744     }{
5745         \seq_clear:N \l_tmpa_seq
5746         \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5747             \tl_if_empty:nF{ ##1 }{
5748                 \stex_get_symbol:n { ##1 }
5749                 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5750                     \l_stex_get_symbol_uri_str
5751                 }
5752             }
5753         }
5754         \exp_args:Nnnx
5755         \begin{stex_annotate_env}{spfcase}{\seq_use:Nn \l_tmpa_seq {,}}
5756         \str_if_empty:NF \spftype {
5757             \stex_annotate_invisible:nnn{type}{\spftype}{}}
5758     }
5759     \clist_set:Nn \l_tmpa_clist \spftype
5760     \tl_set:Nn \l_tmpa_tl {
5761         \item[\sproofnumber]

```

```

5762     \bool_set_true:N \l__stex_sproof_inc_counter_bool
5763   }
5764   \clist_map_inline:Nn \l_tmpa_clist {
5765     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5766       \tl_clear:N \l_tmpa_tl
5767     }
5768   }
5769   \l_tmpa_tl
5770   \tl_if_empty:nF{#2}{
5771     \titleemph{#2}:~
5772   }
5773 }
5774 \__stex_sproof_add_counter:
5775 \stex_smsmode_do:
5776 ){
5777   \__stex_sproof_remove_counter:
5778   \bool_if:NT \l__stex_sproof_inc_counter_bool {
5779     \__stex_sproof_inc_counter:
5780   }
5781   \stex_if_smsmode:F{
5782     \clist_set:No \l_tmpa_clist \spftype
5783     \tl_set:Nn \l_tmpa_tl{\sproofend}
5784     \clist_map_inline:Nn \l_tmpa_clist {
5785       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5786         \tl_clear:N \l_tmpa_tl
5787       }
5788     }
5789     \l_tmpa_tl
5790     \end{stex_annotate_env}
5791   }
5792 }

```

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

```

5793 \newcommand\spfcasesketch[3][]{
5794   \begin{spfcase}[#1]{#2}#3\end{spfcase}
5795 }

```

### 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.

```

5796 \keys_define:nn { stex / just }{
5797   id      .str_set:x:N = \l__stex_sproof_just_id_str,
5798   method  .tl_set:N    = \l__stex_sproof_just_method_tl,
5799   premises .tl_set:N    = \l__stex_sproof_just_premises_tl,
5800   args    .tl_set:N    = \l__stex_sproof_just_args_tl
5801 }

```

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**

```
5802 \newenvironment{justification}[1] [] {}{}
```

**\premise**

```
5803 \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.

```
5804 \newcommand\justarg[2] [] {#2}
```

```
5805 \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

```
5806 <*package>
5807
5808 %%%%%%%%%% others.dtx %%%%%%%%%%
5809
5810 <@@=stex_others>
    Warnings and error messages
5811 % None

\MSC Math subject classifier

5812 \NewDocumentCommand \MSC {m} {
5813 % TODO
5814 }

(End definition for \MSC. This function is documented on page ??.)
    Patching tikzinput, if loaded
5815 \@ifpackageloaded{tikzinput}{
5816 \RequirePackage{stex-tikzinput}
5817 }{}
5818 </package>
```

## Chapter 35

# STEX -Metatheory Implementation

```
5819 <*package>
5820 <@@=stex_modules>
5821
5822 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
5823
5824 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
5825 \begingroup
5826 \stex_module_setup:nn{
5827   ns=\c_stex_metatheory_ns_str,
5828   meta=NONE
5829 }{Metatheory}
5830 \stex_reactivate_macro:N \symdecl
5831 \stex_reactivate_macro:N \notation
5832 \stex_reactivate_macro:N \symdef
5833 \ExplSyntaxOff
5834 \csname stex_suppress_html:n\endcsname{
5835   % is-a (a:A, a \in A, a is an A, etc.)
5836   \symdecl{isa}[args=ai]
5837   \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
5838   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
5839   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
5840
5841   % bind (\forall, \Pi, \lambda etc.)
5842   \symdecl{bind}[args=Bi]
5843   \notation{bind}[forall]{\comp\forall #1.;#2}{##1 \comp, ##2}
5844   \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
5845   \notation{bind}[depfun]{\comp( #1 \comp{}\;\to\;} #2}{##1 \comp, ##2}
5846
5847   % implicit bind
5848   \symdef{implicitbind}[args=Bi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
5849
5850   % dummy variable
5851   \symdecl{dummyvar}
5852   \notation{dummyvar}[underscore]{\comp\_}
5853   \notation{dummyvar}[dot]{\comp\cdot}
```

```

5854 \notation{dummyvar}[dash]{\comp{\rm --}}
5855
5856 %fromto (function space, Hom-set, implication etc.)
5857 \symdecl{fromto}[args=ai]
5858 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
5859 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
5860
5861 % mapto (lambda etc.)
5862 \symdecl{mapto}[args=Bi]
5863 \notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
5864 \notation{mapto}[lambda]{\comp\lambda #1 \comp.; #2}{#1 \comp, #2}
5865 \notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.; #2}{#1 \comp, #2}
5866
5867 % function/operator application
5868 \symdecl{apply}[args=ia]
5869 \notation{apply}[prec=0;0x\infpres,parens]{#1 \comp( #2 \comp)}{##1 \comp, ##2}
5870 \notation{apply}[prec=0;0x\infpres,lambda]{#1 \; #2 }{##1 \; ; ##2}
5871
5872 % ‘‘type’’ of all collections (sets,classes,types,kinds)
5873 \symdecl{metacollection}
5874 \notation{metacollection}[U]{\comp{\mathcal{U}}}
5875 \notation{metacollection}[set]{\comp{\textsf{Set}}}
5876
5877 % collection of propositions/booleans/truth values
5878 \symdecl{prop}[name=proposition]
5879 \notation{prop}[prop]{\comp{\rm prop}}
5880 \notation{prop}[B00L]{\comp{\rm B00L}}
5881
5882 % sequences
5883 \symdecl{seqtype}[args=1]
5884 \notation{seqtype}[kleene]{#1^{\comp\ast}}
5885
5886 \symdef{sequence-index}[args=2,li,prec=nobrackets]{#{1}_{#2}}
5887 \notation{sequence-index}[ui,prec=nobrackets]{#{1}^{\comp\ast}}
5888
5889 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{,\ellipses}}{##1\comp,##2}
5890 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{,\ellipses,}#2}{##1\comp,##2}
5891 \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{,\ellipses,}#2\comp{,\ellipses,}#3}
5892
5893 % letin (‘‘let’’, local definitions, variable substitution)
5894 \symdecl{letin}[args=bii]
5895 \notation{letin}[let]{\comp{\rm let}}\;#1\comp{=}#2\;\comp{\rm in}}\;#3}
5896 \notation{letin}[subst]{#3 \comp[ #1 \comp/ #2 \comp]}
5897 \notation{letin}[frac]{#3 \comp[ \frac{#2}{#1} \comp]}
5898
5899 % structures
5900 \symdecl*{module-type}[args=1]
5901 \notation{module-type}{\mathtt{MOD} #1}
5902 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
5903 \notation{mathstruct}[angle,prec=nobrackets]{\comp\angle #1 \comp\rangle}{##1 \comp, ##2}
5904
5905 }
5906 \ExplSyntaxOn
5907 \stex_add_to_current_module:n{

```

```

5908 \let\nappa\apply
5909 \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
5910 \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
5911 \def\livar{\csname sequence-index\endcsname[li]}
5912 \def\uivar{\csname sequence-index\endcsname[ui]}
5913 \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
5914 \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
5915 \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
5916 }
5917 \__stex_modules_end_module:
5918 \endgroup
5919 </package>

```

## Chapter 36

# Tikzinput Implementation

```
5920 <*package>
5921
5922 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
5923
5924 \ProvidesExplPackage{tikzinput}{2022/02/26}{3.0.1}{tikzinput package}
5925 \RequirePackage{l3keys2e}
5926
5927 \keys_define:nn { tikzinput } {
5928   image .bool_set:N = \c_tikzinput_image_bool,
5929   image .default:n = false ,
5930   unknown .code:n = {}
5931 }
5932
5933 \ProcessKeysOptions { tikzinput }
5934
5935 \bool_if:NTF \c_tikzinput_image_bool {
5936   \RequirePackage{graphicx}
5937
5938   \providecommand\usetikzlibrary[]{}
5939   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
5940 }{
5941   \RequirePackage{tikz}
5942   \RequirePackage{standalone}
5943
5944   \newcommand \tikzinput [2] [] {
5945     \setkeys{Gin}{#1}
5946     \ifx \Gin@ewidth \Gin@exclamation
5947       \ifx \Gin@eheight \Gin@exclamation
5948         \input { #2 }
5949       \else
5950         \resizebox{!}{ \Gin@eheight }{
5951           \input { #2 }
5952         }
5953       \fi
5954     \else
5955       \ifx \Gin@eheight \Gin@exclamation
5956         \resizebox{ \Gin@ewidth }{!}{
5957           \input { #2 }
```

```

5958     }
5959     \else
5960     \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
5961     \input { #2 }
5962     }
5963     \fi
5964     \fi
5965   }
5966 }
5967
5968 \newcommand \ctikzinput [2] [] {
5969   \begin{center}
5970     \tikzinput [1] {#2}
5971   \end{center}
5972 }
5973
5974 \@ifpackageloaded{stex}{
5975   \RequirePackage{stex-tikzinput}
5976 }{}
5977
5978 </package>
5979 <*stex>
5980 \ProvidesExplPackage{stex-tikzinput}{2022/02/26}{3.0.1}{stex-tikzinput}
5981 \RequirePackage{stex}
5982 \RequirePackage{tikzinput}
5983
5984 \newcommand\mhtikzinput [2] [] {%
5985   \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
5986   \stex_in_repository:nn\Gin@mhrepos{
5987     \tikzinput [1]{\mhp{##1}{#2}}
5988   }
5989 }
5990 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [1] {#2}\end{center}}
5991 </stex>

```

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

## 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.

```
5992 \*cls)
5993 \@@=document_structure)
5994 \ProvidesExplClass{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure Class}
5995 \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`

```
5996 \keys_define:nn{ document-structure / pkg }{
5997   class      .str_set_x:N = \c_document_structure_class_str,
5998   minimal    .bool_set:N = \c_document_structure_minimal_bool,
5999   report     .code:n      = {
6000     \ClassWarning{document-structure}{the option 'report' is deprecated, use 'class=report',
6001     \str_set:Nn \c_document_structure_class_str {report}
6002   },
6003   book       .code:n      = {
6004     \ClassWarning{document-structure}{the option 'book' is deprecated, use 'class=book', ins
6005     \str_set:Nn \c_document_structure_class_str {book}
6006   },
6007   bookpart   .code:n      = {
6008     \ClassWarning{document-structure}{the option 'bookpart' is deprecated, use 'class=book,t
6009     \str_set:Nn \c_document_structure_class_str {book}
6010     \str_set:Nn \c_document_structure_topsect_str {chapter}
6011   },
```

```

6012 docopt      .str_set_x:N = \c_document_structure_docopt_str,
6013 unknown     .code:n      = {
6014   \PassOptionsToPackage{ \CurrentOption }{ document-structure }
6015 }
6016 }
6017 \ProcessKeysOptions{ document-structure / pkg }
6018 \str_if_empty:NT \c_document_structure_class_str {
6019   \str_set:Nn \c_document_structure_class_str {article}
6020 }
6021 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
6022   {\c_document_structure_class_str}
6023

```

### 37.3 Beefing up the document environment

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

```

6024 \RequirePackage{document-structure}
6025 \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>

```

6026 \keys_define:nn { document-structure / document }{
6027   id .str_set_x:N = \c_document_structure_document_id_str
6028 }
6029 \let\__document_structure_orig_document=\document
6030 \renewcommand{\document}[1][]{
6031   \keys_set:nn{ document-structure / document }{ #1 }
6032   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
6033   \__document_structure_orig_document
6034 }

```

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

```

6035 }
6036 \</cls>

```

### 37.4 Implementation: document-structure Package

```

6037 \<package>
6038 \ProvidesExplPackage{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure}
6039 \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



```

6040
6041 \keys_define:nn{ document-structure / pkg }{
6042   class      .str_set_x:N = \c_document_structure_class_str,
6043   topsect    .str_set_x:N = \c_document_structure_topsect_str,
6044   % showignores .bool_set:N = \c_document_structure_showignores_bool,
6045 }
6046 \ProcessKeysOptions{ document-structure / pkg }
6047 \str_if_empty:NT \c_document_structure_class_str {
6048   \str_set:Nn \c_document_structure_class_str {article}
6049 }
6050 \str_if_empty:NT \c_document_structure_topsect_str {
6051   \str_set:Nn \c_document_structure_topsect_str {section}
6052 }

```

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

```

6053 \RequirePackage{xspace}
6054 \RequirePackage{comment}
6055 \AddToHook{begindocument}{
6056   \ltx@ifpackageloaded{babel}{
6057     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6058     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6059       \makeatletter\input{document-structure-ngerman.ldf}\makeatother
6060     }
6061   }{}
6062 }

```

`\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.

```

6063 \int_new:N \l_document_structure_section_level_int
6064 \str_case:VnF \c_document_structure_topsect_str {
6065   {part}}{
6066     \int_set:Nn \l_document_structure_section_level_int {0}
6067   }
6068   {chapter}{
6069     \int_set:Nn \l_document_structure_section_level_int {1}
6070   }
6071 }{
6072   \str_case:VnF \c_document_structure_class_str {
6073     {book}}{
6074       \int_set:Nn \l_document_structure_section_level_int {0}
6075     }
6076     {report}}{
6077       \int_set:Nn \l_document_structure_section_level_int {0}
6078     }
6079   }{
6080     \int_set:Nn \l_document_structure_section_level_int {2}
6081   }
6082 }

```

## 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

```
6083 \def\current@section@level{document}%
6084 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
6085 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
6086 \cs_new_protected:Npn \skipomgroup {
6087   \ifcase\l_document_structure_section_level_int
6088   \or\stepcounter{part}
6089   \or\stepcounter{chapter}
6090   \or\stepcounter{section}
6091   \or\stepcounter{subsection}
6092   \or\stepcounter{subsubsection}
6093   \or\stepcounter{paragraph}
6094   \or\stepcounter{subparagraph}
6095   \fi
6096 }
```

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

`blindfragment`

```
6097 \newcommand\at@begin@blindomgroup[1]{%
6098 \newenvironment{blindfragment}
6099 {
6100   \int_incr:N\l_document_structure_section_level_int
6101   \at@begin@blindomgroup\l_document_structure_section_level_int
6102 }{}}
```

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

```
6103 \newcommand\omgroup@nonum[2]{
6104   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
6105   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
6106 }
```

*(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.

```
6107 \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.

```

6108 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
6109   \@nameuse{#1}{#2}
6110 }{
6111   \cs_if_exist:NTF\rdfmata@sectioning{
6112     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
6113   }{
6114     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
6115   }
6116 }
6117 %\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\omgroup@id
6118 }

```

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

**sfragment**

```

6119 \keys_define:nn { document-structure / omgroup }{
6120   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
6121   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
6122   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
6123   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
6124   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
6125   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
6126   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
6127   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
6128   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
6129   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
6130 }
6131 \cs_new_protected:Nn \__document_structure_omgroup_args:n {
6132   \str_clear:N \l__document_structure_omgroup_id_str
6133   \str_clear:N \l__document_structure_omgroup_date_str
6134   \clist_clear:N \l__document_structure_omgroup_creators_clist
6135   \clist_clear:N \l__document_structure_omgroup_contributors_clist
6136   \tl_clear:N \l__document_structure_omgroup_srccite_tl
6137   \tl_clear:N \l__document_structure_omgroup_type_tl
6138   \tl_clear:N \l__document_structure_omgroup_short_tl
6139   \tl_clear:N \l__document_structure_omgroup_display_tl
6140   \tl_clear:N \l__document_structure_omgroup_intro_tl
6141   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
6142   \keys_set:nn { document-structure / omgroup } { #1 }
6143 }

```

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.

```

6144 \newif\if@mainmatter\@mainmattertrue
6145 \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.

```

6146 \keys_define:nn { document-structure / sectioning }{
6147   name .str_set_x:N = \l__document_structure_sect_name_str ,
6148   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
6149   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
6150   clear .default:n = {true} ,
6151   num .bool_set:N = \l__document_structure_sect_num_bool ,

```

```

6152   num      .default:n      = {true}
6153 }
6154 \cs_new_protected:Nn \__document_structure_sect_args:n {
6155   \str_clear:N \l__document_structure_sect_name_str
6156   \str_clear:N \l__document_structure_sect_ref_str
6157   \bool_set_false:N \l__document_structure_sect_clear_bool
6158   \bool_set_false:N \l__document_structure_sect_num_bool
6159   \keys_set:nn { document-structure / sectioning } { #1 }
6160 }
6161 \newcommand\omdoc@sectioning[3][]{
6162   \__document_structure_sect_args:n {#1}
6163   \let\omdoc@sect@name\l__document_structure_sect_name_str
6164   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
6165   \if@mainmatter% numbering not overridden by frontmatter, etc.
6166     \bool_if:NTF \l__document_structure_sect_num_bool {
6167       \omgroup@num{#2}{#3}
6168     }{
6169       \omgroup@nonum{#2}{#3}
6170     }
6171     \def\current@section@level{\omdoc@sect@name}
6172   \else
6173     \omgroup@nonum{#2}{#3}
6174   \fi
6175 }% 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.

```

6176 \newcommand\omgroup@redefine@addtocontents[1]{%
6177 %\edef\__document_structureimport{#1}%
6178 %\@for\@I:=\__document_structureimport\do{%
6179 %\edef\@path{\csname module@\@I @path\endcsname}%
6180 %\@ifundefined{tf@toc}\relax%
6181 %   {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
6182 %\ifx\hyper@anchor\undefined% hyperref.sty loaded?
6183 %\def\addcontentsline##1##2##3{%
6184 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6185 %\else% hyperref.sty not loaded
6186 %\def\addcontentsline##1##2##3{%
6187 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6188 %\fi
6189 }% 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.

```

6190 \newenvironment{sfragment}[2][]{% keys, title
6191 {
6192   \__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.

```

6193   \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
6194     \omgroup@redefine@addtocontents{
6195       \@ifundefined{module@id}\used@modules%

```

```

6196     %{\@ifundefined{module@}\module@id @path}{\used@modules}\module@id}
6197   }
6198 }

```

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

```

6199 \int_incr:N\l_document_structure_section_level_int
6200 \ifcase\l_document_structure_section_level_int
6201   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
6202   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
6203   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
6204   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
6205   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
6206   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
6207   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
6208 \fi
6209 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
6210 \str_if_empty:NF \l__document_structure_omgroup_id_str {
6211   \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
6212 }
6213 }% for customization
6214 {}

```

and finally, we localize the sections

```

6215 \newcommand\omdoc@part@kw{Part}
6216 \newcommand\omdoc@chapter@kw{Chapter}
6217 \newcommand\omdoc@section@kw{Section}
6218 \newcommand\omdoc@subsection@kw{Subsection}
6219 \newcommand\omdoc@subsubsection@kw{Subsubsection}
6220 \newcommand\omdoc@paragraph@kw{paragraph}
6221 \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`

```

6222 \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).

```

6223 \cs_if_exist:NTF\frontmatter{
6224   \let\__document_structure_orig_frontmatter\frontmatter
6225   \let\frontmatter\relax
6226 }{
6227   \tl_set:Nn\__document_structure_orig_frontmatter{
6228     \clearpage
6229     \@mainmatterfalse
6230     \pagenumbering{roman}

```

```

6231 }
6232 }
6233 \cs_if_exist:NTF\backmatter{
6234   \let\__document_structure_orig_backmatter\backmatter
6235   \let\backmatter\relax
6236 }{
6237   \tl_set:Nn\__document_structure_orig_backmatter{
6238     \clearpage
6239     \@mainmatterfalse
6240     \pagenumbering{roman}
6241   }
6242 }

```

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.

```

6243 \newenvironment{frontmatter}{
6244   \__document_structure_orig_frontmatter
6245 }{
6246   \cs_if_exist:NTF\mainmatter{
6247     \mainmatter
6248   }{
6249     \clearpage
6250     \@mainmattertrue
6251     \pagenumbering{arabic}
6252   }
6253 }

```

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

```

6254 \newenvironment{backmatter}{
6255   \__document_structure_orig_backmatter
6256 }{
6257   \cs_if_exist:NTF\mainmatter{
6258     \mainmatter
6259   }{
6260     \clearpage
6261     \@mainmattertrue
6262     \pagenumbering{arabic}
6263   }
6264 }

```

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

```

6265 \@mainmattertrue\pagenumbering{arabic}

```

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

```

6266 \def \c__document_structure_document_str{document}
6267 \newcommand\afterprematurestop{}
6268 \def\prematurestop@endomgroup{
6269   \unless\ifx\@currenvir\c__document_structure_document_str
6270     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter\expandafter\expandafter
6271     \expandafter\prematurestop@endomgroup

```

```

6272 \fi
6273 }
6274 \providecommand\prematurestop{
6275   \message{Stopping~sTeX~processing~prematurely}
6276   \prematurestop@endumgroup
6277   \afterprematurestop
6278   \end{document}
6279 }

```

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

## 37.8 Global Variables

**\setSGvar** set a global variable

```

6280 \RequirePackage{etoolbox}
6281 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

**\useSGvar** use a global variable

```

6282 \newrobustcmd\useSGvar[1]{%
6283   \@ifundefined{sTeX@Gvar@#1}
6284   {\PackageError{document-structure}
6285     {The sTeX Global variable #1 is undefined}
6286     {set it with \protect\setSGvar}}
6287   \@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.

```

6288 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
6289   \@ifundefined{sTeX@Gvar@#1}
6290   {\PackageError{document-structure}
6291     {The sTeX Global variable #1 is undefined}
6292     {set it with \protect\setSGvar}}
6293   {\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.

```
6294 \*cls)
6295 \@@=notesslides)
6296 \ProvidesExplClass{notesslides}{2022/02/28}{3.1.0}{notesslides Class}
6297 \RequirePackage{13keys2e}
6298
6299 \keys_define:nn{notesslides / cls}{
6300   class .code:n = {
6301     \PassOptionsToClass{\CurrentOption}{document-structure}
6302     \str_if_eq:nnT{#1}{book}{
6303       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
6304     }
6305     \str_if_eq:nnT{#1}{report}{
6306       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
6307     }
6308   },
6309   notes .bool_set:N = \c__notesslides_notes_bool ,
6310   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
6311   unknown .code:n = {
6312     \PassOptionsToClass{\CurrentOption}{document-structure}
6313     \PassOptionsToClass{\CurrentOption}{beamer}
6314     \PassOptionsToPackage{\CurrentOption}{notesslides}
6315   }
6316 }
6317 \ProcessKeysOptions{ notesslides / cls }
6318 \bool_if:NTF \c__notesslides_notes_bool {
6319   \PassOptionsToPackage{notes=true}{notesslides}
6320 }{
6321   \PassOptionsToPackage{notes=false}{notesslides}
6322 }
6323 \</cls)
```



now we do the same for the notesslides package.

```

6324 <*package>
6325 \ProvidesExplPackage{notesslides}{2022/02/28}{3.1.0}{notesslides Package}
6326 \RequirePackage{13keys2e}
6327
6328 \keys_define:nn{notesslides / pkg}{
6329   topsect      .str_set_x:N = \c_notesslides_topsect_str,
6330   defaulttopsect .str_set_x:N = \c_notesslides_defaulttopsec_str,
6331   notes        .bool_set:N = \c_notesslides_notes_bool ,
6332   slides       .code:n      = { \bool_set_false:N \c_notesslides_notes_bool },
6333   sectocframes .bool_set:N = \c_notesslides_sectocframes_bool ,
6334   frameimages  .bool_set:N = \c_notesslides_frameimages_bool ,
6335   fiboxed      .bool_set:N = \c_notesslides_fiboxed_bool ,
6336   noproblems   .bool_set:N = \c_notesslides_noproblems_bool,
6337   unknown      .code:n      = {
6338     \PassOptionsToClass{\CurrentOption}{stex}
6339     \PassOptionsToClass{\CurrentOption}{tikzinput}
6340   }
6341 }
6342 \ProcessKeysOptions{ notesslides / pkg }
6343 \newif\ifnotes
6344 \bool_if:NTF \c_notesslides_notes_bool {
6345   \notesttrue
6346 }{
6347   \notesfalse
6348 }
6349

```

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

```

6350 \str_if_empty:NTF \c_notesslides_topsect_str {
6351   \str_set_eq:NN \__notesslidestopsect \c_notesslides_defaulttopsec_str
6352 }{
6353   \str_set_eq:NN \__notesslidestopsect \c_notesslides_topsect_str
6354 }
6355 </package>

```

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

```

6356 <*cls>
6357 \bool_if:NTF \c_notesslides_notes_bool {
6358   \LoadClass{document-structure}
6359 }{
6360   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
6361   \newcounter{Item}
6362   \newcounter{paragraph}
6363   \newcounter{subparagraph}
6364   \newcounter{Hfootnote}
6365   \RequirePackage{document-structure}
6366 }

```

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

```

6367 \RequirePackage{notesslides}
6368 </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.

```

6369 <*package>
6370 \bool_if:NT \c__notesslides_notes_bool {
6371   \RequirePackage{a4wide}
6372   \RequirePackage{marginnote}
6373   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
6374   \RequirePackage{mdframed}
6375   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
6376   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
6377 }
6378 \RequirePackage{stex-tikzinput}
6379 \RequirePackage{etoolbox}
6380 \RequirePackage{amssymb}
6381 \RequirePackage{amsmath}
6382 \RequirePackage{comment}
6383 \RequirePackage{textcomp}
6384 \RequirePackage{url}
6385 \RequirePackage{graphicx}
6386 \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>

```

6387 \bool_if:NT \c__notesslides_notes_bool {
6388   \renewcommand\usetheme[2][\usepackage[#1]{beamernotestheme#2}]
6389 }
6390
6391
6392 \NewDocumentCommand \libusetheme {0{} m} {
6393   \bool_if:NTF \c__notesslides_notes_bool {
6394     \libusepackage[#1]{beamernotestheme#2}
6395   }{
6396     \libusepackage[#1]{beamertheme#2}
6397   }
6398 }

```

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

```

6399 \newcounter{slide}
6400 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
6401 \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.

```

6402 \bool_if:NTF \c__notesslides_notes_bool {
6403   \renewenvironment{note}{\ignorespaces}{}
6404 }{
6405   \excludecomment{note}
6406 }

```

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.

```

6407 \bool_if:NT \c__notesslides_notes_bool {
6408   \newlength{\slideframewidth}
6409   \setlength{\slideframewidth}{1.5pt}

```

**frame** We first define the keys.

```

6410 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
6411   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
6412     \bool_set_true:N #1
6413   }{
6414     \bool_set_false:N #1
6415   }
6416 }
6417 \keys_define:nn{notesslides / frame}{
6418   label .str_set_x:N = \l__notesslides_frame_label_str,
6419   allowframebreaks .code:n = {
6420     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
6421   },
6422   allowdisplaybreaks .code:n = {
6423     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
6424   },
6425   fragile .code:n = {
6426     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
6427   },
6428   shrink .code:n = {
6429     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
6430   },
6431   squeeze .code:n = {
6432     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
6433   },
6434   t .code:n = {
6435     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
6436   },
6437 }
6438 \cs_new_protected:Nn \__notesslides_frame_args:n {
6439   \str_clear:N \l__notesslides_frame_label_str
6440   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
6441   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
6442   \bool_set_true:N \l__notesslides_frame_fragile_bool
6443   \bool_set_true:N \l__notesslides_frame_shrink_bool
6444   \bool_set_true:N \l__notesslides_frame_squeeze_bool
6445   \bool_set_true:N \l__notesslides_frame_t_bool

```

```

6446 \keys_set:nn { notesslides / frame }{ #1 }
6447 }

```

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

```

6448 \renewenvironment{frame}[1][]{
6449 \_notesslides_frame_args:n{#1}
6450 \sffamily
6451 \stepcounter{slide}
6452 \def\@currentlabel{\theslide}
6453 \str_if_empty:NF \l__notesslides_frame_label_str {
6454 \label{\l__notesslides_frame_label_str}
6455 }

```

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

```

6456 \def\itemize@level{outer}
6457 \def\itemize@outer{outer}
6458 \def\itemize@inner{inner}
6459 \renewcommand\newpage{\addtocounter{framenumber}{1}}
6460 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
6461 \renewenvironment{itemize}{
6462 \ifx\itemize@level\itemize@outer
6463 \def\itemize@label{$\rhd$}
6464 \fi
6465 \ifx\itemize@level\itemize@inner
6466 \def\itemize@label{$\scriptstyle\rhd$}
6467 \fi
6468 \begin{list}
6469 {\itemize@label}
6470 {\setlength{\labelsep}{.3em}
6471 \setlength{\labelwidth}{.5em}
6472 \setlength{\leftmargin}{1.5em}
6473 }
6474 \edef\itemize@level{\itemize@inner}
6475 }{
6476 \end{list}
6477 }

```

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

```

6478 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth]
6479 }{
6480 \medskip\miko@slidelabel\end{mdframed}
6481 }

```

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

`\frametitle`

```

6482 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
6483 }

```

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

EdN:15

`\pause`

```

15
6484 \bool_if:NT \c__notesslides_notes_bool {
6485 \newcommand\pause{}
6486 }

```

---

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

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

**nparagraph**

```
6487 \bool_if:NTF \c__notesslides_notes_bool {
6488   \newenvironment{nparagraph}[1] [] {\begin{sparagraph}[#1]}\end{sparagraph}}
6489 }{
6490   \excludecomment{nparagraph}
6491 }
```

**nfragment**

```
6492 \bool_if:NTF \c__notesslides_notes_bool {
6493   \newenvironment{nfragment}[2] [] {\begin{sfragment}[#1]{#2}}\end{sfragment}}
6494 }{
6495   \excludecomment{nfragment}
6496 }
```

**ndefinition**

```
6497 \bool_if:NTF \c__notesslides_notes_bool {
6498   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}\end{sdefinition}}
6499 }{
6500   \excludecomment{ndefinition}
6501 }
```

**nassertion**

```
6502 \bool_if:NTF \c__notesslides_notes_bool {
6503   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}\end{sassertion}}
6504 }{
6505   \excludecomment{nassertion}
6506 }
```

**nsproof**

```
6507 \bool_if:NTF \c__notesslides_notes_bool {
6508   \newenvironment{nsproof}[2] [] {\begin{sproof}[#1]{#2}}\end{sproof}}
6509 }{
6510   \excludecomment{nsproof}
6511 }
```

**nexample**

```
6512 \bool_if:NTF \c__notesslides_notes_bool {
6513   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}\end{sexample}}
6514 }{
6515   \excludecomment{nexample}
6516 }
```

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

```
6517 \def\inputref@preskip{\smallskip}
6518 \def\inputref@postskip{\medskip}
```

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

`\inputref*`

```
6519 \let\orig@inputref\inputref
6520 \def\inputref{\@ifstar\ninputref\orig@inputref}
6521 \newcommand\ninputref[2][] {
6522   \bool_if:NT \c__notesslides_notes_bool {
6523     \orig@inputref[#1]{#2}
6524   }
6525 }
```

(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>}`.

```
6526 \newlength{\slidelogoheight}
6527
6528 \bool_if:NTF \c__notesslides_notes_bool {
6529   \setlength{\slidelogoheight}{.4cm}
6530 }{
6531   \setlength{\slidelogoheight}{1cm}
6532 }
6533 \newsavebox{\slidelogo}
6534 \sbox{\slidelogo}{\sTeX}
6535 \newrobustcmd{\setslidelogo}[1]{
6536   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
6537 }
```

(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.

```
6538 \def\source{Michael Kohlhase}% customize locally
6539 \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.

```
6540 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
6541 \newsavebox{\cclogo}
6542 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
6543 \newif\ifcchref\cchreffalse
6544 \AtBeginDocument{
6545   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
6546 }
6547 \def\licensing{
6548   \ifcchref
```

```

6549 \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
6550 \else
6551   {\usebox{\cclogo}}
6552 \fi
6553 }
6554 \newrobustcmd{\setlicensing}[2][]{
6555   \def\@url{#1}
6556   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
6557   \ifx\@url\@empty
6558     \def\licensing{\usebox{\cclogo}}
6559   \else
6560     \def\licensing{
6561       \ifcchref
6562         \href{#1}{\usebox{\cclogo}}
6563       \else
6564         {\usebox{\cclogo}}
6565       \fi
6566     }
6567   \fi
6568 }

```

(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>

```

6569 \newrobustcmd\miko@slidelabel{
6570   \vbox to \slidelogoheight{
6571     \vss\hbox to \slidewidth
6572     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
6573   }
6574 }

```

(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.

```

6575 \def\Gin@mhrepos{}
6576 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
6577 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
6578 \newrobustcmd\frameimage[2][]{
6579   \stepcounter{slide}
6580   \bool_if:NT \c__notesslides_frameimages_bool {
6581     \def\Gin@ewidth{}\setkeys{Gin}{#1}
6582     \bool_if:NF \c__notesslides_notes_bool { \vfill }
6583     \begin{center}
6584       \bool_if:NTF \c__notesslides_fiboxed_bool {
6585         \fbox{
6586           \ifx\Gin@ewidth\@empty
6587             \ifx\Gin@mhrepos\@empty
6588               \mhgraphics[width=\slidewidth,#1]{#2}
6589             \else

```

---

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

```

6590         \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
6591     \fi
6592 \else% Gin@ewidth empty
6593     \ifx\Gin@mhrepos\@empty
6594         \mhgraphics[#1]{#2}
6595     \else
6596         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
6597     \fi
6598 \fi% Gin@ewidth empty
6599 }
6600 }{
6601     \ifx\Gin@ewidth\@empty
6602         \ifx\Gin@mhrepos\@empty
6603             \mhgraphics[width=\slidewidth,#1]{#2}
6604         \else
6605             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
6606         \fi
6607         \ifx\Gin@mhrepos\@empty
6608             \mhgraphics[#1]{#2}
6609         \else
6610             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
6611         \fi
6612     \fi% Gin@ewidth empty
6613 }
6614 \end{center}
6615 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
6616 \bool_if:NF \c__notesslides_notes_bool { \vfill }
6617 }
6618 } % 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.

```

6619 \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

```

6620 \AddToHook{begindocument}{
6621     \definecolor{green}{rgb}{0,.5,0}
6622     \definecolor{purple}{cmyk}{.3,1,0,.17}
6623 }

```

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`.

```

6624 % \def\STpresent#1{\textcolor{blue}{#1}}
6625 \def\defemph#1{\textcolor{magenta}{#1}}
6626 \def\symrefemph#1{\textcolor{cyan}{#1}}
6627 \def\compemph#1{\textcolor{blue}{#1}}
6628 \def\titleemph#1{\textcolor{blue}{#1}}
6629 \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.

```

6630 \pgfdeclareimage[width=.8em]{miko@small@dbend}{stex-dangerous-bend}
6631 \def\smalltextwarning{
6632   \pgfuseimage{miko@small@dbend}
6633   \xspace
6634 }
6635 \pgfdeclareimage[width=1.2em]{miko@dbend}{stex-dangerous-bend}
6636 \newrobustcmd\textwarning{
6637   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
6638   \xspace
6639 }
6640 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{stex-dangerous-bend}
6641 \newrobustcmd\bigtextwarning{
6642   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
6643   \xspace
6644 }

```

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

```

6645 \newrobustcmd\putgraphicsat[3]{
6646   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
6647 }
6648 \newrobustcmd\putat[2]{
6649   \begin{picture}(0,0)\put(#1){#2}\end{picture}
6650 }

```

## 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`.

```

6651 \bool_if:NT \c__notesslides_sectocframes_bool {
6652   \str_if_eq:VnTF \__notesslidesstopsect{part}{
6653     \newcounter{chapter}\counterwithin*{section}{chapter}
6654   }{
6655     \str_if_eq:VnT \__notesslidesstopsect{chapter}{
6656       \newcounter{chapter}\counterwithin*{section}{chapter}
6657     }
6658   }
6659 }

```

`\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
6660 \def\part@prefix{}
6661 \@ifpackageloaded{document-structure}{}{
6662   \str_case:VnF \__notesslidesstopsect {
6663     {part}{
6664       \int_set:Nn \l_document_structure_section_level_int {0}
6665       \def\thesection{\arabic{chapter}.\arabic{section}}

```

```

6666     \def\part@prefix{\arabic{chapter}.}
6667   }
6668   {chapter}{
6669     \int_set:Nn \l_document_structure_section_level_int {1}
6670     \def\thesection{\arabic{chapter}.\arabic{section}}
6671     \def\part@prefix{\arabic{chapter}.}
6672   }
6673 }{
6674   \int_set:Nn \l_document_structure_section_level_int {2}
6675   \def\part@prefix{}
6676 }
6677 }
6678
6679 \bool_if:NF \c__notesslides_notes_bool { % only in slides

```

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

The new counters are used in the `omgroup` environment that chooses the L<sup>A</sup>T<sub>E</sub>X sectioning macros according to `\section@level`.

#### sfragment

```

6680 \renewenvironment{sfragment}[2][]{
6681   \_document_structure_omgroup_args:n { #1 }
6682   \int_incr:N \l_document_structure_section_level_int
6683   \bool_if:NT \c__notesslides_sectocframes_bool {
6684     \stepcounter{slide}
6685     \begin{frame}[noframenumbering]
6686     \vfill\Large\centering
6687     \red{
6688       \ifcase\l_document_structure_section_level_int\or
6689         \stepcounter{part}
6690         \def\_notesslideslabel{\omdoc@part@kw~\Roman{part}}
6691         \def\currentsectionlevel{\omdoc@part@kw}
6692       \or
6693         \stepcounter{chapter}
6694         \def\_notesslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
6695         \def\currentsectionlevel{\omdoc@chapter@kw}
6696       \or
6697         \stepcounter{section}
6698         \def\_notesslideslabel{\part@prefix\arabic{section}}
6699         \def\currentsectionlevel{\omdoc@section@kw}
6700       \or
6701         \stepcounter{subsection}
6702         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
6703         \def\currentsectionlevel{\omdoc@subsection@kw}
6704       \or
6705         \stepcounter{subsubsection}
6706         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
6707         \def\currentsectionlevel{\omdoc@subsubsection@kw}
6708       \or
6709         \stepcounter{paragraph}
6710         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
6711         \def\currentsectionlevel{\omdoc@paragraph@kw}
6712       \else
6713         \def\_notesslideslabel{}

```

```

6714         \def\currentsectionlevel{\omdoc@paragraph@kw}
6715         \fi% end ifcase
6716         \_notesslideslabel%\sref@label@id\_notesslideslabel
6717         \quad #2%
6718     }%
6719     \vfill%
6720     \end{frame}%
6721 }
6722 \str_if_empty:NF \l__document_structure_omgroup_id_str {
6723     \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
6724 }
6725 }{}
6726 }

```

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

```

6727 \def\inserttheorembodyfont{\normalfont}
6728 %\bool_if:NF \c__notesslides_notes_bool {
6729 % \defbeamertemplate{theorem begin}{miko}
6730 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
6731 % \ifx\inserttheoremaddition@empty\else\ (\inserttheoremaddition)\fi%
6732 % \inserttheorempunctuation\inserttheorembodyfont\xspace}
6733 % \defbeamertemplate{theorem end}{miko}{}}

```

and we set it as the default one.

```

6734 % \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.

```

6735 % \expandafter\def\csname Parent2\endcsname{}
6736 %}
6737
6738 \AddToHook{begindocument}{% this does not work for some reason
6739     \setbeamertemplate{theorems}[ams style]
6740 }
6741 \bool_if:NT \c__notesslides_notes_bool {
6742     \renewenvironment{columns}[1][{}]{%
6743         \par\noindent%
6744         \begin{minipage}%
6745             \slidewidth\centering\leavevmode%
6746     }{%
6747         \end{minipage}\par\noindent%
6748     }%
6749     \newsavebox\columnbox%
6750     \renewenvironment<>{column}[2][{}]{%
6751         \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
6752     }{%
6753         \end{minipage}\end{lrbox}\usebox\columnbox%
6754     }%
6755 }
6756 \bool_if:NTF \c__notesslides_noproblems_bool {
6757     \newenvironment{problems}{}{}
6758 }{
6759     \excludecomment{problems}
6760 }

```

## 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.

```

6761 \gdef\printexcursions{}
6762 \newcommand\excursionref[2]{% label, text
6763   \bool_if:NT \c__notesslides_notes_bool {
6764     \begin{sparagraph}[title=Excursion]
6765       #2 \sref[fallback=the appendix]{#1}.
6766     \end{sparagraph}
6767   }
6768 }
6769 \newcommand\activate@excursion[2][]{
6770   \gappto\printexcursions{\inputref{#1}{#2}}
6771 }
6772 \newcommand\excursion[4][]{% repos, label, path, text
6773   \bool_if:NT \c__notesslides_notes_bool {
6774     \activate@excursion[#1]{#3}\excursionref{#2}{#4}
6775   }
6776 }

```

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

`\excursiongroup`

```

6777 \keys_define:nn{notesslides / excursiongroup }{
6778   id          .str_set_x:N = \l__notesslides_excursion_id_str,
6779   intro       .tl_set:N   = \l__notesslides_excursion_intro_tl,
6780   mhrepos     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
6781 }
6782 \cs_new_protected:Nn \__notesslides_excursion_args:n {
6783   \tl_clear:N \l__notesslides_excursion_intro_tl
6784   \str_clear:N \l__notesslides_excursion_id_str
6785   \str_clear:N \l__notesslides_excursion_mhrepos_str
6786   \keys_set:nn {notesslides / excursiongroup }{ #1 }
6787 }
6788 \newcommand\excursiongroup[1][]{
6789   \__notesslides_excursion_args:n{ #1 }
6790   \ifdefempty\printexcursions{}% only if there are excursions
6791   {\begin{note}
6792     \begin{sfragment}[#1]{Excursions}%
6793     \ifdefempty\l__notesslides_excursion_intro_tl{\
6794       \inputref[\l__notesslides_excursion_mhrepos_str]{
6795         \l__notesslides_excursion_intro_tl
6796       }
6797     }
6798     \printexcursions%
6799     \end{sfragment}
6800   }
6801 }
6802 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{\fi
6803 \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.

```
6804 <*package>
6805 <@@=problems>
6806 \ProvidesExplPackage{problem}{2022/02/26}{3.0.1}{Semantic Markup for Problems}
6807 \RequirePackage{l3keys2e,stex}
6808
6809 \keys_define:nn { problem / pkg }{
6810   notes      .default:n    = { true },
6811   notes      .bool_set:N   = \c__problems_notes_bool,
6812   gnotes     .default:n    = { true },
6813   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
6814   hints      .default:n    = { true },
6815   hints      .bool_set:N   = \c__problems_hints_bool,
6816   solutions  .default:n    = { true },
6817   solutions  .bool_set:N   = \c__problems_solutions_bool,
6818   pts        .default:n    = { true },
6819   pts        .bool_set:N   = \c__problems_pts_bool,
6820   min        .default:n    = { true },
6821   min        .bool_set:N   = \c__problems_min_bool,
6822   boxed      .default:n    = { true },
6823   boxed      .bool_set:N   = \c__problems_boxed_bool,
6824   unknown    .code:n       = {}
6825 }
6826 \newif\ifsolutions
6827
6828 \ProcessKeysOptions{ problem / pkg }
6829 \bool_if:NTF \c__problems_solutions_bool {
6830   \solutionstrue
6831 }{
6832   \solutionsfalse
6833 }
```

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

```
6834 \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.

```
6835 \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.

```
6836 \def\prob@problem@kw{Problem}
6837 \def\prob@solution@kw{Solution}
6838 \def\prob@hint@kw{Hint}
6839 \def\prob@note@kw{Note}
6840 \def\prob@gnote@kw{Grading}
6841 \def\prob@pt@kw{pt}
6842 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
6843 \AddToHook{begindocument}{
6844   \ltx@ifpackageloaded{babel}{
6845     \makeatletter
6846     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6847     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6848       \input{problem-ngerman.ldf}
6849     }
6850     \clist_if_in:NnT \l_tmpa_clist {finnish}{
6851       \input{problem-finnish.ldf}
6852     }
6853     \clist_if_in:NnT \l_tmpa_clist {french}{
6854       \input{problem-french.ldf}
6855     }
6856     \clist_if_in:NnT \l_tmpa_clist {russian}{
6857       \input{problem-russian.ldf}
6858     }
6859     \makeatother
6860   }{ }
6861 }
```

## 39.2 Problems and Solutions

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

```
6862 \keys_define:nn{ problem / problem }{
6863   id      .str_set:x:N = \l__problems_prob_id_str,
6864   pts     .tl_set:N    = \l__problems_prob_pts_tl,
6865   min     .tl_set:N    = \l__problems_prob_min_tl,
6866   title   .tl_set:N    = \l__problems_prob_title_tl,
6867   type    .tl_set:N    = \l__problems_prob_type_tl,
6868   refnum  .int_set:N   = \l__problems_prob_refnum_int
6869 }
6870 \cs_new_protected:Nn \__problems_prob_args:n {
```

```

6871 \str_clear:N \l__problems_prob_id_str
6872 \tl_clear:N \l__problems_prob_pts_tl
6873 \tl_clear:N \l__problems_prob_min_tl
6874 \tl_clear:N \l__problems_prob_title_tl
6875 \tl_clear:N \l__problems_prob_type_tl
6876 \int_zero_new:N \l__problems_prob_refnum_int
6877 \keys_set:nn { problem / problem }{ #1 }
6878 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
6879   \let\l__problems_prob_refnum_int\undefined
6880 }
6881 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

6882 \newcounter{problem}
6883 \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.

```

6884 \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

```

6885 \newcommand\prob@number{
6886   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
6887     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
6888   }{
6889     \int_if_exist:NTF \l__problems_prob_refnum_int {
6890       \prob@label{\int_use:N \l__problems_prob_refnum_int }
6891     }{
6892       \prob@label\theproblem
6893     }
6894   }
6895 }

```

(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.

```

6896 \newcommand\prob@title[3]{%
6897   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
6898     #2 \l__problems_inclprob_title_tl #3
6899   }{
6900     \tl_if_exist:NTF \l__problems_prob_title_tl {
6901       #2 \l__problems_prob_title_tl #3
6902     }{
6903       #1
6904     }
6905   }
6906 }

```

(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.

```

6907 \def\prob@heading{
6908   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
6909   %\sref@label{id}\prob@problem@kw~\prob@number}{~}
6910 }

```

(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`

```

6911 \newenvironment{sproblem}[1][]{
6912   \__problems_prob_args:n{#1}%\sref@target%
6913   \@in@omtexttrue% we are in a statement (for inline definitions)
6914   \stepcounter{problem}\record@problem
6915   \def\current@section@level{\prob@problem@kw}
6916   \tl_if_exist:NTF \l__problems_inclprob_type_tl {
6917     \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
6918   }{
6919     \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
6920   }
6921   \str_if_exist:NTF \l__problems_inclprob_id_str {
6922     \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
6923   }{
6924     \str_set_eq:NN \sproblemid \l__problems_prob_id_str
6925   }
6926
6927
6928   \clist_set:No \l_tmpa_clist \sproblemtype
6929   \tl_clear:N \l_tmpa_tl
6930   \clist_map_inline:Nn \l_tmpa_clist {
6931     \tl_if_exist:cT {\__problems_sproblem_##1_start:}{
6932       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_start:}}
6933     }
6934   }
6935   \tl_if_empty:NTF \l_tmpa_tl {
6936     \__problems_sproblem_start:
6937   }{
6938     \l_tmpa_tl
6939   }
6940   \stex_ref_new_doc_target:n \sproblemid
6941 }{
6942   \clist_set:No \l_tmpa_clist \sproblemtype
6943   \tl_clear:N \l_tmpa_tl
6944   \clist_map_inline:Nn \l_tmpa_clist {
6945     \tl_if_exist:cT {\__problems_sproblem_##1_end:}{
6946       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_end:}}
6947     }

```



```

6948 }
6949 \tl_if_empty:NTF \l_tmpa_tl {
6950   \__problems_sproblem_end:
6951 }{
6952   \l_tmpa_tl
6953 }
6954
6955
6956 \smallskip
6957 }
6958
6959
6960 \cs_new_protected:Nn \__problems_sproblem_start: {
6961   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
6962 }
6963 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
6964
6965 \newcommand\stexpatchproblem[3][] {
6966   \str_set:Nx \l_tmpa_str{ #1 }
6967   \str_if_empty:NTF \l_tmpa_str {
6968     \tl_set:Nn \__problems_sproblem_start: { #2 }
6969     \tl_set:Nn \__problems_sproblem_end: { #3 }
6970   }{
6971     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
6972     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
6973   }
6974 }
6975
6976
6977 \bool_if:NT \c__problems_boxed_bool {
6978   \surroundwithmdframed{problem}
6979 }

```

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

```

6980 \def\record@problem{
6981   \protected@write\@auxout{}
6982   {
6983     \string\@problem{\prob@number}
6984     {
6985       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
6986         \l__problems_inclprob_pts_tl
6987       }{
6988         \l__problems_prob_pts_tl
6989       }
6990     }%
6991     {
6992       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
6993         \l__problems_inclprob_min_tl
6994       }{
6995         \l__problems_prob_min_tl
6996       }
6997     }
6998   }
6999 }

```

(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).

```
7000 \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.

```
7001 \keys_define:nn { problem / solution }{
7002   id                .str_set_x:N = \l__problems_solution_id_str ,
7003   for               .tl_set:N    = \l__problems_solution_for_tl ,
7004   height            .dim_set:N    = \l__problems_solution_height_dim ,
7005   creators          .clist_set:N = \l__problems_solution_creators_clist ,
7006   contributors      .clist_set:N = \l__problems_solution_contributors_clist ,
7007   srccite           .tl_set:N    = \l__problems_solution_srccite_tl
7008 }
7009 \cs_new_protected:Nn \__problems_solution_args:n {
7010   \str_clear:N \l__problems_solution_id_str
7011   \tl_clear:N \l__problems_solution_for_tl
7012   \tl_clear:N \l__problems_solution_srccite_tl
7013   \clist_clear:N \l__problems_solution_creators_clist
7014   \clist_clear:N \l__problems_solution_contributors_clist
7015   \dim_zero:N \l__problems_solution_height_dim
7016   \keys_set:nn { problem / solution }{ #1 }
7017 }
```

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

```
7018 \newcommand\@startsolution[1][{}]{
7019   \__problems_solution_args:n { #1 }
7020   \@in@omtexttrue% we are in a statement.
7021   \bool_if:NF \c__problems_boxed_bool { \hrule }
7022   \smallskip\noindent
7023   {\textbf\prob@solution@kw : \enspace}
7024   \begin{small}
7025   \def\current@section@level{\prob@solution@kw}
7026   \ignorespacesandpars
7027 }
```

**\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.

```
7028 \newcommand\startsolutions{
7029   \specialcomment{solution}{\@startsolution}{
7030     \bool_if:NF \c__problems_boxed_bool {
7031       \hrule\medskip
7032     }
7033     \end{small}%
7034   }
7035   \bool_if:NT \c__problems_boxed_bool {
7036     \surroundwithmdframed{solution}
7037   }
7038 }
```

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

\stopsolutions

```
7039 \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.

```
7040 \ifsolutions
7041 \startsolutions
7042 \else
7043 \stopsolutions
7044 \fi
```

exnote

```
7045 \bool_if:NTF \c__problems_notes_bool {
7046 \newenvironment{exnote}[1][]{
7047 \par\smallskip\hrule\smallskip
7048 \noindent\textbf{\prob@note@kw : }\small
7049 }{
7050 \smallskip\hrule
7051 }
7052 }{
7053 \excludecomment{exnote}
7054 }
```

hint

```
7055 \bool_if:NTF \c__problems_notes_bool {
7056 \newenvironment{hint}[1][]{
7057 \par\smallskip\hrule\smallskip
7058 \noindent\textbf{\prob@hint@kw :~ }\small
7059 }{
7060 \smallskip\hrule
7061 }
7062 \newenvironment{exhint}[1][]{
7063 \par\smallskip\hrule\smallskip
7064 \noindent\textbf{\prob@hint@kw :~ }\small
7065 }{
7066 \smallskip\hrule
7067 }
7068 }{
7069 \excludecomment{hint}
7070 \excludecomment{exhint}
7071 }
```

gnote

```
7072 \bool_if:NTF \c__problems_notes_bool {
7073 \newenvironment{gnote}[1][]{
7074 \par\smallskip\hrule\smallskip
7075 \noindent\textbf{\prob@gnote@kw : }\small
7076 }{
7077 \smallskip\hrule
7078 }
7079 }{
7080 \excludecomment{gnote}
7081 }
```

### 39.3 Multiple Choice Blocks

EdN:17

mcb 17

```

7082 \newenvironment{mcb}{
7083   \begin{enumerate}
7084 }{
7085   \end{enumerate}
7086 }
```

we define the keys for the mcc macro

```

7087 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
7088   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
7089     \bool_set_true:N #1
7090   }{
7091     \bool_set_false:N #1
7092   }
7093 }
7094 \keys_define:nn { problem / mcc }{
7095   id          .str_set_x:N = \l__problems_mcc_id_str ,
7096   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
7097   T           .default:n    = { true } ,
7098   T           .bool_set:N   = \l__problems_mcc_t_bool ,
7099   F           .default:n    = { true } ,
7100   F           .bool_set:N   = \l__problems_mcc_f_bool ,
7101   Ttext       .code:n       = {
7102     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
7103   } ,
7104   Ftext       .code:n       = {
7105     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
7106   }
7107 }
7108 \cs_new_protected:Nn \l__problems_mcc_args:n {
7109   \str_clear:N \l__problems_mcc_id_str
7110   \tl_clear:N \l__problems_mcc_feedback_tl
7111   \bool_set_true:N \l__problems_mcc_t_bool
7112   \bool_set_true:N \l__problems_mcc_f_bool
7113   \bool_set_true:N \l__problems_mcc_Ttext_bool
7114   \bool_set_false:N \l__problems_mcc_Ftext_bool
7115   \keys_set:nn { problem / mcc }{ #1 }
7116 }
```

\mcc

```

7117 \newcommand\mcc[2][] {
7118   \l__problems_mcc_args:n{ #1 }
7119   \item #2
7120   \ifsolutions
7121     \\\
7122     \bool_if:NT \l__problems_mcc_t_bool {
7123       % TODO!
7124       % \ifcsstring{mcc@T}{T}{\mcc@Ttext}%
7125     }
7126     \bool_if:NT \l__problems_mcc_f_bool {
```

---

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

```

7127      % TODO!
7128      % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
7129    }
7130    \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
7131      !
7132    }{
7133      \l__problems_mcc_feedback_tl
7134    }
7135    \fi
7136  } %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.

```

7137
7138 \keys_define:nn{ problem / inclproblem }{
7139   id      .str_set:N = \l__problems_inclprob_id_str,
7140   pts     .tl_set:N  = \l__problems_inclprob_pts_tl,
7141   min     .tl_set:N  = \l__problems_inclprob_min_tl,
7142   title   .tl_set:N  = \l__problems_inclprob_title_tl,
7143   refnum  .int_set:N  = \l__problems_inclprob_refnum_int,
7144   type    .tl_set:N  = \l__problems_inclprob_type_tl,
7145   mhrepos .str_set:N = \l__problems_inclprob_mhrepos_str
7146 }
7147 \cs_new_protected:Nn \l__problems_inclprob_args:n {
7148   \str_clear:N \l__problems_prob_id_str
7149   \tl_clear:N \l__problems_inclprob_pts_tl
7150   \tl_clear:N \l__problems_inclprob_min_tl
7151   \tl_clear:N \l__problems_inclprob_title_tl
7152   \tl_clear:N \l__problems_inclprob_type_tl
7153   \int_zero_new:N \l__problems_inclprob_refnum_int
7154   \str_clear:N \l__problems_inclprob_mhrepos_str
7155   \keys_set:nn { problem / inclproblem }{ #1 }
7156   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
7157     \let\l__problems_inclprob_pts_tl\undefined
7158   }
7159   \tl_if_empty:NT \l__problems_inclprob_min_tl {
7160     \let\l__problems_inclprob_min_tl\undefined
7161   }
7162   \tl_if_empty:NT \l__problems_inclprob_title_tl {
7163     \let\l__problems_inclprob_title_tl\undefined
7164   }
7165   \tl_if_empty:NT \l__problems_inclprob_type_tl {
7166     \let\l__problems_inclprob_type_tl\undefined
7167   }
7168   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
7169     \let\l__problems_inclprob_refnum_int\undefined
7170   }
7171 }

```

```

7172
7173 \cs_new_protected:Nn \__problems_inclprob_clear: {
7174   \let\l__problems_inclprob_id_str\undefined
7175   \let\l__problems_inclprob_pts_tl\undefined
7176   \let\l__problems_inclprob_min_tl\undefined
7177   \let\l__problems_inclprob_title_tl\undefined
7178   \let\l__problems_inclprob_type_tl\undefined
7179   \let\l__problems_inclprob_refnum_int\undefined
7180   \let\l__problems_inclprob_mhrepos_str\undefined
7181 }
7182 \__problems_inclprob_clear:
7183
7184 \newcommand\includeproblem[2][ ]{
7185   \__problems_inclprob_args:n{ #1 }
7186   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
7187     \input{#2}
7188   }{
7189     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
7190       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
7191     }
7192   }
7193   \__problems_inclprob_clear:
7194 }

```

(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.

```

7195 \AddToHook{enddocument}{
7196   \bool_if:NT \c__problems_pts_bool {
7197     \message{Total:~\arabic{pts}~points}
7198   }
7199   \bool_if:NT \c__problems_min_bool {
7200     \message{Total:~\arabic{min}~minutes}
7201   }
7202 }

```

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

```

7203 \def\pts#1{
7204   \bool_if:NT \c__problems_pts_bool {
7205     \marginpar{#1~\prob@pt@kw}
7206   }
7207 }
7208 \def\min#1{
7209   \bool_if:NT \c__problems_min_bool {
7210     \marginpar{#1~\prob@min@kw}
7211   }
7212 }

```

`\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.

```

7213 \newcounter{pts}
7214 \def\show@pts{
7215   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7216     \bool_if:NT \c__problems_pts_bool {
7217       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
7218       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
7219     }
7220   }{
7221     \tl_if_exist:NT \l__problems_prob_pts_tl {
7222       \bool_if:NT \c__problems_pts_bool {
7223         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
7224         \addtocounter{pts}{\l__problems_prob_pts_tl}
7225       }
7226     }
7227   }
7228 }

```

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

`\show@min`

```

7229 \newcounter{min}
7230 \def\show@min{
7231   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7232     \bool_if:NT \c__problems_min_bool {
7233       \marginpar{\l__problems_inclprob_min_tl\ min}
7234       \addtocounter{min}{\l__problems_inclprob_min_tl}
7235     }
7236   }{
7237     \tl_if_exist:NT \l__problems_prob_min_tl {
7238       \bool_if:NT \c__problems_min_bool {
7239         \marginpar{\l__problems_prob_min_tl\ min}
7240         \addtocounter{min}{\l__problems_prob_min_tl}
7241       }
7242     }
7243   }
7244 }
7245 \</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.

```
7246 \@@=hwexam>
7247 \*cls>
7248 \ProvidesExplClass{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
7249 \RequirePackage{l3keys2e}
7250 \DeclareOption*{
7251   \PassOptionsToClass{\CurrentOption}{document-structure}
7252   \PassOptionsToPackage{\CurrentOption}{stex}
7253   \PassOptionsToPackage{\CurrentOption}{hwexam}
7254   \PassOptionsToPackage{\CurrentOption}{tikzinput}
7255 }
7256 \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.

```
7257 \LoadClass{document-structure}
7258 \RequirePackage{stex}
7259 \RequirePackage{hwexam}
7260 \RequirePackage{tikzinput}
7261 \RequirePackage{graphicx}
7262 \RequirePackage{a4wide}
7263 \RequirePackage{amssymb}
7264 \RequirePackage{amstext}
7265 \RequirePackage{amsmath}
```

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



```

7266 \newcommand\assig@default@type{\hwexam@assignment@kw}
7267 \def\document@hwexamtype{\assig@default@type}
7268 <@@=document_structure>
7269 \keys_define:nn { document-structure / document }{
7270 id .str_set_x:N = \c_document_structure_document_id_str,
7271 hwexamtype .tl_set:N = \document@hwexamtype
7272 }
7273 <@@=hwexam>
7274 </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.

```
7275 \*package>
7276 \ProvidesExplPackage{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
7277 \RequirePackage{13keys2e}
7278
7279 \newif\iftest\testfalse
7280 \DeclareOption{test}{\testtrue}
7281 \newif\ifmultiple\multiplefalse
7282 \DeclareOption{multiple}{\multipletrue}
7283 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
7284 \ProcessOptions
```

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

```
7285 \RequirePackage{keyval}[1997/11/10]
7286 \RequirePackage{problem}
```

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

```
7287 \newcommand\hwexam@assignment@kw{Assignment}
7288 \newcommand\hwexam@given@kw{Given}
7289 \newcommand\hwexam@due@kw{Due}
7290 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
7291 blank~for~extra~space}
7292 \def\hwexam@minutes@kw{minutes}
7293 \newcommand\correction@probs@kw{prob.}
7294 \newcommand\correction@pts@kw{total}
7295 \newcommand\correction@reached@kw{reached}
7296 \newcommand\correction@sum@kw{Sum}
7297 \newcommand\correction@grade@kw{grade}
7298 \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

```

7299 \AddToHook{begindocument}{
7300 \ltx@ifpackageloaded{babel}{
7301 \makeatletter
7302 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7303 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7304 \input{hwexam-ngerman.ldf}
7305 }
7306 \clist_if_in:NnT \l_tmpa_clist {finnish}{
7307 \input{hwexam-finnish.ldf}
7308 }
7309 \clist_if_in:NnT \l_tmpa_clist {french}{
7310 \input{hwexam-french.ldf}
7311 }
7312 \clist_if_in:NnT \l_tmpa_clist {russian}{
7313 \input{hwexam-russian.ldf}
7314 }
7315 \makeatother
7316 }{}
7317 }
7318

```

## 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.

```

7319 \newcounter{assignment}
7320 \numberproblemsin{assignment}
7321 \renewcommand\prob@label[1]{\assignment@number.#1}

```

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

```

7322 \keys_define:nn { hwexam / assignment } {
7323 id .str_set:N = \l__hwexam_assign_id_str,
7324 number .int_set:N = \l__hwexam_assign_number_int,
7325 title .tl_set:N = \l__hwexam_assign_title_tl,
7326 type .tl_set:N = \l__hwexam_assign_type_tl,
7327 given .tl_set:N = \l__hwexam_assign_given_tl,
7328 due .tl_set:N = \l__hwexam_assign_due_tl,
7329 loadmodules .code:n = {
7330 \bool_set_true:N \l__hwexam_assign_loadmodules_bool
7331 }
7332 }
7333 \cs_new_protected:Nn \__hwexam_assignment_args:n {
7334 \str_clear:N \l__hwexam_assign_id_str
7335 \int_set:Nn \l__hwexam_assign_number_int {-1}
7336 \tl_clear:N \l__hwexam_assign_title_tl
7337 \tl_clear:N \l__hwexam_assign_type_tl
7338 \tl_clear:N \l__hwexam_assign_given_tl
7339 \tl_clear:N \l__hwexam_assign_due_tl
7340 \bool_set_false:N \l__hwexam_assign_loadmodules_bool

```

```

7341 \keys_set:nn { hwexam / assignment }{ #1 }
7342 }

```

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.

```

7343 \newcommand\given@due[2]{
7344 \bool_lazy_all:nF {
7345 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
7346 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
7347 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
7348 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
7349 }{ #1 }
7350
7351 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
7352 \tl_if_empty:NF \l__hwexam_assign_given_tl {
7353 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
7354 }
7355 }{
7356 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
7357 }
7358
7359 \bool_lazy_or:nnF {
7360 \bool_lazy_and_p:nn {
7361 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
7362 }{
7363 \tl_if_empty_p:V \l__hwexam_assign_due_tl
7364 }
7365 }{
7366 \bool_lazy_and_p:nn {
7367 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
7368 }{
7369 \tl_if_empty_p:V \l__hwexam_assign_due_tl
7370 }
7371 }{ ,~ }
7372
7373 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
7374 \tl_if_empty:NF \l__hwexam_assign_due_tl {
7375 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
7376 }
7377 }{
7378 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
7379 }
7380
7381 \bool_lazy_all:nF {
7382 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
7383 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
7384 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
7385 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
7386 }{ #2 }
7387 }

```

`\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.

```

7388 \newcommand\assignment@title[3]{
7389 \tl_if_empty:NTF \l__hwexam_inclasssign_title_tl {
7390 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
7391 #1
7392 }{
7393 #2\l__hwexam_assign_title_tl#3
7394 }
7395 }{
7396 #2\l__hwexam_inclasssign_title_tl#3
7397 }
7398 }

```

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

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

```

7399 \newcommand\assignment@number{
7400 \int_compare:nNnTF \l__hwexam_inclasssign_number_int = {-1} {
7401 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
7402 \arabic{assignment}
7403 } {
7404 \int_use:N \l__hwexam_assign_number_int
7405 }
7406 }{
7407 \int_use:N \l__hwexam_inclasssign_number_int
7408 }
7409 }

```

(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.

```

7410 \newenvironment{assignment}[1][ ]{
7411 \__hwexam_assignment_args:n { #1 }
7412 %\sref@target
7413 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
7414 \global\stepcounter{assignment}
7415 }{
7416 \global\setcounter{assignment}{\int_use:N\l__hwexam_assign_number_int}
7417 }
7418 \setcounter{problem}{0}
7419 \def\current@section@level{\document@hwexamtype}
7420 %\sref@label@id{\document@hwexamtype \thesection}
7421 \begin{@assignment}
7422 }{
7423 \end{@assignment}
7424 }

```

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

```

7425 \def\ass@title{
7426 \protect\document@hwexamtype~\arabic{assignment}
7427 \assignment@title{}\{;\}{} -- \given@due{}\}{}
7428 }
7429 \ifmultiple
7430 \newenvironment{@assignment}{
7431 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
7432 \begin{sfragment}[loadmodules]{\ass@title}
7433 }{
7434 \begin{sfragment}{\ass@title}
7435 }
7436 }{
7437 \end{sfragment}
7438 }

```

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

```

7439 \else
7440 \newenvironment{@assignment}{
7441 \begin{center}\bf
7442 \Large@title\strut\
7443 \document@hwexamtype~\arabic{assignment}\assignment@title{}\{;\}{}{\}{}
7444 \large\given@due{--;\}{}\}{}
7445 \end{center}
7446 }{}
7447 \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.

```

7448 \keys_define:nn { hwexam / inclassignment } {
7449 %id .str_set_x:N = \l__hwexam_assign_id_str,
7450 number .int_set:N = \l__hwexam_inclassign_number_int,
7451 title .tl_set:N = \l__hwexam_inclassign_title_tl,
7452 type .tl_set:N = \l__hwexam_inclassign_type_tl,
7453 given .tl_set:N = \l__hwexam_inclassign_given_tl,
7454 due .tl_set:N = \l__hwexam_inclassign_due_tl,
7455 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
7456 }
7457 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
7458 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
7459 \tl_clear:N \l__hwexam_inclassign_title_tl
7460 \tl_clear:N \l__hwexam_inclassign_type_tl
7461 \tl_clear:N \l__hwexam_inclassign_given_tl
7462 \tl_clear:N \l__hwexam_inclassign_due_tl
7463 \str_clear:N \l__hwexam_inclassign_mhrepos_str
7464 \keys_set:nn { hwexam / inclassignment }{ #1 }
7465 }
7466 \__hwexam_inclassignment_args:n {}
7467
7468 \newcommand\inputassignment[2][{}]{

```

```

7469 \_hwexam_inclassnment_args:n { #1 }
7470 \str_if_empty:NTF \l__hwexam_inclassn_mhrepos_str {
7471 \input{#2}
7472 }{
7473 \stex_in_repository:nn{\l__hwexam_inclassn_mhrepos_str}{
7474 \input{\mhp{path}\l__hwexam_inclassn_mhrepos_str}{#2}}
7475 }
7476 }
7477 \_hwexam_inclassnment_args:n {}
7478 }
7479 \newcommand\includeassignment[2][]{
7480 \newpage
7481 \inputassignment[#1]{#2}
7482 }

```

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

## 41.4 Typesetting Exams

\quizheading

```

7483 \ExplSyntaxOff
7484 \newcommand\quizheading[1]{%
7485 \def\@tas{#1}%
7486 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
7487 \ifx\@tas\@empty\else%
7488 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
7489 \fi%
7490 }
7491 \ExplSyntaxOn

```

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

\testheading

```

7492
7493 \def\hwexamheader{\input{hwexam-default.header}}
7494
7495 \def\hwexamminutes{
7496 \tl_if_empty:NTF \testheading@duration {
7497 {\testheading@min}~\hwexam@minutes@kw
7498 }{
7499 \testheading@duration
7500 }
7501 }
7502
7503 \keys_define:nn { hwexam / testheading } {
7504 min .tl_set:N = \testheading@min,
7505 duration .tl_set:N = \testheading@duration,
7506 reqpts .tl_set:N = \testheading@reqpts,
7507 tools .tl_set:N = \testheading@tools
7508 }
7509 \cs_new_protected:Nn \_hwexam_testheading_args:n {
7510 \tl_clear:N \testheading@min
7511 \tl_clear:N \testheading@duration

```

```

7512 \tl_clear:N \testheading@reqpts
7513 \tl_clear:N \testheading@tools
7514 \keys_set:nn { hwexam / testheading }{ #1 }
7515 }
7516 \newenvironment{testheading}[1][]{
7517   \__hwexam_testheading_args:n{ #1 }
7518   \newcount\check@time\check@time=\testheading@min
7519   \advance\check@time by -\theassignment@totalmin
7520   \newif\if@bonuspoints
7521   \tl_if_empty:NTF \testheading@reqpts {
7522     \@bonuspointsfalse
7523   }{
7524     \newcount\bonus@pts
7525     \bonus@pts=\theassignment@totalpts
7526     \advance\bonus@pts by -\testheading@reqpts
7527     \edef\bonus@pts{\the\bonus@pts}
7528     \@bonuspointstrue
7529   }
7530   \edef\check@time{\the\check@time}
7531
7532   \makeatletter\hwexamheader\makeatother
7533 }{
7534   \newpage
7535 }

```

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

\testspace

```

7536 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

7537 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

7538 \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.

```

7539 <@=problems>
7540 \renewcommand\@problem[3]{
7541   \stepcounter{assignment@probs}
7542   \def\__problemspts{#2}
7543   \ifx\__problemspts\@empty\else
7544     \addtocounter{assignment@totalpts}{#2}
7545   \fi
7546   \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
7547   \xdef\correction@probs{\correction@probs & #1}%
7548   \xdef\correction@pts{\correction@pts & #2}
7549   \xdef\correction@reached{\correction@reached &}

```



```

7550 }
7551 <@@=hwexam>

```

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

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

```

7552 \newcounter{assignment@probs}
7553 \newcounter{assignment@totalpts}
7554 \newcounter{assignment@totalmin}
7555 \def\correction@probs{\correction@probs@kw}
7556 \def\correction@pts{\correction@pts@kw}
7557 \def\correction@reached{\correction@reached@kw}
7558 \stepcounter{assignment@probs}
7559 \newcommand\correction@table{
7560 \resizebox{\textwidth}{!}{%
7561 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
7562 &\multicolumn{\theassignment@probs}{c|}|%|
7563 {\footnotesize\correction@forgrading@kw} &\\ \hline
7564 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
7565 \correction@pts & \theassignment@totalpts & \\ \hline
7566 \correction@reached & & \[.7cm]\hline
7567 \end{tabular}}
7568 </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}

```