

# The sTeX3 Package \*

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

2022-03-06

## Abstract

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

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

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

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

---

\*Version 3.0 (last revised 2022-03-06)

# 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	31
3.4.5	The interpretmodule Environment	33
3.5	Primitive Symbols (The sTeX Metatheory)	33
<b>4</b>	<b>Using sTeX Symbols</b>	<b>34</b>
4.1	Using sTeX Symbols in Text Mode	34
4.2	Customizing Highlighting	34
<b>5</b>	<b>sTeX Statements (Definitions, Theorems, Examples, ...)</b>	<b>35</b>
<b>6</b>	<b>Additional Packages</b>	<b>36</b>
6.1	Modular Document Structuring	36
6.2	Slides and Course Notes	36
6.3	Homework, Problems and Exams	36

<b>7</b>	<b>Stuff</b>	<b>37</b>
7.0.1	Semantic Macros and Notations . . . . .	37
	Other Argument Types . . . . .	40
	Precedences . . . . .	41
7.0.2	Archives and Imports . . . . .	42
	Namespaces . . . . .	42
	Paths in Import-Statements . . . . .	42
<b>II</b>	<b>Documentation</b>	<b>43</b>
<b>8</b>	<b>sTeX-Basics</b>	<b>44</b>
8.1	Macros and Environments . . . . .	44
8.1.1	HTML Annotations . . . . .	44
8.1.2	Babel Languages . . . . .	45
8.1.3	Auxiliary Methods . . . . .	45
<b>9</b>	<b>sTeX-MathHub</b>	<b>46</b>
9.1	Macros and Environments . . . . .	46
9.1.1	Files, Paths, URIs . . . . .	46
9.1.2	MathHub Archives . . . . .	47
9.1.3	Using Content in Archives . . . . .	48
<b>10</b>	<b>sTeX-References</b>	<b>49</b>
10.1	Macros and Environments . . . . .	49
10.1.1	Setting Reference Targets . . . . .	49
10.1.2	Using References . . . . .	50
<b>11</b>	<b>sTeX-Modules</b>	<b>51</b>
11.1	Macros and Environments . . . . .	51
11.1.1	The <code>smodule</code> environment . . . . .	53
<b>12</b>	<b>sTeX-Module Inheritance</b>	<b>55</b>
12.1	Macros and Environments . . . . .	55
12.1.1	SMS Mode . . . . .	55
12.1.2	Imports and Inheritance . . . . .	56
<b>13</b>	<b>sTeX-Symbols</b>	<b>58</b>
13.1	Macros and Environments . . . . .	58
<b>14</b>	<b>sTeX-Terms</b>	<b>60</b>
14.1	Macros and Environments . . . . .	60
<b>15</b>	<b>sTeX-Structural Features</b>	<b>62</b>
15.1	Macros and Environments . . . . .	62
15.1.1	Structures . . . . .	62
<b>16</b>	<b>sTeX-Statements</b>	<b>63</b>
16.1	Macros and Environments . . . . .	63

<b>17</b>	<b>STeX-Proofs: Structural Markup for Proofs</b>	<b>64</b>
17.1	Introduction	66
17.2	The User Interface	67
17.2.1	Package Options	67
17.2.2	Proofs and Proof steps	67
17.2.3	Justifications	67
17.2.4	Proof Structure	69
17.2.5	Proof End Markers	69
17.2.6	Configuration of the Presentation	69
17.3	Limitations	70
<b>18</b>	<b>STeX-Metatheory</b>	<b>71</b>
18.1	Symbols	71
<b>III</b>	<b>Extensions</b>	<b>72</b>
<b>19</b>	<b>Tikzinput</b>	<b>73</b>
19.1	Macros and Environments	73
<b>20</b>	<b>document-structure: Semantic Markup for Open Mathematical Documents in L<sup>A</sup>T<sub>E</sub>X</b>	<b>74</b>
20.1	Introduction	74
20.2	The User Interface	75
20.2.1	Package and Class Options	75
20.2.2	Document Structure	75
20.2.3	Ignoring Inputs	77
20.2.4	Structure Sharing	77
20.2.5	Global Variables	77
20.2.6	Colors	78
20.3	Limitations	78
<b>21</b>	<b>NotesSlides – Slides and Course Notes</b>	<b>79</b>
21.1	Introduction	79
21.2	The User Interface	79
21.2.1	Package Options	79
21.2.2	Notes and Slides	80
21.2.3	Header and Footer Lines of the Slides	81
21.2.4	Frame Images	81
21.2.5	Colors and Highlighting	82
21.2.6	Front Matter, Titles, etc.	82
21.2.7	Excursions	82
21.2.8	Miscellaneous	83
21.3	Limitations	83

<b>22</b>	<b>problem.sty: An Infrastructure for formatting Problems</b>	<b>84</b>
22.1	Introduction . . . . .	84
22.2	The User Interface . . . . .	84
22.2.1	Package Options . . . . .	84
22.2.2	Problems and Solutions . . . . .	85
22.2.3	Multiple Choice Blocks . . . . .	86
22.2.4	Including Problems . . . . .	86
22.2.5	Reporting Metadata . . . . .	86
22.3	Limitations . . . . .	86
<b>23</b>	<b>hwexam.sty/cls: An Infrastructure for formatting Assignments and Exams</b>	<b>88</b>
23.1	Introduction . . . . .	89
23.2	The User Interface . . . . .	89
23.2.1	Package and Class Options . . . . .	89
23.2.2	Assignments . . . . .	89
23.2.3	Typesetting Exams . . . . .	89
23.2.4	Including Assignments . . . . .	90
23.3	Limitations . . . . .	90
<b>IV</b>	<b>Implementation</b>	<b>92</b>
<b>24</b>	<b>gTeX-Basics Implementation</b>	<b>93</b>
24.1	The gTeXDocument Class . . . . .	93
24.2	Preliminaries . . . . .	93
24.3	Messages and logging . . . . .	94
24.4	HTML Annotations . . . . .	95
24.5	Babel Languages . . . . .	98
24.6	Auxiliary Methods . . . . .	99
<b>25</b>	<b>gTeX-MathHub Implementation</b>	<b>100</b>
25.1	Generic Path Handling . . . . .	100
25.2	PWD and kpsewhich . . . . .	102
25.3	File Hooks and Tracking . . . . .	103
25.4	MathHub Repositories . . . . .	104
25.5	Using Content in Archives . . . . .	108
<b>26</b>	<b>gTeX-References Implementation</b>	<b>113</b>
26.1	Document URIs and URLs . . . . .	113
26.2	Setting Reference Targets . . . . .	115
26.3	Using References . . . . .	117
<b>27</b>	<b>gTeX-Modules Implementation</b>	<b>120</b>
27.1	The smodule environment . . . . .	124
27.2	Invoking modules . . . . .	129
<b>28</b>	<b>gTeX-Module Inheritance Implementation</b>	<b>131</b>
28.1	SMS Mode . . . . .	131
28.2	Inheritance . . . . .	134

<b>29</b>	<b>STEX-Symbols Implementation</b>	<b>139</b>
29.1	Symbol Declarations . . . . .	139
29.2	Notations . . . . .	146
29.3	Variables . . . . .	155
<b>30</b>	<b>STEX-Terms Implementation</b>	<b>162</b>
30.1	Symbol Invocations . . . . .	162
30.2	Terms . . . . .	169
30.3	Notation Components . . . . .	173
30.4	Variables . . . . .	175
30.5	Sequences . . . . .	177
<b>31</b>	<b>STEX-Structural Features Implementation</b>	<b>178</b>
31.1	Imports with modification . . . . .	179
31.2	The feature environment . . . . .	185
31.3	Structure . . . . .	186
<b>32</b>	<b>STEX-Statements Implementation</b>	<b>195</b>
32.1	Definitions . . . . .	195
32.2	Assertions . . . . .	200
32.3	Examples . . . . .	203
32.4	Logical Paragraphs . . . . .	206
<b>33</b>	<b>The Implementation</b>	<b>211</b>
33.1	Package Options . . . . .	211
33.2	Proofs . . . . .	211
33.3	Justifications . . . . .	222
<b>34</b>	<b>STEX-Others Implementation</b>	<b>224</b>
<b>35</b>	<b>STEX-Metatheory Implementation</b>	<b>225</b>
<b>36</b>	<b>Tikzinput Implementation</b>	<b>228</b>
<b>37</b>	<b>document-structure.sty Implementation</b>	<b>230</b>
37.1	The document-structure Class . . . . .	230
37.2	Class Options . . . . .	230
37.3	Beefing up the document environment . . . . .	231
37.4	Implementation: document-structure Package . . . . .	231
37.5	Package Options . . . . .	231
37.6	Document Structure . . . . .	233
37.7	Front and Backmatter . . . . .	236
37.8	Global Variables . . . . .	238

<b>38 NotesSlides – Implementation</b>	<b>239</b>
38.1 Class and Package Options . . . . .	239
38.2 Notes and Slides . . . . .	241
38.3 Header and Footer Lines . . . . .	245
38.4 Frame Images . . . . .	246
38.5 Colors and Highlighting . . . . .	247
38.6 Sectioning . . . . .	248
38.7 Excursions . . . . .	251
<b>39 The Implementation</b>	<b>252</b>
39.1 Package Options . . . . .	252
39.2 Problems and Solutions . . . . .	253
39.3 Multiple Choice Blocks . . . . .	259
39.4 Including Problems . . . . .	260
39.5 Reporting Metadata . . . . .	261
<b>40 Implementation: The hwexam Class</b>	<b>263</b>
40.1 Class Options . . . . .	263
<b>41 Implementation: The hwexam Package</b>	<b>265</b>
41.1 Package Options . . . . .	265
41.2 Assignments . . . . .	266
41.3 Including Assignments . . . . .	269
41.4 Typesetting Exams . . . . .	270
41.5 Leftovers . . . . .	272

# Part I

## Manual



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



Boxes like this one explain how some  $\text{\texttt{sTeX}}$  concept relates to the 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.

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>
        <mi resource="1" property="stex:arg">n</mi>
        <mo class="rel">=</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">
          <mo class="opening">(</mo>
          <msup resource="...realarith?exponentiation##" property="stex:OMA">
            <mi resource="1" property="stex:arg">2</mi>
            <mi resource="2" property="stex:arg">n</mi>
          </msup>
          <mo class="closing">)</mo>
        </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,  $\text{\texttt{STeX}}$  expressions are converted to OMDoc/MMT terms, which use the syntax of OPENMATH.

## 3.2 $\text{\texttt{STeX}}$ 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,  $\text{\texttt{STeX}}$  uses *archives* that determine the global namespaces for symbols and statements and make it possible for  $\text{\texttt{STeX}}$  to find content referenced via such URIs.

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

1. If the  $\text{\texttt{STeX}}$  package is loaded with the option `mathhub=/path/to/mathhub`, then  $\text{\texttt{STeX}}$  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  $\text{\texttt{STeX}}$ -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,  $\text{\texttt{STeX}}$  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 $\text{\texttt{STeX}}$ Archives

An  $\text{\texttt{STeX}}$  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  $\text{\texttt{STeX}}$  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  $\text{\texttt{STeX}}$  will look for files included via `\libinput`.

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

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

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

### 3.2.3 MANIFEST.MF-Files

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

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

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

- `id`: The name of the archive, including its group (e.g. `smglom/calculus`),
- `source-base` or  
  - `ns`: The namespace from which all symbol and module URIs in this repository are formed, see (TODO),
- `narration-base`: The namespace from which all document URIs in this repository are formed, see (TODO),
- `url-base`: The URL that is formed as a basis for *external references*, see (TODO),
- `dependencies`: All archives that this archive depends on.  $\text{\TeX}$  ignores this field, but MMT can pick up on them to resolve dependencies, e.g. for `lmh install`.

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

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

---

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

---

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

---

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

---

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

---

$\backslash\text{libinput}$	$\backslash\text{libinput}$ {some/file} searches for a file some/file in <ul style="list-style-type: none"><li>• the <code>lib</code>-directory of the current archive, and</li><li>• the <code>lib</code>-directory of a <code>meta-inf</code>-archive in (any of) the archive groups containing the current archive</li></ul> and include all found files in reverse order; e.g. $\backslash\text{libinput}$ {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 `\smoduletitle`, `\smoduletype` and `\smoduleid`. For example:

#### Example 2

Input:

```

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

```

Output:

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

### 3.3.2 Declaring New Symbols and Notations

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

`\symdecl`

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

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

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

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

### Example 3

Input:

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

Output:

Given a `foo`, we can...

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

### Example 4

Input:

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

Output:

`this` is a symbol taking two arguments.

`\notation`

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




#### Example 5

Input:

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

Output:

First:  $a$ ; Second:  $b$

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

`\comp`

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

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

#### Example 6

Input:

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

Output:

First:  $a$ ; Second:  $b$



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

Note that it is required that

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

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

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



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

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

## `\symdef`

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

### Example 7

Input:

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

Output:

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

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



---

`\setnotation`

---

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

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

### Operator Notations

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

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

#### Example 8

Input:

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

Output:

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

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

### 3.3.3 Argument Types

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

## b-Type Arguments

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

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

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

### Example 9

Input:

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

Output:

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

where the variable  $x$  is now *bound* by the `\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:  $\forall a <_S b <_{Sc} c <_{sd} d <_{se} e$

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

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

### Example 13

Input:

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

Output:

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

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

#### Example 14

Input:

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

Output:

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

### 3.3.5 Precedences and Automated Bracketing

Having done `\addition`, the obvious next thing to implement is `\multiplication`. This is 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:  
 $\begin{array}{ll} \text{---M--}\rightarrow & \text{\code{\begin{smodule}[lang=<lang>]{Foo}} generates a theory } \text{some/namespace?Foo} \\ \text{---M--}\rightarrow & \text{that only contains the "formal" part of the module -- i.e. exactly the content} \\ \text{---T--}\rightarrow & \text{that is exported when using } \text{\code{\importmodule}}. \\ \text{---T--}\rightarrow & \text{Additionally, MMT generates a } \textit{language theory} \text{ some/namespace/Foo?<lang> that} \\ & \text{includes } \text{some/namespace?Foo} \text{ and contains all the other document content -- vari-} \\ & \text{able declarations, includes for each } \text{\code{\usemodule}}, \text{ etc.} \end{array}$

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.

### 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=\funtyp{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`.

$\hookrightarrow M$  `\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 $\varM!:=\mathstrut{\varM{universe},\varM{op}!,\varM{unit}}$
5 such that
6 $\varM{op}!: \funtype{\varM{universe},\varM{universe}}{\varM{universe}}$
7 and...
```

Output:

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

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

## example 28

```

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]{\comp{-1}}
12 \end{smodule}

```

--

### Example 29

```

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

```

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

32

### 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 sTeX Metatheory)

TODO: metatheory documentation



## Chapter 4

# Using $\text{\TeX}$ Symbols

### 4.1 Using $\text{\TeX}$ Symbols in Text Mode

### 4.2 Customizing Highlighting

TODO: references documentation

## Chapter 5

# TeX Statements (Definitions, Theorems, Examples, ...)

TODO: statements documentation  
TODO: sproofs documentation

## Chapter 6

# Additional Packages

TODO: tikzinput documentation

### 6.1 Modular Document Structuring

TODO: document-structure documentation

### 6.2 Slides and Course Notes

TODO: notesslides documentation

### 6.3 Homework, Problems and Exams

TODO: problem documentation

TODO: hwexam documentation

# Chapter 7

## Stuff

---

`\sTeX`  
`\stex`

---

Both print this  $\text{\TeX}$  logo.

### 7.0.1 Semantic Macros and Notations

Semantic macros invoke a formally declared symbol.

To declare a symbol (in a module), we use `\symdecl`, which takes as argument the name of the corresponding semantic macro, e.g. `\symdecl{foo}` introduces the macro `\foo`. Additionally, `\symdecl` takes several options, the most important one being its arity. `foo` as declared above yields a *constant* symbol. To introduce an *operator* which takes arguments, we have to specify which arguments it takes.

For example, to introduce binary multiplication, we can do `\symdecl{mult}[args=2]`. We can then supply the semantic macro with arbitrarily many notations, such as `\notation{mult}{#1 #2}`.

#### Example 31

Input:

```
1 \symdecl{mult}[args=2]
2 \notation{mult}{#1 #2}
3 $\mult{a}{b}$
```

Output:

$ab$

Since usually, a freshly introduced symbol also comes with a notation from the start, the `\symdef` command combines `\symdecl` and `\notation`. So instead of the above, we could have also written

$$\text{\code{\symdef{mult}[args=2]{#1 #2}}}$$

Adding more notations like `\notation{mult}[cdot]{#1 \comp{\cdot} #2}` or `\notation{mult}[times]{#1 \comp{\times} #2}` allows us to write  $\mult[cdot]{a}{b}$  and  $\mult[times]{a}{b}$ :

### Example 32

Input:

```
1 \notation{mult}[cdot]{#1 \comp{\cdot} #2}
2 \notation{mult}[times]{#1 \comp{\times} #2}
3 $\mult[cdot]{a}{b}$ and $\mult[times]{a}{b}$
```

Output:

$a \cdot b$  and  $a \times b$

Not using an explicit option with a semantic macro yields the first declared notation, unless changed<sup>2</sup>.

Outside of math mode, or by using the starred variant `\foo*`, allows to provide a custom notation, where notational (or textual) components can be given explicitly in square brackets.

### Example 33

Input:

```
1 $\mult*{\arg{a}\comp{\ast}\arg{b}}$ is the
2 \mult{\comp{product of} \arg{$a$} \comp{and} \arg{$b$}}
```

Output:

$a * b$  is the product of  $a$  and  $b$

In custom mode, prefixing an argument with a star will not print that argument, but still export it to OMDoc:

### Example 34

Input:

```
1 \mult{\comp{Multiplying} \arg*{$\mult{a}{b}$} again by \arg{$b$}} yields...
```

Output:

Multiplying again by  $b$  yields...

The syntax `*[<int>]` allows switching the order of arguments. For example, given a 2-ary semantic macro `\forevery` with exemplary notation `\forall #1. #2`, we can write

<sup>2</sup>EdNOTE: TODO

### Example 35

Input:

```

1 \symdecl{forevery}[args=2]
2 \forevery{\arg[2]{The proposition  $P$ } \comp{holds for every} \arg[1]{ $x$  in  $A$ }}

```

Output:

The proposition  $P$  holds for every  $x \in A$

When using  $*[n]$ , after reading the provided ( $n$ th) argument, the “argument counter” automatically continues where we left off, so the  $*[1]$  in the above example can be omitted.

For a macro with arity  $> 0$ , we can refer to the operator *itself* semantically by suffixing the semantic macro with an exclamation point  $!$  in either text or math mode. For that reason `\notation` (and thus `\symdef`) take an additional optional argument `op=`, which allows to assign a notation for the operator itself. e.g.

### Example 36

Input:

```

1 \symdef{add}[args=2,op={+}]{#1 \comp+ #2}
2 The operator  $\text{\add!}$  adds two elements, as in  $\text{\add} ab$ .

```

Output:

The operator  $+$  adds two elements, as in  $a+b$ .

$*$  is composable with  $!$  for custom notations, as in:

### Example 37

Input:

```

1 \mult!{\comp{Multiplication}} (denoted by  $\text{\mult!}\cdot$ ) is defined by...

```

Output:

Multiplication (denoted by  $\cdot$ ) is defined by...

The macro `\comp` as used everywhere above is responsible for highlighting, linking, and tooltips, and should be wrapped around the notation (or text) components that should be treated accordingly. While it is attractive to just wrap a whole notation, this would also wrap around e.g. the arguments themselves, so instead, the user is tasked with marking the notation components themselves.

The precise behaviour of `\comp` is governed by the macro `\@comp`, which takes two arguments: The tex code of the text (unexpanded) to highlight, and the URI of the current symbol. `\@comp` can be safely redefined to customize the behaviour.

The starred variant `\symdecl*{foo}` does not introduce a semantic macro, but still declares a corresponding symbol. `foo` (like any other symbol, for that matter) can then be accessed via `\STEXsymbol{foo}` or (if `foo` was declared in a module `Foo`) via `\STEXModule{Foo}?{foo}`.

both `\STEXsymbol` and `\STEXModule` take any arbitrary ending segment of a full URI to determine which symbol or module is meant. e.g. `\STEXsymbol{Foo?foo}` is also valid, as are e.g. `\STEXModule{path?Foo}?{foo}` or `\STEXsymbol{path?Foo?foo}`

There’s also a convient shortcut `\symref{?foo}{some text}` for `\STEXsymbol{?foo}![some text]`

## Other Argument Types

So far, we have stated the arity of a semantic macro directly. This works if we only have “normal” (or more precisely: i-type) arguments. To make use of other argument types, instead of providing the arity numerically, we can provide it as a sequence of characters representing the argument types – e.g. instead of writing `args=2`, we can equivalently write `args=ii`, indicating that the macro takes two i-type arguments.

Besides i-type arguments,  $\text{\TeX}$  has two other types, which we will discuss now.

The first are *binding* (b-type) arguments, representing variables that are *bound* by the operator. This is the case for example in the above `\forevery`-macro: The first argument is not actually an argument that the `forevery` “function” is “applied” to; rather, the first argument is a new variable (e.g.  $x$ ) that is *bound* in the subsequent argument. More accurately, the macro should therefore have been implemented thusly:

```
\symdef{forevery}[args=bi]{\forall #1.\; #2}
```

b-type arguments are indistinguishable from i-type arguments within  $\text{\TeX}$ , but are treated very differently in OMDoc and by MMT. More interesting *within*  $\text{\TeX}$  are a-type arguments, which represent (associative) arguments of flexible arity, which are provided as comma-separated lists. This allows e.g. better representing the `\mult`-macro above:

### Example 38

Input:

```
1 \symdef{mult}[args=a]{#1}{##1 \comp\cdot ##2}
2 $\mult{a,b,c,{d^e},f}$
```

Output:

$$a \cdot b \cdot c \cdot d^e \cdot f$$

As the example above shows, notations get a little more complicated for associative arguments. For every a-type argument, the `\notation`-macro takes an additional argument that declares how individual entries in an a-type argument list are aggregated. The first notation argument then describes how the aggregated expression is combined into the full representation.

For a more interesting example, consider a flexary operator for ordered sequences in ordered set, that taking arguments `{a,b,c}` and `\mathbb{R}` prints  $a \leq b \leq c \in \mathbb{R}$ . This operator takes two arguments (an a-type argument and an i-type argument), aggregates the individuals of the associative argument using `\leq`, and combines the result with `\in` and the second argument thusly:

EdN:3  
EdN:4

### Example 39

Input:

```
1 \symdef{numseq}[args=ai]{#1 \comp\in #2}{##1 \comp\leq ##2}
2 $\numseq{a,b,c}{\mathbb R}$
```

Output:

$$a \leq b \leq c \in \mathbb{R}$$

Finally, B-type arguments combine the functionalities of a and b, i.e. they represent flexary binding operator arguments.  
<sup>3</sup> <sup>4</sup>

### Precedences

Every notation has an (upwards) *operator precedence* and for each argument a (downwards) *argument precedence* used for automated bracketing. For example, a notation for a binary operator `\foo` could be declared like this:

```
\notation{foo}[prec=200;500x600]{#1 \comp{+} #2}
```

assigning an operator precedence of 200, an argument precedence of 500 for the first argument, and an argument precedence of 600 for the second argument.

TeX insert brackets thusly: Upon encountering a semantic macro (such as `\foo`), its operator precedence (e.g. 200) is compared to the current downwards precedence (initially `\neginfprec`). If the operator precedence is *larger* than the current downwards precedence, parentheses are inserted around the semantic macro.

Notations for symbols of arity 0 have a default precedence of `\infprec`, i.e. by default, parentheses are never inserted around constants. Notations for symbols with arity > 0 have a default operator precedence of 0. If no argument precedences are explicitly provided, then by default they are equal to the operator precedence.

Consequently, if some operator *A* should bind stronger than some operator *B*, then *A*s operator precedence should be smaller than *B*s argument precedences.

For example:

### Example 40

Input:

```
1 \notation{plus}[prec=100]{#1 \comp{+} #2}
2 \notation{times}[prec=50]{#1 \comp{\cdot} #2}
3 $\plus{a}{\times{b}{c}}$ and $\times{a}{\plus{b}{c}}$
```

Output:

$$a + b \cdot c \text{ and } a \cdot (b + c)$$

<sup>3</sup>EdNOTE: what about e.g.  $\int \int \int f \, dx \, dy \, dz$ ?  
<sup>4</sup>EdNOTE: “decompose” a-type arguments into fixed-arity operators?



## **7.0.2 Archives and Imports**

**Namespaces**

**Paths in Import-Statements**

## Part II

# Documentation

# Chapter 8

## sTeX-Basics

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

### 8.1 Macros and Environments

---

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

---

---

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

---

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

#### 8.1.1 HTML Annotations

---

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

---

---

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

---

---

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

---

---

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

---

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

---

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

---

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

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

### 8.1.2 Babel Languages

---

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

---

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

### 8.1.3 Auxiliary Methods

---

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

---

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

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

---

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

---

## Chapter 9

# STEX-MathHub

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

### 9.1 Macros and Environments

---

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

---

#### 9.1.1 Files, Paths, URIs

---

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

---

---

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

---

---

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

---

---

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

---

---

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

---

---

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

---



---

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

---

### 9.1.2 MathHub Archives

---

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

---

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

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

---

<code>\l_stex_current_repository_prop</code>
--

---

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

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

---

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

---

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

---

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

---



---

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

---

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

### 9.1.3 Using Content in Archives

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

# Chapter 10

## STEX-References

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

### 10.1 Macros and Environments

---

---

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

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

---

---

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

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

---

---

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

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

---

---

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

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

---

---

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

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

#### 10.1.1 Setting Reference Targets

---

---

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

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

---

---

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

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



### 10.1.2 Using References

---

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

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

---

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

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

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

---

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

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

# Chapter 11

## STEX-Modules

This sub package contains code related to Modules

### 11.1 Macros and Environments

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

---

---

`\c_stex_module_<URI>_prop`

A property list with the following fields:

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

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

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

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

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

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

**meta** the metatheory of the module.

---

---

`\c_stex_module_<URI>_code`

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

---

---

`\c_stex_module_<URI>_constants`

The names of all constants declared in the module

---

---

`\c_stex_module_<URI>_constants`

The full URIs of all modules imported in this module

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

---

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

---

Computes the current namespace as follows:

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

The result is stored in `\l_stex_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 {<archive-ID>} {<module-path>}`**

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

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

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

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

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

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

2. Otherwise:

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

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

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

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

---

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

---

stores the result in these four variables.

---

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

---

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

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



# Chapter 13

## STEX-Symbols

Code related to symbol declarations and notations

### 13.1 Macros and Environments

---

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

---

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

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

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

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

# Chapter 14

## STEX-Terms

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

### 14.1 Macros and Environments

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

<hr/> <hr/>	
<code>\infprec</code> <code>\neginfprec</code>	Maximal and minimal notation precedences.
<hr/> <hr/>	
<code>\dobrackets</code>	<code>\dobrackets {⟨body⟩}</code>
	Puts $\langle body \rangle$ in parentheses; scaled if in display mode unscaled otherwise. Uses the current $\text{\SIX}$ brackets (by default ( and )), which can be changed temporarily using <code>\withbrackets</code> .
<hr/> <hr/>	
<code>\withbrackets</code>	<code>\withbrackets ⟨left⟩ ⟨right⟩ {⟨body⟩}</code>
	Temporarily (i.e. within $\langle body \rangle$ ) sets the brackets used by $\text{\SIX}$ for automated bracketing (by default ( and )) to $\langle left \rangle$ and $\langle right \rangle$ . Note that $\langle left \rangle$ and $\langle right \rangle$ need to be allowed after <code>\left</code> and <code>\right</code> in display-mode.
<hr/> <hr/>	
<code>\stex_term_custom:nn</code>	<code>\stex_term_custom:nn{⟨URI⟩}{⟨args⟩}</code>
	Implements custom one-time notation. Invoked by <code>\stex_invoke_symbol:n</code> in text mode, or if followed by <code>*</code> in math mode, or whenever followed by <code>!</code> .
<hr/> <hr/>	
<code>\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>5</sup>

<sup>5</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>6</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>6</sup>EdNOTE: we might want to develop an extension `sproof-babel` in the future.

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

## 17.3 Limitations

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

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

## Chapter 18

# sTeX-Metatheory

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

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

### 18.1 Symbols

**Part III**  
**Extensions**

## Chapter 19

# Tikzinput

### 19.1 Macros and Environments

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



## Chapter 20

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

The `document-structure` package is part of the  $\S\TeX$  collection, a version of  $\TeX$ / $\LaTeX$  that allows to markup  $\TeX$ / $\LaTeX$  documents semantically without leaving the document format, essentially turning  $\TeX$ / $\LaTeX$  into a document format for mathematical knowledge management (MKM).

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

### 20.1 Introduction

$\S\TeX$  is a version of  $\TeX$ / $\LaTeX$  that allows to markup  $\TeX$ / $\LaTeX$  documents semantically without leaving the document format, essentially turning  $\TeX$ / $\LaTeX$  into a document format for mathematical knowledge management (MKM). The package supports direct translation to the OMDOC format [Koh06]

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

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

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-strcture` class accept the following options:

<code>class=<i>&lt;name&gt;</i></code>	load <i>&lt;name&gt;.cls</i> instead of <code>article.cls</code>
<code>topsect=<i>&lt;sect&gt;</i></code>	The top-level sectioning level; the default for <i>&lt;sect&gt;</i> 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>2</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.barderiv,loadmodules]{Introducing $\protect\bar$ Derivation

```

<sup>7</sup>EdNOTE: integrate with latexml's XMRef in the Math mode.  
<sup>2</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>3</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>3</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[⟨URL⟩]{⟨label⟩}`, which expands to the previously stored content. If the `\STRlabel` macro was in a different file, then we can give a URL `⟨URL⟩` that lets L<sup>A</sup>T<sub>E</sub>XML generate the correct reference.

`\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>8</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{⟨vname⟩}{⟨text⟩}` to set the global variable `⟨vname⟩` to `⟨text⟩` and `\useSGvar{⟨vname⟩}` to reference it.

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

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

<sup>8</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>9</sup>

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

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

## 21.2.2 Notes and Slides

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

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

```
\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>9</sup>EDNOTE: leaving out `noproblems` for the moment until we decide what to do with it.

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

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

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

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

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

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

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

### 21.2.3 Header and Footer Lines of the Slides

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

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

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

### 21.2.4 Frame Images

Sometimes, we want to integrate slides as images after all – e.g. because we already have a PowerPoint presentation, to which we want to add  $\text{\TeX}$ notes. In this case we can use `\frameimage[<opt>]{<path>}`, where `<opt>` are the options of `\includegraphics` from the `graphicx` package [CR99] and `<path>` is the file path (extension can be left off like in `\includegraphics`). We have added the `label` key that allows to give a frame label that can be referenced like a regular `beamer` frame.<sup>10</sup>

`\frameimage`  
`\mhframeimage`

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

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

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

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

<sup>10</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>5</sup>. Furthermore, we can specify how long the solution to a given problem is estimated to take and how many points will be awarded for a perfect solution.

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

### 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>5</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 Elefants]
    How many Elefants can you fit into a Volkswagen beetle?
  \begin{hint}
    Think positively, this is simple!
  \end{hint}
  \begin{exnote}
    Justify your answer
  \end{exnote}
  \begin{solution}[for=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 Elefants)
How many Elefants can you fit into a Volkswagen beetle?


---


Hint: Think positively, this is simple!


---


Note:Justify your answer


---


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


---


```

Example 6: The Formatted Problem from Figure 5

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

**exnote**

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

may arise in grading.

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

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

### 22.2.3 Multiple Choice Blocks

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

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

See Figure ?? for an example

### 22.2.4 Including Problems

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

### 22.2.5 Reporting Metadata

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

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

## 22.3 Limitations

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

1. none reported yet

```

\begin{sproblem}[title=Functions]
  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  
`number` — 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` `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.  
`min`  
`reqpts`



### 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)

2022-03-06

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 44.)

## 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 44.)

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 44.)

`\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 44.)

`\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
108 \bool_new:N \_stex_html_do_output_bool
109 \bool_set_true:N \_stex_html_do_output_bool
110
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 44.)

```

\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 44.)

`\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>).

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.

```

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   }
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 45.)

## 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 45.)

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:nxx{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 45.)

`\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 45.)

`\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 45.)

## 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 46.)

`\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 46.)

`\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 46.)

`\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:N $\underline{TF}$` . This function is documented on page 46.)

## 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 46.)

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 46.)

## 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 46.)

`\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 47.)

`\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 47.)

`\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 47.)

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` `kpsewhich` for the MATHHUB system variable.

`\c_stex_mathhub_str`

```

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 47.)

`\_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 47.)

`\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 47.)

`\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 47.)

`\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 [47](#).)

## 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 48.)

`\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 48.)

`\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 48.)

### \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:nxxx{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 48.)

### \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 48.)

`\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 48.)

`\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 48.)*

## 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 49.)*

### 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 49.)*



`\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 49.)

`\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 49.)

`\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 49.)

## 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 49.)

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 sref_sym_#1_label_str\exp_not:N\endcsname
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 49.)

## 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 50.)

## \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 50.)

**\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 50.)

```

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 52.)
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 52.)
```

```

\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 52.)

```

```

\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 52.)

```

```

\stex_add_to_current_module:n
\STEXexport
Only allowed within modules:
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 52.)

```

```

\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
52.)

```

```

\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 52.)

`\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 52.)

`\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   }

```

(End definition for `\stex_do_up_to_module:n`. This function is documented on page 52.)

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

1143

(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.

```

1144 \str_new:N \l_stex_modules_ns_str
1145 \str_new:N \l_stex_modules_subpath_str

```

```

1146 \cs_new_protected:Nn \__stex_modules_compute_namespace:nN {
1147   \str_set:Nx \l_tmpa_str { #1 }
1148   \seq_set_eq:NN \l_tmpa_seq #2
1149   % split off file extension
1150   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1151   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1152   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1153   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1154
1155   \bool_set_true:N \l_tmpa_bool
1156   \bool_while_do:Nn \l_tmpa_bool {
1157     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1158     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1159       {source} { \bool_set_false:N \l_tmpa_bool }
1160     }{}{
1161       \seq_if_empty:NT \l_tmpa_seq {
1162         \bool_set_false:N \l_tmpa_bool
1163       }
1164     }
1165   }
1166
1167   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
1168   \str_if_empty:NTF \l_stex_modules_subpath_str {
1169     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
1170   }{
1171     \str_set:Nx \l_stex_modules_ns_str {
1172       \l_tmpa_str/\l_stex_modules_subpath_str
1173     }
1174   }
1175 }
1176
1177 \cs_new_protected:Nn \stex_modules_current_namespace: {
1178   \str_clear:N \l_stex_modules_subpath_str
1179   \prop_if_exist:NTF \l_stex_current_repository_prop {
1180     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1181     \__stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1182   }{
1183     % split off file extension
1184     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1185     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1186     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1187     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1188     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1189     \str_set:Nx \l_stex_modules_ns_str {
1190       file:\stex_path_to_string:N \l_tmpa_seq
1191     }
1192   }
1193 }

```

(End definition for `\stex_modules_current_namespace:`. This function is documented on page 53.)

## 27.1 The smodule environment

smodule arguments:

```

1194 \keys_define:nn { stex / module } {
1195   title      .tl_set:N      = \smodulename ,
1196   type       .str_set_x:N   = \smoduletype ,
1197   id         .str_set_x:N   = \smoduleid ,
1198   deprecate  .str_set_x:N   = \l_stex_module_deprecate_str ,
1199   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1200   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1201   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1202   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1203   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1204   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1205   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1206 }
1207
1208 \cs_new_protected:Nn \__stex_modules_args:n {
1209   \str_clear:N \smodulename
1210   \str_clear:N \smoduletype
1211   \str_clear:N \smoduleid
1212   \str_clear:N \l_stex_module_ns_str
1213   \str_clear:N \l_stex_module_deprecate_str
1214   \str_clear:N \l_stex_module_lang_str
1215   \str_clear:N \l_stex_module_sig_str
1216   \str_clear:N \l_stex_module_creators_str
1217   \str_clear:N \l_stex_module_contributors_str
1218   \str_clear:N \l_stex_module_meta_str
1219   \str_clear:N \l_stex_module_srccite_str
1220   \keys_set:nn { stex / module } { #1 }
1221 }
1222
1223 % module parameters here? In the body?
1224
```

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

```

1225 \cs_new_protected:Nn \stex_module_setup:nn {
1226   \str_set:Nx \l_stex_module_name_str { #2 }
1227   \__stex_modules_args:n { #1 }
1228
1229   First, we set up the name and namespace of the module.
1230   Are we in a nested module?
1231
1232   \stex_if_in_module:TF {
1233     % Nested module
1234     \prop_get:cnN {c_stex_module_\l_stex_current_module_str _prop}
1235       { ns } \l_stex_module_ns_str
1236     \str_set:Nx \l_stex_module_name_str {
1237       \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
1238       { name } / \l_stex_module_name_str
1239     }
1240   }{
1241     % not nested:
1242     \str_if_empty:NT \l_stex_module_ns_str {
1243       \stex_modules_current_namespace:
1244     }
1245   }

```

```

1240 \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1241 \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1242 / {\l_stex_module_ns_str}
1243 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1244 \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1245 \str_set:Nx \l_stex_module_ns_str {
1246 \stex_path_to_string:N \l_tmpa_seq
1247 }
1248 }
1249 }
1250 }

```

Next, we determine the language of the module:

```

1251 \str_if_empty:NT \l_stex_module_lang_str {
1252 \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1253 \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1254 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1255 \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1256 \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1257 \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1258 inferred~from~file~name}
1259 \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1260 }
1261 }
1262
1263 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
1264 \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1265 \l_tmpa_str {
1266 \ltx@ifpackageloaded{babel}{
1267 \exp_args:Nx \selectlanguage { \l_tmpa_str }
1268 }{}
1269 } {
1270 \msg_error:nxx{stex}{error/unknownlanguage}{\l_tmpa_str}
1271 }
1272 }}

```

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

```

1273 \str_if_empty:NTF \l_stex_module_sig_str {
1274 \exp_args:Nnx \prop_gset_from_keyval:cn {
1275 c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1276 } {
1277 name = \l_stex_module_name_str ,
1278 ns = \l_stex_module_ns_str ,
1279 file = \exp_not:o { \g_stex_currentfile_seq } ,
1280 lang = \l_stex_module_lang_str ,
1281 sig = \l_stex_module_sig_str ,
1282 deprecate = \l_stex_module_deprecate_str ,
1283 meta = \l_stex_module_meta_str
1284 }
1285 \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1286 \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1287 \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1288 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1289 \str_if_empty:NT \l_stex_module_meta_str {
1290   \str_set:Nx \l_stex_module_meta_str {
1291     \c_stex_metatheory_ns_str ? Metatheory
1292   }
1293 }
1294 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1295   \bool_set_true:N \l_stex_in_meta_bool
1296   \exp_args:Nx \stex_add_to_current_module:n {
1297     \bool_set_true:N \l_stex_in_meta_bool
1298     \stex_activate_module:n {\l_stex_module_meta_str}
1299     \bool_set_false:N \l_stex_in_meta_bool
1300   }
1301   \stex_activate_module:n {\l_stex_module_meta_str}
1302   \bool_set_false:N \l_stex_in_meta_bool
1303 }
1304 }{
1305   \str_if_empty:NT \l_stex_module_lang_str {
1306     \msg_error:nnxx{stex}{error/siglanguage}{
1307       \l_stex_module_ns_str?\l_stex_module_name_str
1308     }{\l_stex_module_sig_str}
1309   }
1310
1311   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1312   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1313   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1314   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1315   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1316   \str_set:Nx \l_tmpa_str {
1317     \stex_path_to_string:N \l_tmpa_seq /
1318     \l_tmpa_str . \l_stex_module_sig_str .tex
1319   }
1320   \IfFileExists \l_tmpa_str {
1321     \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1322       \str_clear:N \l_stex_current_module_str
1323       \seq_clear:N \l_stex_all_modules_seq
1324       \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1325     }
1326   }{
1327     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1328   }
1329   \stex_if_smsmode:F {
1330     \stex_activate_module:n {
1331       \l_stex_module_ns_str ? \l_stex_module_name_str
1332     }
1333   }
1334   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1335 }
1336 \str_if_empty:NF \l_stex_module_deprecate_str {
1337   \msg_warning:nnxx{stex}{warning/deprecated}{
1338     Module~\l_stex_current_module_str
1339   }{
1340     \l_stex_module_deprecate_str
1341   }

```

```

1342 }
1343 \seq_put_right:Nx \l_stex_all_modules_seq {
1344   \l_stex_module_ns_str ? \l_stex_module_name_str
1345 }
1346 }

```

(End definition for `\stex_module_setup:nn`. This function is documented on page 53.)

**smodule** The module environment.

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

```

1347 \cs_new_protected:Nn \_stex_modules_begin_module: {
1348   \stex_reactivate_macro:N \STEXexport
1349   \stex_reactivate_macro:N \importmodule
1350   \stex_reactivate_macro:N \symdecl
1351   \stex_reactivate_macro:N \notation
1352   \stex_reactivate_macro:N \symdef
1353
1354   \stex_debug:nn{modules}{
1355     New~module:\\
1356     Namespace:~\l_stex_module_ns_str\\
1357     Name:~\l_stex_module_name_str\\
1358     Language:~\l_stex_module_lang_str\\
1359     Signature:~\l_stex_module_sig_str\\
1360     Metatheory:~\l_stex_module_meta_str\\
1361     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1362   }
1363
1364   \stex_if_smsmode:F{
1365     \begin{stex_annotate_env} {theory} {
1366       \l_stex_module_ns_str ? \l_stex_module_name_str
1367     }
1368
1369     \stex_annotate_invisible:nnn{header}{} {
1370       \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1371       \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1372       \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1373         \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1374       }
1375       \str_if_empty:NF \smoduletype {
1376         \stex_annotate:nnn{type}{\smoduletype}{}
1377       }
1378     }
1379   }
1380   \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1381   % TODO: Inherit metatheory for nested modules?
1382 }
1383 \iffalse \end{stex_annotate_env} \fi %^^A make syntax highlighting work again

```

(End definition for `\_stex_modules_begin_module:.`)

`\_stex_modules_end_module:` implements `\end{module}`

```

1384 \cs_new_protected:Nn \_stex_modules_end_module: {
1385   \stex_debug:nn{modules}{Closing~module~\prop_item:cn {c_stex_module\_l_stex_current_module}
1386 }

```

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

The core environment

```

1387 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1388 \NewDocumentEnvironment { smodule } { 0{} m } {
1389   \stex_module_setup:nn{#1}{#2}
1390   \par
1391   \stex_if_smsmode:F{
1392     \tl_clear:N \l_tmpa_tl
1393     \clist_map_inline:Nn \smoduletype {
1394       \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1395         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1396       }
1397     }
1398     \tl_if_empty:NTF \l_tmpa_tl {
1399       \__stex_modules_smodule_start:
1400     }{
1401       \l_tmpa_tl
1402     }
1403   }
1404   \__stex_modules_begin_module:
1405   \str_if_empty:NF \smoduleid {
1406     \stex_ref_new_doc_target:n \smoduleid
1407   }
1408   \stex_smsmode_do:
1409 } {
1410   \__stex_modules_end_module:
1411   \stex_if_smsmode:F {
1412     \end{stex_annotate_env}
1413     \clist_set:No \l_tmpa_clist \smoduletype
1414     \tl_clear:N \l_tmpa_tl
1415     \clist_map_inline:Nn \l_tmpa_clist {
1416       \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1417         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1418       }
1419     }
1420     \tl_if_empty:NTF \l_tmpa_tl {
1421       \__stex_modules_smodule_end:
1422     }{
1423       \l_tmpa_tl
1424     }
1425   }
1426 }

```

**\stexpatchmodule**

```

1427 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1428 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1429
1430 \newcommand\stexpatchmodule[3] [] {
1431   \str_set:Nx \l_tmpa_str{ #1 }
1432   \str_if_empty:NTF \l_tmpa_str {
1433     \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1434     \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1435   }{

```

```

1436     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1437     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1438   }
1439 }

```

(End definition for `\stexpatchmodule`. This function is documented on page 53.)

## 27.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n
1440 \NewDocumentCommand \STEXModule { m } {
1441   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1442   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1443   \tl_set:Nn \l_tmpa_tl {
1444     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1445   }
1446   \seq_map_inline:Nn \l_stex_all_modules_seq {
1447     \str_set:Nn \l_tmpb_str { ##1 }
1448     \str_if_eq:eeT { \l_tmpa_str } {
1449       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1450     } {
1451       \seq_map_break:n {
1452         \tl_set:Nn \l_tmpa_tl {
1453           \stex_invoke_module:n { ##1 }
1454         }
1455       }
1456     }
1457   }
1458   \l_tmpa_tl
1459 }
1460
1461 \cs_new_protected:Nn \stex_invoke_module:n {
1462   \stex_debug:nn{modules}{Invoking~module~#1}
1463   \peek_charcode_remove:NTF ! {
1464     \__stex_modules_invoke_uri:nN { #1 }
1465   } {
1466     \peek_charcode_remove:NTF ? {
1467       \__stex_modules_invoke_symbol:nn { #1 }
1468     } {
1469       \msg_error:nnx{stex}{error/syntax}{
1470         ?~or~!~expected~after~
1471         \c_backslash_str STEXModule{#1}
1472       }
1473     }
1474   }
1475 }
1476
1477 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1478   \str_set:Nn #2 { #1 }
1479 }
1480
1481 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1482   \stex_invoke_symbol:n{#1?#2}

```



```
1483 }
```

(End definition for `\STEXModule` and `\stex_invoke_module:n`. These functions are documented on page 53.)

`\stex_activate_module:n`

```
1484 \bool_new:N \l_stex_in_meta_bool
1485 \bool_set_false:N \l_stex_in_meta_bool
1486 \cs_new_protected:Nn \stex_activate_module:n {
1487   \stex_debug:nn{modules}{Activating~module~#1}
1488   \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1489     \msg_error:nnn{stex}{error/conflictingmodules}{ #1 }
1490   }
1491   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1492     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1493     \use:c{ c_stex_module_#1_code }
1494   }
1495 }
```

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

```
1496 \</package>
```

## Chapter 28

# STEX -Module Inheritance Implementation

```
1497 <*package>
1498
1499 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1500
```

### 28.1 SMS Mode

```
1501 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1502 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1503 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1504 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1505
1506 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1507   \makeatletter
1508   \makeatother
1509   \ExplSyntaxOn
1510   \ExplSyntaxOff
1511   \rustexBREAK
1512 }
1513
1514 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1515   \symdef
1516   \importmodule
1517   \notation
1518   \symdecl
1519   \STEXexport
1520   \inlineass
1521   \inlinedef
1522   \inlineex
1523   \endinput
1524   \setnotation
```

```

1525 \copynotation
1526 }
1527
1528 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1529   \tl_to_str:n {
1530     smodule,
1531     copymodule,
1532     interpretmodule,
1533     sdefinition,
1534     sexample,
1535     sassertion,
1536     sparagraph
1537   }
1538 }

```

(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 55.)

`\stex_if_smsmode_p:`  
`\stex_if_smsmode:TF`

```

1539 \bool_new:N \g__stex_smsmode_bool
1540 \bool_set_false:N \g__stex_smsmode_bool
1541 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1542   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1543 }

```

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

`\_stex_smsmode_in_smsmode:nn`

```

1544 \cs_new_protected:Nn \_stex_smsmode_in_smsmode:nn {
1545   \vbox_set:Nn \l_tmpa_box {
1546     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1547     \bool_gset_true:N \g__stex_smsmode_bool
1548     #2
1549     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1550   }
1551   \box_clear:N \l_tmpa_box
1552 }

```

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

`\stex_file_in_smsmode:nn`

```

1553 \quark_new:N \q__stex_smsmode_break
1554
1555 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1556   \stex_filestack_push:n{#1}
1557   \_stex_smsmode_in_smsmode:nn{#1} {
1558     #2
1559     \everyeof{\q__stex_smsmode_break\noexpand}
1560     \expandafter\expandafter\expandafter
1561     \stex_smsmode_do:
1562     \csname @ @ input\endcsname "#1"\relax
1563   }
1564   \stex_filestack_pop:
1565 }

```

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

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

```

1566 \cs_new_protected:Npn \stex_smsmode_do: {
1567   \stex_if_smsmode:T {
1568     \__stex_smsmode_do:w
1569   }
1570 }
1571 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1572   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
1573     \expandafter\if\expandafter\relax\noexpand#1
1574     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1575   } \else\expandafter\__stex_smsmode_do:w\fi
1576 }{
1577   \__stex_smsmode_do:w % #1
1578 }
1579 }
1580 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1581   \cs_if_eq:NNTF #1 \q__stex_smsmode_break {
1582     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1583       #1\__stex_smsmode_do:w
1584     }{
1585       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1586         #1
1587       }{
1588         \cs_if_eq:NNTF \begin #1 {
1589           \__stex_smsmode_check_begin:n
1590         }{
1591           \cs_if_eq:NNTF \end #1 {
1592             \__stex_smsmode_check_end:n
1593           }{
1594             \__stex_smsmode_do:w
1595           }
1596         }
1597       }
1598     }
1599   }
1600 }
1601
1602 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1603   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1604     \begin{#1}
1605   }{
1606     \__stex_smsmode_do:w
1607   }
1608 }
1609 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1610   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1611     \end{#1}\__stex_smsmode_do:w
1612   }{
1613     \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1614   }
1615 }

```

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

## 28.2 Inheritance

```

1616 <@@=stex_importmodule>

\stex_import_module_uri:nn

1617 \cs_new_protected:Nn \stex_import_module_uri:nn {
1618   \str_set:Nx \l_stex_import_archive_str { #1 }
1619   \str_set:Nn \l_stex_import_path_str { #2 }
1620
1621   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1622   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1623   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1624
1625   \stex_modules_current_namespace:
1626   \bool_lazy_all:nTF {
1627     {\str_if_empty_p:N \l_stex_import_archive_str}
1628     {\str_if_empty_p:N \l_stex_import_path_str}
1629     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1630   }{
1631     \str_set_eq:NN \l_stex_import_path_str \l_stex_modules_subpath_str
1632     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1633   }{
1634     \str_if_empty:NT \l_stex_import_archive_str {
1635       \prop_if_exist:NT \l_stex_current_repository_prop {
1636         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1637       }
1638     }
1639     \str_if_empty:NTF \l_stex_import_archive_str {
1640       \str_if_empty:NF \l_stex_import_path_str {
1641         \str_set:Nx \l_stex_import_ns_str {
1642           \l_stex_module_ns_str / \l_stex_import_path_str
1643         }
1644       }
1645     }{
1646       \stex_require_repository:n \l_stex_import_archive_str
1647       \prop_get:cnN { c_stex_mathhub \l_stex_import_archive_str _manifest_prop } { ns }
1648       \l_stex_import_ns_str
1649       \str_if_empty:NF \l_stex_import_path_str {
1650         \str_set:Nx \l_stex_import_ns_str {
1651           \l_stex_import_ns_str / \l_stex_import_path_str
1652         }
1653       }
1654     }
1655   }
1656 }
```

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

<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>	1657 \str_new:N \l_stex_import_name_str
<code>\l_stex_import_path_str</code>	1658 \str_new:N \l_stex_import_archive_str
<code>\l_stex_import_ns_str</code>	1659 \str_new:N \l_stex_import_path_str

```
1660 \str_new:N \l_stex_import_ns_str
```

(End definition for `\l_stex_import_name_str` and others. These variables are documented on page 57.)

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

1661 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1662   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1663
1664     % archive
1665     \str_set:Nx \l_tmpa_str { #2 }
1666     \str_if_empty:NTF \l_tmpa_str {
1667       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1668     } {
1669       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1670       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1671       \seq_put_right:Nn \l_tmpa_seq { source }
1672     }
1673
1674     % path
1675     \str_set:Nx \l_tmpb_str { #3 }
1676     \str_if_empty:NTF \l_tmpb_str {
1677       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1678
1679       \ltx@ifpackageloaded{babel} {
1680         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1681           { \language } \l_tmpb_str {
1682           \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1683         }
1684       } {
1685         \str_clear:N \l_tmpb_str
1686       }
1687
1688       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1689       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1690         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1691       }{
1692         \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1693         \IfFileExists{ \l_tmpa_str.tex }{
1694           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1695         }{
1696           % try english as default
1697           \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1698           \IfFileExists{ \l_tmpa_str.en.tex }{
1699             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1700           }{
1701             \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1702           }
1703         }
1704       }
1705
1706     } {
1707       \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1708       \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1709
```

```

1710 \ltx@ifpackageloaded{babel} {
1711   \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1712     { \language } \l_tmpb_str {
1713       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1714     }
1715   } {
1716     \str_clear:N \l_tmpb_str
1717   }
1718
1719   \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1720
1721   \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1722   \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1723     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1724   }{
1725     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1726     \IfFileExists{ \l_tmpa_str/#4.tex }{
1727       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1728     }{
1729       % try english as default
1730       \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1731       \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1732         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1733       }{
1734         \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1735         \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1736           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1737         }{
1738           \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1739           \IfFileExists{ \l_tmpa_str.tex }{
1740             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1741           }{
1742             % try english as default
1743             \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1744             \IfFileExists{ \l_tmpa_str.en.tex }{
1745               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1746             }{
1747               \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1748             }
1749           }
1750         }
1751       }
1752     }
1753   }
1754 }
1755
1756 \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
1757   \seq_clear:N \l_stex_all_modules_seq
1758   \str_clear:N \l_stex_current_module_str
1759   \str_set:Nx \l_tmpb_str { #2 }
1760   \str_if_empty:NF \l_tmpb_str {
1761     \stex_set_current_repository:n { #2 }
1762   }
1763   \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}

```

```

1764     }
1765
1766     \stex_if_module_exists:nF { #1 ? #4 } {
1767       \msg_error:nnx{stex}{error/unknownmodule}{
1768         #1?#4~(in~file~\g__stex_importmodule_file_str)
1769       }
1770     }
1771   }
1772   \stex_activate_module:n { #1 ? #4 }
1773 }

```

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

### `\importmodule`

```

1774 \NewDocumentCommand \importmodule { 0{ } m } {
1775   \stex_import_module_uri:nn { #1 } { #2 }
1776   \stex_debug:nn{modules}{Importing~module:~
1777     \l_stex_import_ns_str ? \l_stex_import_name_str
1778   }
1779   \stex_if_smsmode:F {
1780     \stex_import_require_module:nnnn
1781     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1782     { \l_stex_import_path_str } { \l_stex_import_name_str }
1783     \stex_annotate_invisible:nnn
1784     {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1785   }
1786   \exp_args:Nx \stex_add_to_current_module:n {
1787     \stex_import_require_module:nnnn
1788     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1789     { \l_stex_import_path_str } { \l_stex_import_name_str }
1790   }
1791   \exp_args:Nx \stex_add_import_to_current_module:n {
1792     \l_stex_import_ns_str ? \l_stex_import_name_str
1793   }
1794   \stex_smsmode_do:
1795   \ignorespacesandpars
1796 }
1797 \stex_deactivate_macro:Nn \importmodule {module-environments}

```

(End definition for `\importmodule`. This function is documented on page 56.)

### `\usemodule`

```

1798 \NewDocumentCommand \usemodule { 0{ } m } {
1799   \stex_if_smsmode:F {
1800     \stex_import_module_uri:nn { #1 } { #2 }
1801     \stex_import_require_module:nnnn
1802     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1803     { \l_stex_import_path_str } { \l_stex_import_name_str }
1804     \stex_annotate_invisible:nnn
1805     {usemodule} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1806   }
1807   \stex_smsmode_do:
1808   \ignorespacesandpars
1809 }

```



*(End definition for \usemodule. This function is documented on page 56.)*

1810 `\endpackage`

## Chapter 29

# STEX -Symbols Implementation

```
1811 <*package>
1812
1813 %%%%%%%%%% symbols.dtx %%%%%%%%%%
1814
      Warnings and error messages
1815 \msg_new:nnn{stex}{error/wrongargs}{
1816   args~value~in~symbol~declaration~for~#1~
1817   needs~to~be~i,~a,~b~or~B,~but~#2~given
1818 }
1819 \msg_new:nnn{stex}{error/unknownsymbol}{
1820   No~symbol~#1~found!
1821 }
1822 \msg_new:nnn{stex}{error/seqlength}{
1823   Expected~#1~arguments;~got~#2!
1824 }
```

### 29.1 Symbol Declarations

```
1825 <@@=stex_symdecl>

\stex_all_symbols:n Map over all available symbols
1826 \cs_new_protected:Nn \stex_all_symbols:n {
1827   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
1828   \seq_map_inline:Nn \l_stex_all_modules_seq {
1829     \seq_map_inline:cn{c_stex_module_##1_constants}{
1830       \__stex_symdecl_all_symbols_cs{##1?####1}
1831     }
1832   }
1833 }

(End definition for \stex_all_symbols:n. This function is documented on page 59.)

\STEXsymbol
1834 \NewDocumentCommand \STEXsymbol { m } {
1835   \stex_get_symbol:n { #1 }
```

```

1836 \exp_args:No
1837 \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1838 }

```

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

`symdecl` arguments:

```

1839 \keys_define:nn { stex / symdecl } {
1840   name      .str_set:N = \l_stex_symdecl_name_str ,
1841   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1842   args      .str_set:N = \l_stex_symdecl_args_str ,
1843   type      .tl_set:N  = \l_stex_symdecl_type_tl ,
1844   deprecate .str_set:N = \l_stex_symdecl_deprecate_str ,
1845   align     .str_set:N = \l_stex_symdecl_align_str , % TODO(?)
1846   gfc       .str_set:N = \l_stex_symdecl_gfc_str , % TODO(?)
1847   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
1848   def       .tl_set:N  = \l_stex_symdecl_definiens_tl ,
1849   assoc     .choices:nn =
1850     {bin,binl,binr,pre,conj,pwconj}
1851     {\str_set:Nx \l_stex_symdecl_astype_str {\l_keys_choice_tl}}
1852 }
1853
1854 \bool_new:N \l_stex_symdecl_make_macro_bool
1855
1856 \cs_new_protected:Nn \__stex_symdecl_args:n {
1857   \str_clear:N \l_stex_symdecl_name_str
1858   \str_clear:N \l_stex_symdecl_args_str
1859   \str_clear:N \l_stex_symdecl_deprecate_str
1860   \str_clear:N \l_stex_symdecl_astype_str
1861   \bool_set_false:N \l_stex_symdecl_local_bool
1862   \tl_clear:N \l_stex_symdecl_type_tl
1863   \tl_clear:N \l_stex_symdecl_definiens_tl
1864
1865   \keys_set:nn { stex / symdecl } { #1 }
1866 }

```

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

```

1867
1868 \NewDocumentCommand \symdecl { s m O{} } {
1869   \__stex_symdecl_args:n { #3 }
1870   \IfBooleanTF #1 {
1871     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1872   } {
1873     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1874   }
1875   \stex_symdecl_do:n { #2 }
1876   \stex_smsmode_do:
1877 }
1878
1879 \cs_new_protected:Nn \stex_symdecl_do:nn {
1880   \__stex_symdecl_args:n{#1}
1881   \bool_set_false:N \l_stex_symdecl_make_macro_bool
1882   \stex_symdecl_do:n{#2}
1883 }

```

```

1884
1885 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

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

```

1886 \cs_new_protected:Nn \stex_symdecl_do:n {
1887   \stex_if_in_module:F {
1888     % TODO throw error? some default namespace?
1889   }
1890
1891   \str_if_empty:NT \l_stex_symdecl_name_str {
1892     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1893   }
1894
1895   \prop_if_exist:cT { l_stex_symdecl_
1896     \l_stex_current_module_str ?
1897     \l_stex_symdecl_name_str
1898   }_prop
1899   {
1900     % TODO throw error (beware of circular dependencies)
1901   }
1902
1903   \prop_clear:N \l_tmpa_prop
1904   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
1905   \seq_clear:N \l_tmpa_seq
1906   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1907   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1908
1909   \str_if_empty:NT \l_stex_symdecl_deprecate_str {
1910     \str_if_empty:NF \l_stex_module_deprecate_str {
1911       \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
1912     }
1913   }
1914   \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
1915
1916   \exp_args:No \stex_add_constant_to_current_module:n {
1917     \l_stex_symdecl_name_str
1918   }
1919
1920   % arity/args
1921   \int_zero:N \l_tmpb_int
1922
1923   \bool_set_true:N \l_tmpa_bool
1924   \str_map_inline:Nn \l_stex_symdecl_args_str {
1925     \token_case_meaning:NnF ##1 {
1926       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1927       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1928       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1929       {\tl_to_str:n a} {
1930         \bool_set_false:N \l_tmpa_bool
1931         \int_incr:N \l_tmpb_int
1932       }
1933       {\tl_to_str:n B} {

```

```

1934     \bool_set_false:N \l_tmpa_bool
1935     \int_incr:N \l_tmpb_int
1936   }
1937 }{
1938   \msg_error:nnxx{stex}{error/wrongargs}{
1939     \l_stex_current_module_str ?
1940     \l_stex_symdecl_name_str
1941   }{##1}
1942 }
1943 }
1944 \bool_if:NTF \l_tmpa_bool {
1945   % possibly numeric
1946   \str_if_empty:NTF \l_stex_symdecl_args_str {
1947     \prop_put:Nnn \l_tmpa_prop { args } {}
1948     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1949   }{
1950     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1951     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1952     \str_clear:N \l_tmpa_str
1953     \int_step_inline:nn \l_tmpa_int {
1954       \str_put_right:Nn \l_tmpa_str i
1955     }
1956     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1957   }
1958 } {
1959   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1960   \prop_put:Nnx \l_tmpa_prop { arity }
1961     { \str_count:N \l_stex_symdecl_args_str }
1962 }
1963 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
1964
1965 \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
1966   \prop_put:Nnx \l_tmpa_prop { defined }{ false }
1967 }{
1968   \prop_put:Nnx \l_tmpa_prop { defined }{ true }
1969 }
1970
1971 % semantic macro
1972
1973 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1974   \exp_args:Nx \stex_do_up_to_module:n {
1975     \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1976       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1977     }}
1978   }
1979
1980   \bool_if:NF \l_stex_symdecl_local_bool {
1981     \exp_args:Nx \stex_add_to_current_module:n {
1982       \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1983         \l_stex_current_module_str ? \l_stex_symdecl_name_str
1984       } }
1985     }
1986   }
1987 }

```

```

1988
1989 \stex_debug:nn{symbols}{New~symbol:~
1990   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
1991   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
1992   Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
1993   Definiens:~\exp_not:o { \l_stex_symdecl_definiens_tl }
1994 }
1995
1996 % circular dependencies require this:
1997
1998 \prop_if_exist:cF {
1999   \l_stex_symdecl_
2000   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2001   _prop
2002 } {
2003   \exp_args:Nx \stex_do_up_to_module:n {
2004     \prop_set_from_keyval:cn {
2005       \l_stex_symdecl_
2006       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2007       _prop
2008     } {\prop_to_keyval:N \l_tmpa_prop}
2009     \seq_clear:c {
2010       \l_stex_symdecl_
2011       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2012       _notations
2013     }
2014   }
2015 }
2016
2017
2018
2019 \bool_if:NF \l_stex_symdecl_local_bool {
2020   \exp_args:Nx
2021   \stex_add_to_current_module:n {
2022     \seq_clear:c {
2023       \l_stex_symdecl_
2024       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2025       _notations
2026     }
2027     \prop_set_from_keyval:cn {
2028       \l_stex_symdecl_
2029       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2030       _prop
2031     } {
2032       name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2033       module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2034       type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2035       args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2036       arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2037       assoc     = \prop_item:Nn \l_tmpa_prop { assoc }     ,
2038     }
2039   }
2040 }
2041

```

```

2042 \stex_if_smsmode:F {
2043 % \exp_args:Nx \stex_do_up_to_module:n {
2044 % \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2045 % \l_stex_current_module_str ? \l_stex_symdecl_name_str
2046 % }
2047 % }
2048 \stex_if_do_html:T {
2049 \stex_annotate_invisible:nnn {symdecl} {
2050 \l_stex_current_module_str ? \l_stex_symdecl_name_str
2051 } {
2052 \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}{}}{ $\l_st
2053 \stex_annotate_invisible:nnn{args}{}{
2054 \prop_item:Nn \l_tmpa_prop { args }
2055 }
2056 \stex_annotate_invisible:nnn{macroname}{#1}{}
2057 \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2058 \stex_annotate_invisible:nnn{definiens}{}
2059 { $\l_stex_symdecl_definiens_tl$
2060 }
2061 \str_if_empty:NF \l_stex_symdecl_assoc_type_str {
2062 \stex_annotate_invisible:nnn{assoc_type}{\l_stex_symdecl_assoc_type_str}{}
2063 }
2064 }
2065 }
2066 }
2067 }

```

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

`\stex_get_symbol:n`

```

2068 \str_new:N \l_stex_get_symbol_uri_str
2069
2070 \cs_new_protected:Nn \stex_get_symbol:n {
2071 \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2072 \tl_set:Nn \l_tmpa_tl { #1 }
2073 \__stex_symdecl_get_symbol_from_cs:
2074 }{
2075 % argument is a string
2076 % is it a command name?
2077 \cs_if_exist:cTF { #1 }{
2078 \cs_set_eq:Nc \l_tmpa_tl { #1 }
2079 \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2080 \str_if_empty:NTF \l_tmpa_str {
2081 \exp_args:Nx \cs_if_eq:NNTF {
2082 \tl_head:N \l_tmpa_tl
2083 } \stex_invoke_symbol:n {
2084 \__stex_symdecl_get_symbol_from_cs:
2085 }{
2086 \__stex_symdecl_get_symbol_from_string:n { #1 }
2087 }
2088 } {
2089 \__stex_symdecl_get_symbol_from_string:n { #1 }
2090 }
2091 }{

```

```

2092     % argument is not a command name
2093     \_stex_symdecl_get_symbol_from_string:n { #1 }
2094     % \l_stex_all_symbols_seq
2095   }
2096 }
2097 \str_if_eq:eeF {
2098   \prop_item:cn {
2099     l_stex_symdecl\_l_stex_get_symbol_uri_str _prop
2100   }{ deprecate }
2101 }{}{
2102   \msg_warning:nnxx{stex}{warning/deprecated}{
2103     Symbol~\l_stex_get_symbol_uri_str
2104   }{
2105     \prop_item:cn {l_stex_symdecl\_l_stex_get_symbol_uri_str _prop}{ deprecate }
2106   }
2107 }
2108 }
2109
2110 \cs_new_protected:Nn \_stex_symdecl_get_symbol_from_string:n {
2111   \tl_set:Nn \l_tmpa_tl {
2112     \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2113   }
2114   \str_set:Nn \l_tmpa_str { #1 }
2115   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2116
2117   \stex_all_symbols:n {
2118     \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2119       \seq_map_break:n{ \seq_map_break:n{
2120         \tl_set:Nn \l_tmpa_tl {
2121           \str_set:Nn \l_stex_get_symbol_uri_str { ##1 }
2122         }
2123       }}
2124     }
2125   }
2126
2127   \l_tmpa_tl
2128 }
2129
2130 \cs_new_protected:Nn \_stex_symdecl_get_symbol_from_cs: {
2131   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2132     { \tl_tail:N \l_tmpa_tl }
2133   \tl_if_single:NTF \l_tmpa_tl {
2134     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2135       \exp_after:wN \str_set:Nn \exp_after:wN
2136         \l_stex_get_symbol_uri_str \l_tmpa_tl
2137     }{
2138       % TODO
2139       % tail is not a single group
2140     }
2141   }{
2142     % TODO
2143     % tail is not a single group
2144   }
2145 }

```



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

## 29.2 Notations

2146 `<@@=stex_notation>`

notation arguments:

```
2147 \keys_define:nn { stex / notation } {
2148   lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2149   variant   .tl_set_x:N = \l__stex_notation_variant_str ,
2150   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2151   op        .tl_set:N   = \l__stex_notation_op_tl ,
2152   primary   .bool_set:N = \l__stex_notation_primary_bool ,
2153   primary   .default:n   = {true} ,
2154   unknown   .code:n      = \str_set:Nx
2155               \l__stex_notation_variant_str \l_keys_key_str
2156 }
2157
2158 \cs_new_protected:Nn \_stex_notation_args:n {
2159   \str_clear:N \l__stex_notation_lang_str
2160   \str_clear:N \l__stex_notation_variant_str
2161   \str_clear:N \l__stex_notation_prec_str
2162   \tl_clear:N \l__stex_notation_op_tl
2163   \bool_set_false:N \l__stex_notation_primary_bool
2164
2165   \keys_set:nn { stex / notation } { #1 }
2166 }
```

`\notation`

```
2167 \NewDocumentCommand \notation { s m O{} } {
2168   \_stex_notation_args:n { #3 }
2169   \tl_clear:N \l_stex_symdecl_definiens_tl
2170   \stex_get_symbol:n { #2 }
2171   \tl_set:Nn \l_stex_notation_after_do_tl {
2172     \__stex_notation_final:
2173     \IfBooleanTF#1{
2174       \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2175     }{}
2176     \stex_smsmode_do:\ignorespacesandpars
2177   }
2178   \stex_notation_do:nnnnn
2179   { \prop_item:cn {l_stex_symdecl\_l_stex_get_symbol_uri_str_prop } { args } }
2180   { \prop_item:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str_prop } { arity } }
2181   { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2182   { \l__stex_notation_prec_str }
2183 }
2184 \stex_deactivate_macro:Nn \notation {module~environments}
```

(End definition for `\notation`. This function is documented on page 59.)

`\stex_notation_do:nnnnn`

```
2185 \seq_new:N \l__stex_notation_precedences_seq
2186 \tl_new:N \l__stex_notation_opprec_tl
2187 \int_new:N \l__stex_notation_currarg_int
```

```

2188 \tl_new:N \stex_symbol_after_invokation_tl
2189
2190 \cs_new_protected:Nn \stex_notation_do:nnnnn {
2191   \let\l_stex_current_symbol_str\relax
2192   \seq_clear:N \l__stex_notation_precedences_seq
2193   \tl_clear:N \l__stex_notation_opprec_tl
2194   \str_set:Nx \l__stex_notation_args_str { #1 }
2195   \str_set:Nx \l__stex_notation_arity_str { #2 }
2196   \str_set:Nx \l__stex_notation_suffix_str { #3 }
2197   \str_set:Nx \l__stex_notation_prec_str { #4 }
2198
2199   % precedences
2200   \str_if_empty:NTF \l__stex_notation_prec_str {
2201     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2202       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2203     }{
2204       \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2205     }
2206   } {
2207     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2208       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2209       \int_step_inline:nn { \l__stex_notation_arity_str } {
2210         \exp_args:NNo
2211         \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2212       }
2213     }{
2214       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2215       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2216         \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2217         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2218           \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2219             \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2220           \seq_map_inline:Nn \l_tmpa_seq {
2221             \seq_put_right:Nn \l_tmpb_seq { ##1 }
2222           }
2223         }
2224       }{
2225         \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2226           \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2227         }{
2228           \tl_set:No \l__stex_notation_opprec_tl { 0 }
2229         }
2230       }
2231     }
2232   }
2233
2234   \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2235   \int_step_inline:nn { \l__stex_notation_arity_str } {
2236     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpb_str {
2237       \exp_args:NNo
2238       \seq_put_right:No \l__stex_notation_precedences_seq {
2239         \l__stex_notation_opprec_tl
2240       }
2241     }
  
```

```

2242 }
2243 \tl_clear:N \l_stex_notation_dummyargs_tl
2244
2245 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2246   \exp_args:NNe
2247   \cs_set:Npn \l_stex_notation_macrocode_cs {
2248     \_stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2249     { \l__stex_notation_suffix_str }
2250     { \l__stex_notation_opprec_tl }
2251     { \exp_not:n { #5 } }
2252   }
2253   \l_stex_notation_after_do_tl
2254 }{
2255   \str_if_in:NnTF \l__stex_notation_args_str b {
2256     \exp_args:Nne \use:nn
2257     {
2258       \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2259       \cs_set:Npn \l__stex_notation_arity_str } { {
2260         \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2261         { \l__stex_notation_suffix_str }
2262         { \l__stex_notation_opprec_tl }
2263         { \exp_not:n { #5 } }
2264       }}
2265   }{
2266     \str_if_in:NnTF \l__stex_notation_args_str B {
2267       \exp_args:Nne \use:nn
2268       {
2269         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2270         \cs_set:Npn \l__stex_notation_arity_str } { {
2271           \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2272           { \l__stex_notation_suffix_str }
2273           { \l__stex_notation_opprec_tl }
2274           { \exp_not:n { #5 } }
2275         } }
2276     }{
2277       \exp_args:Nne \use:nn
2278       {
2279         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2280         \cs_set:Npn \l__stex_notation_arity_str } { {
2281           \_stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2282           { \l__stex_notation_suffix_str }
2283           { \l__stex_notation_opprec_tl }
2284           { \exp_not:n { #5 } }
2285         } }
2286     }
2287   }
2288
2289   \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2290   \int_zero:N \l__stex_notation_currarg_int
2291   \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2292   \__stex_notation_arguments:
2293 }
2294 }

```

(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

```

2295 \cs_new_protected:Nn \_stex_notation_arguments: {
2296   \int_incr:N \l__stex_notation_currarg_int
2297   \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2298     \l_stex_notation_after_do_tl
2299   }{
2300     \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2301     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2302     \str_if_eq:NnTF \l_tmpa_str a {
2303       \_stex_notation_argument_assoc:n
2304     }{
2305       \str_if_eq:NnTF \l_tmpa_str B {
2306         \_stex_notation_argument_assoc:n
2307       }{
2308         \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2309         \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2310           { \_stex_term_math_arg:nnn
2311             { \int_use:N \l__stex_notation_currarg_int }
2312             { \l_tmpa_str }
2313             { ####\int_use:N \l__stex_notation_currarg_int }
2314           }
2315         }
2316         \_stex_notation_arguments:
2317       }
2318     }
2319   }
2320 }
```

(End definition for `\_stex_notation_arguments:.`)

`\_stex_notation_argument_assoc:n`

```

2321 \cs_new_protected:Nn \_stex_notation_argument_assoc:n {
2322
2323   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2324     {\l__stex_notation_arity_str}{
2325     #1
2326   }
2327   \int_zero:N \l_tmpa_int
2328   \tl_clear:N \l_tmpa_tl
2329   \str_map_inline:Nn \l__stex_notation_args_str {
2330     \int_incr:N \l_tmpa_int
2331     \tl_put_right:Nx \l_tmpa_tl {
2332       \str_if_eq:nnTF {##1}{a}{ {} }{
2333         \str_if_eq:nnTF {##1}{B}{ {} }{
2334           {\_stex_term_arg:nn{\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa_in
2335         }
2336       }
2337     }
2338   }
2339   \exp_after:wN\exp_after:wN\exp_after:wN \def
2340   \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2341   \exp_after:wN\exp_after:wN\exp_after:wN ##
2342   \exp_after:wN\exp_after:wN\exp_after:wN 1
2343   \exp_after:wN\exp_after:wN\exp_after:wN ##
```

```

2344 \exp_after:wN\exp_after:wN\exp_after:wN 2
2345 \exp_after:wN\exp_after:wN\exp_after:wN {
2346   \exp_after:wN \exp_after:wN \exp_after:wN
2347   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2348     \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2349   }
2350 }
2351
2352 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2353 \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
2354   \stex_term_math_assoc_arg:nnnn
2355   { \int_use:N \l__stex_notation_currarg_int }
2356   { \l_tmpa_str }
2357   { ####\int_use:N \l__stex_notation_currarg_int }
2358   { \l_tmpa_cs {####1} {####2} }
2359 } }
2360 \__stex_notation_arguments:
2361 }

```

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

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

```

2362 \cs_new_protected:Nn \__stex_notation_final: {
2363 % \exp_args:Nne \use:nn
2364 % {
2365 % \cs_generate_from_arg_count:cNnn {
2366 %   stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2367 %   \l__stex_notation_suffix_str
2368 %   _cs
2369 % }
2370 % \cs_set:Npn \l__stex_notation_arity_str } { {
2371 %   \exp_after:wN \exp_after:wN \exp_after:wN
2372 %   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2373 %   { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2374 % } }
2375
2376 % \tl_if_empty:NF \l__stex_notation_op_tl {
2377 %   \cs_set:cpx {
2378 %     stex_op_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2379 %     \l__stex_notation_suffix_str
2380 %     _cs
2381 %   } { \exp_not:N \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2382 % }
2383
2384 \exp_args:Nx \stex_do_up_to_module:n {
2385   \cs_generate_from_arg_count:cNnn {
2386     stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2387     \l__stex_notation_suffix_str
2388     _cs
2389   } \cs_set:Npn { \l__stex_notation_arity_str } {
2390     \exp_after:wN \exp_after:wN \exp_after:wN
2391     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2392     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2393   }

```

```

2394 \tl_if_empty:NF \l__stex_notation_op_tl {
2395   \cs_set:cpn {
2396     stex_op_notation_\l_stex_get_symbol_uri_str \c_hash_str
2397     \l__stex_notation_suffix_str
2398     _cs
2399   } { \exp_not:N \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2400 }
2401 }
2402
2403 \exp_args:Ne
2404 \stex_add_to_current_module:n {
2405   \cs_generate_from_arg_count:cNnn {
2406     stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2407     \l__stex_notation_suffix_str
2408     _cs
2409   } \cs_set:Npn {\l__stex_notation_arity_str} {
2410     \exp_after:wN \exp_after:wN \exp_after:wN
2411     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2412     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2413   }
2414   \tl_if_empty:NF \l__stex_notation_op_tl {
2415     \cs_set:cpn {
2416       stex_op_notation_\l_stex_get_symbol_uri_str \c_hash_str
2417       \l__stex_notation_suffix_str
2418       _cs
2419     } { \exp_not:N \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2420   }
2421 }
2422
2423 \stex_debug:nn{symbols}{
2424   Notation~\l__stex_notation_suffix_str
2425   ~for~\l_stex_get_symbol_uri_str^^J
2426   Operator~precedence:~\l__stex_notation_opprec_tl^^J
2427   Argument~precedences:~
2428   \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2429   Notation: \cs_meaning:c {
2430     stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2431     \l__stex_notation_suffix_str
2432     _cs
2433   }
2434 }
2435
2436 \exp_args:Nx
2437 \stex_do_up_to_module:n {
2438   \seq_put_right:cx {
2439     l_stex_symdecl_ \l_stex_get_symbol_uri_str
2440     _notations
2441   } {
2442     \l__stex_notation_suffix_str
2443   }
2444 }
2445 \exp_args:Ne
2446 \stex_add_to_current_module:n {
2447   \seq_put_right:cn {

```

```

2448     \l_stex_symdecl \l_stex_get_symbol_uri_str
2449     _notations
2450   } { \l__stex_notation_suffix_str }
2451 }
2452
2453 \stex_if_smsmode:F {
2454
2455   % HTML annotations
2456   \stex_if_do_html:T {
2457     \stex_annotate_invisible:nnn { notation }
2458     { \l_stex_get_symbol_uri_str } {
2459       \stex_annotate_invisible:nnn { notationfragment }
2460       { \l__stex_notation_suffix_str }{}
2461       \stex_annotate_invisible:nnn { precedence }
2462       { \l__stex_notation_prec_str }{}
2463
2464       \int_zero:N \l_tmpa_int
2465       \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2466       \tl_clear:N \l_tmpa_tl
2467       \int_step_inline:nn { \l__stex_notation_arity_str }{
2468         \int_incr:N \l_tmpa_int
2469         \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2470         \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_r
2471         \str_if_eq:VnTF \l_tmpb_str a {
2472           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2473             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2474             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2475           } }
2476         }{
2477           \str_if_eq:VnTF \l_tmpb_str B {
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             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2484               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2485             } }
2486           }
2487         }
2488       }
2489       \stex_annotate_invisible:nnn { notationcomp }{}{
2490         \str_set:Nx \l_stex_current_symbol_str {\l_stex_get_symbol_uri_str }
2491         $ \exp_args:Nno \use:nn { \use:c {
2492           stex_notation_ \l_stex_current_symbol_str
2493           \c_hash_str \l__stex_notation_suffix_str _cs
2494         } } { \l_tmpa_tl } $
2495       }
2496     }
2497   }
2498 }
2499 }

```

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

`\setnotation`

```

2500 \keys_define:nn { stex / setnotation } {
2501   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2502   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2503   unknown .code:n = \str_set:Nx
2504     \l__stex_notation_variant_str \l_keys_key_str
2505 }
2506
2507 \cs_new_protected:Nn \stex_setnotation_args:n {
2508   \str_clear:N \l__stex_notation_lang_str
2509   \str_clear:N \l__stex_notation_variant_str
2510   \keys_set:nn { stex / setnotation } { #1 }
2511 }
2512
2513 \cs_new_protected:Nn \stex_setnotation:n {
2514   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1 _notations }
2515     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2516     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2517       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2518     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2519       { \c_hash_str }
2520     \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_#1 _notations }
2521       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2522     \exp_args:Nx \stex_add_to_current_module:n {
2523       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2524         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2525       \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_#1 _notations }
2526         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2527       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2528         { \c_hash_str }
2529     }
2530     \stex_debug:nn {notations}{
2531       Setting~default~notation~
2532       {\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str}~for~
2533       #1 \\
2534       \expandafter\meaning\csname
2535         l_stex_symdecl_#1 _notations\endcsname
2536     }
2537   }{
2538     % todo throw error
2539   }
2540 }
2541
2542 \NewDocumentCommand \setnotation {m m} {
2543   \stex_get_symbol:n { #1 }
2544   \stex_setnotation_args:n { #2 }
2545   \stex_setnotation:n{\l_stex_get_symbol_uri_str}
2546   \stex_smsmode_do:\ignorespacesandpars
2547 }
2548
2549 \cs_new_protected:Nn \stex_copy_notations:nn {
2550   \stex_debug:nn {notations}{
2551     Copying~notations~from~#2~to~#1\\
2552     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}

```



```

2553 }
2554 \tl_clear:N \l_tmpa_tl
2555 \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2556   \tl_put_right:Nn \l_tmpa_tl { {## ##1} }
2557 }
2558 \seq_map_inline:cn {l_stex_symdecl_#2_notations}{
2559   \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2560   \edef \l_tmpa_tl {
2561     \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2562     \exp_after:wN\exp_after:wN\exp_after:wN {
2563       \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2564     }
2565   }
2566   \exp_args:Nx
2567   \stex_do_up_to_module:n {
2568     \seq_put_right:cn{l_stex_symdecl_#1_notations}{##1}
2569     \cs_generate_from_arg_count:cNnn {
2570       stex_notation_ #1 \c_hash_str ##1 _cs
2571     } \cs_set:Npn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }{
2572       \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl}
2573     }
2574   }
2575 }
2576 }
2577
2578 \NewDocumentCommand \copynotation {m m} {
2579   \stex_get_symbol:n { #1 }
2580   \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2581   \stex_get_symbol:n { #2 }
2582   \exp_args:Noo
2583   \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2584   \exp_args:Nx \stex_add_import_to_current_module:n{
2585     \stex_copy_notations:nn {\l_tmpa_str} {\l_stex_get_symbol_uri_str}
2586   }
2587   \stex_smsmode_do:\ignorespacesandpars
2588 }
2589

```

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

**\symdef**

```

2590 \keys_define:nn { stex / symdef } {
2591   name .str_set_x:N = \l_stex_symdecl_name_str ,
2592   local .bool_set:N = \l_stex_symdecl_local_bool ,
2593   args .str_set_x:N = \l_stex_symdecl_args_str ,
2594   type .tl_set:N = \l_stex_symdecl_type_tl ,
2595   def .tl_set:N = \l_stex_symdecl_definiens_tl ,
2596   op .tl_set:N = \l_stex_notation_op_tl ,
2597   lang .str_set_x:N = \l_stex_notation_lang_str ,
2598   variant .str_set_x:N = \l_stex_notation_variant_str ,
2599   prec .str_set_x:N = \l_stex_notation_prec_str ,
2600   assoc .choices:nn =
2601     {bin,binl,binr,pre,conj,pwconj}
2602     {\str_set:Nx \l_stex_symdecl_assoc_type_str {\l_keys_choice_tl}},

```

```

2603   unknown .code:n      = \str_set:Nx
2604       \l__stex_notation_variant_str \l_keys_key_str
2605 }
2606
2607 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2608   \str_clear:N \l_stex_symdecl_name_str
2609   \str_clear:N \l_stex_symdecl_args_str
2610   \str_clear:N \l_stex_symdecl_assoctype_str
2611   \bool_set_false:N \l_stex_symdecl_local_bool
2612   \tl_clear:N \l_stex_symdecl_type_tl
2613   \tl_clear:N \l_stex_symdecl_definiens_tl
2614   \str_clear:N \l__stex_notation_lang_str
2615   \str_clear:N \l__stex_notation_variant_str
2616   \str_clear:N \l__stex_notation_prec_str
2617   \tl_clear:N \l__stex_notation_op_tl
2618
2619   \keys_set:nn { stex / symdef } { #1 }
2620 }
2621
2622 \NewDocumentCommand \symdef { m O{} } {
2623   \__stex_notation_symdef_args:n { #2 }
2624   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2625   \stex_symdecl_do:n { #1 }
2626   \tl_set:Nn \l_stex_notation_after_do_tl {
2627     \__stex_notation_final:
2628     \stex_smsmode_do:\ignorespacesandpars
2629   }
2630   \str_set:Nx \l_stex_get_symbol_uri_str {
2631     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2632   }
2633   \exp_args:Nx \stex_notation_do:nnnnn
2634     { \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str _prop } { args } }
2635     { \prop_item:cn { l_stex_symdecl\l_stex_get_symbol_uri_str _prop } { arity } }
2636     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2637     { \l__stex_notation_prec_str }
2638   }
2639   \stex_deactivate_macro:Nn \symdef {module~environments}

```

(End definition for \symdef. This function is documented on page 59.)

## 29.3 Variables

```

2640 <@@=stex_variables>
2641
2642 \keys_define:nn { stex / vardef } {
2643   name .str_set_x:N = \l__stex_variables_name_str ,
2644   args .str_set_x:N = \l__stex_variables_args_str ,
2645   type .tl_set:N    = \l__stex_variables_type_tl ,
2646   def  .tl_set:N    = \l__stex_variables_def_tl ,
2647   op   .tl_set:N    = \l__stex_variables_op_tl ,
2648   prec .str_set_x:N = \l__stex_variables_prec_str ,
2649   assoc .choices:nn =
2650     {bin,binl,binr,pre,conj,pwconj}
2651     {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},

```

```

2652   bind      .choices:nn      =
2653           {forall,exists}
2654           {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2655   }
2656
2657   \cs_new_protected:Nn \__stex_variables_args:n {
2658     \str_clear:N \l__stex_variables_name_str
2659     \str_clear:N \l__stex_variables_args_str
2660     \str_clear:N \l__stex_variables_prec_str
2661     \str_clear:N \l__stex_variables_assoctype_str
2662     \str_clear:N \l__stex_variables_bind_str
2663     \tl_clear:N \l__stex_variables_type_tl
2664     \tl_clear:N \l__stex_variables_def_tl
2665     \tl_clear:N \l__stex_variables_op_tl
2666
2667     \keys_set:nn { stex / vardef } { #1 }
2668   }
2669
2670   \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{} } {
2671     \__stex_variables_args:n {#2}
2672     \str_if_empty:NT \l__stex_variables_name_str {
2673       \str_set:Nx \l__stex_variables_name_str { #1 }
2674     }
2675     \prop_clear:N \l_tmpa_prop
2676     \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
2677
2678     \int_zero:N \l_tmpb_int
2679     \bool_set_true:N \l_tmpa_bool
2680     \str_map_inline:Nn \l__stex_variables_args_str {
2681       \token_case_meaning:NnF ##1 {
2682         0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2683         {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2684         {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2685         {\tl_to_str:n a} {
2686           \bool_set_false:N \l_tmpa_bool
2687           \int_incr:N \l_tmpb_int
2688         }
2689         {\tl_to_str:n B} {
2690           \bool_set_false:N \l_tmpa_bool
2691           \int_incr:N \l_tmpb_int
2692         }
2693       }{
2694         \msg_error:nnxx{stex}{error/wrongargs}{
2695           variable~\l__stex_variables_name_str
2696         }{##1}
2697       }
2698     }
2699     \bool_if:NTF \l_tmpa_bool {
2700       % possibly numeric
2701       \str_if_empty:NTF \l__stex_variables_args_str {
2702         \prop_put:Nnn \l_tmpa_prop { args } {}
2703         \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2704       }{
2705         \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }

```

```

2706     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2707     \str_clear:N \l_tmpa_str
2708     \int_step_inline:nn \l_tmpa_int {
2709       \str_put_right:Nn \l_tmpa_str i
2710     }
2711     \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
2712     \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2713   }
2714 } {
2715   \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2716   \prop_put:Nnx \l_tmpa_prop { arity }
2717   { \str_count:N \l__stex_variables_args_str }
2718 }
2719 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
2720 \tl_set:cx { #1 } { \stex_invoke_variable:n { \l__stex_variables_name_str } }
2721
2722 \prop_set_eq:cN { l_stex_variable_\l__stex_variables_name_str _prop } \l_tmpa_prop
2723
2724 \tl_if_empty:NF \l__stex_variables_op_tl {
2725   \cs_set:cpx {
2726     stex_var_op_notation_\l__stex_variables_name_str_cs
2727   } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
2728 }
2729
2730 \tl_set:Nn \l_stex_notation_after_do_tl {
2731   \exp_args:Nne \use:nn {
2732     \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str_cs }
2733     \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
2734   } {{
2735     \exp_after:wN \exp_after:wN \exp_after:wN
2736     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2737     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_symbol
2738   }}
2739 \stex_if_do_html:T {
2740   \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
2741     \stex_annotate_invisible:nnn { precedence }
2742     { \l__stex_variables_prec_str }{}
2743     \tl_if_empty:NF \l__stex_variables_type_tl { \stex_annotate_invisible:nnn{type}{ } { $\l
2744     \stex_annotate_invisible:nnn{args}{ } { \l__stex_variables_args_str }
2745     \stex_annotate_invisible:nnn{macroname}{#1}{ }
2746     \tl_if_empty:NF \l__stex_variables_def_tl {
2747       \stex_annotate_invisible:nnn{definiens}{ }
2748       { $\l__stex_variables_def_tl$ }
2749     }
2750     \str_if_empty:NF \l__stex_variables_assoc_type_str {
2751       \stex_annotate_invisible:nnn{assoc_type}{\l__stex_variables_assoc_type_str}{ }
2752     }
2753     \int_zero:N \l_tmpa_int
2754     \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
2755     \tl_clear:N \l_tmpa_tl
2756     \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } } {{
2757       \int_incr:N \l_tmpa_int
2758       \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
2759       \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables

```

```

2760 \str_if_eq:VnTF \l_tmpb_str a {
2761 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2762 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2763 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2764 } }
2765 }{
2766 \str_if_eq:VnTF \l_tmpb_str B {
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 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2773 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2774 } }
2775 }
2776 }
2777 }
2778 \stex_annotate_invisible:nnn { notationcomp }{}{
2779 \str_set:Nx \l_stex_current_symbol_str {var://\l__stex_variables_name_str }
2780 $ \exp_args:Nno \use:nn { \use:c {
2781 stex_var_notation_\l__stex_variables_name_str _cs
2782 } } { \l_tmpa_tl } $
2783 }
2784 }
2785 }\ignorespacesandpars
2786 }
2787
2788 \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
2789 }
2790
2791 \cs_new:Nn \_stex_reset:N {
2792 \tl_if_exist:NTF #1 {
2793 \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
2794 }{
2795 \let \exp_not:N #1 \exp_not:N \undefined
2796 }
2797 }
2798
2799 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
2800 \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
2801 \exp_args:Nnx \use:nn {
2802 % TODO
2803 \stex_annotate_invisible:nnn {vardecls}{\clist_use:Nn\l__stex_variables_names,}{
2804 #2
2805 }
2806 }{
2807 \_stex_reset:N \varnot
2808 \_stex_reset:N \vartype
2809 \_stex_reset:N \vardefi
2810 }
2811 }
2812
2813 \NewDocumentCommand \vardef { s } {

```

```

2814 \IfBooleanTF#1 {
2815   \__stex_variables_do_complex:nn
2816 }{
2817   \__stex_variables_do_simple:nnn
2818 }
2819 }
2820
2821 \NewDocumentCommand \svar { 0{} m }{
2822   \tl_if_empty:nTF {#1}{
2823     \str_set:Nn \l_tmpa_str { #2 }
2824   }{
2825     \str_set:Nn \l_tmpa_str { #1 }
2826   }
2827   \_stex_term_omv:nn {
2828     var://\l_tmpa_str
2829   }{
2830     \exp_args:Nnx \use:nn {
2831       \def\comp{\_varcomp}
2832       \str_set:Nx \l_stex_current_symbol_str { var://\l_tmpa_str }
2833       \comp{ #2 }
2834     }{
2835       \_stex_reset:N \comp
2836       \_stex_reset:N \l_stex_current_symbol_str
2837     }
2838   }
2839 }
2840
2841
2842
2843 \keys_define:nn { stex / varseq } {
2844   name .str_set_x:N = \l__stex_variables_name_str ,
2845   args .int_set:N   = \l__stex_variables_args_int ,
2846   type .tl_set:N    = \l__stex_variables_type_tl ,
2847   mid .tl_set:N     = \l__stex_variables_mid_tl ,
2848   bind .choices:nn =
2849     {forall,exists}
2850     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2851 }
2852
2853 \cs_new_protected:Nn \__stex_variables_seq_args:n {
2854   \str_clear:N \l__stex_variables_name_str
2855   \int_set:Nn \l__stex_variables_args_int 1
2856   \tl_clear:N \l__stex_variables_type_tl
2857   \str_clear:N \l__stex_variables_bind_str
2858
2859   \keys_set:nn { stex / varseq } { #1 }
2860 }
2861
2862 \NewDocumentCommand \varseq {m 0{} m m m}{
2863   \__stex_variables_seq_args:n { #2 }
2864   \str_if_empty:NT \l__stex_variables_name_str {
2865     \str_set:Nx \l__stex_variables_name_str { #1 }
2866   }
2867   \prop_clear:N \l_tmpa_prop

```

```

2868 \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
2869
2870 \seq_set_from_clist:Nn \l_tmpa_seq {#3}
2871 \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
2872   \msg_error:nnxx{stex}{error/seqlength}
2873   {\int_use:N \l__stex_variables_args_int}
2874   {\seq_count:N \l_tmpa_seq}
2875 }
2876 \seq_set_from_clist:Nn \l_tmpb_seq {#4}
2877 \int_compare:nNnF {\seq_count:N \l_tmpb_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_tmpb_seq}
2881 }
2882 \prop_put:Nnn \l_tmpa_prop {starts} {#3}
2883 \prop_put:Nnn \l_tmpa_prop {ends} {#4}
2884
2885 \cs_generate_from_arg_count:cNnn {stex_varseq\l__stex_variables_name_str _cs}
2886   \cs_set:Npn {\int_use:N \l__stex_variables_args_int} { #5 }
2887
2888 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}
2889 \int_step_inline:nn \l__stex_variables_args_int {
2890   \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
2891 }
2892 \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
2893 \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
2894 \tl_if_empty:NF \l__stex_variables_mid_tl {
2895   \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
2896   \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
2897 }
2898 \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}
2899 \int_step_inline:nn \l__stex_variables_args_int {
2900   \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
2901 }
2902 \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
2903 \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
2904
2905
2906 \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl
2907
2908 \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}
2909
2910 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}
2911
2912 \int_step_inline:nn \l__stex_variables_args_int {
2913   \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
2914     \stex_term_math_arg:nnn{##1}{0}{\exp_not:n{####}##1}
2915   }}
2916 }
2917
2918 \tl_set:Nx \l_tmpa_tl {
2919   \stex_term_math_oma:nnnn { varseq://\l__stex_variables_name_str}{\l_tmpa_tl}
2920   \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
2921 }

```

```

2922 }
2923
2924 \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \stex_symbol_after_invokation_tl} }
2925
2926 \exp_args:Nno \use:nn {
2927 \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
2928 \cs_set:Npn {\int_use:N \l__stex_variables_args_int}}{\l_tmpa_tl}
2929
2930 \stex_debug:nn{sequences}{New~Sequence:~
2931 \expandafter\meaning\csname stex_varseq_\l__stex_variables_name_str _cs\endcsname\\~\\
2932 \prop_to_keyval:N \l_tmpa_prop
2933 }
2934
2935 \prop_set_eq:cN {stex_varseq_\l__stex_variables_name_str _prop}\l_tmpa_prop
2936 \ignorespacesandpars
2937 }
2938
2939 </package>

```



## Chapter 30

# STEX -Terms Implementation

```
2940 <*package>
2941
2942 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2943
2944 <@@=stex_terms>
2945
2946   Warnings and error messages
2947 \msg_new:nnn{stex}{error/nonotation}{
2948   Symbol~#1~invoked,~but~has~no~notation~#2!
2949 }
2950 \msg_new:nnn{stex}{error/notationarg}{
2951   Error~in~parsing~notation~#1
2952 }
2953 \msg_new:nnn{stex}{error/noop}{
2954   Symbol~#1~has~no~operator~notation~for~notation~#2
2955 }
2956 \msg_new:nnn{stex}{error/notallowed}{
2957   Symbol~invocation~#1~not~allowed~in~notation~component~of~#2
2958 }
2959
```

### 30.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
2958
2959
2960 \bool_new:N \l_stex_allow_semantic_bool
2961 \bool_set_true:N \l_stex_allow_semantic_bool
2962
2963 \cs_new_protected:Nn \stex_invoke_symbol:n {
2964   \bool_if:NTF \l_stex_allow_semantic_bool {
2965     \str_if_eq:eeF {
2966       \prop_item:cn {
2967         l_stex_symdecl_#1_prop
2968       }{ deprecate }
2969     }
2970   }
2971 }
```

```

2969   }{}{
2970     \msg_warning:nxxx{stex}{warning/deprecated}{
2971       Symbol~#1
2972     }{
2973       \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
2974     }
2975   }
2976   \if_mode_math:
2977     \exp_after:wN \__stex_terms_invoke_math:n
2978   \else:
2979     \exp_after:wN \__stex_terms_invoke_text:n
2980   \fi: { #1 }
2981 }{
2982   \msg_error:nxxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
2983 }
2984 }
2985
2986 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
2987   \peek_charcode_remove:NTF ! {
2988     \__stex_terms_invoke_op_custom:nn {#1}
2989   }{
2990     \__stex_terms_invoke_custom:nn {#1}
2991   }
2992 }
2993
2994 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2995   \peek_charcode_remove:NTF ! {
2996     % operator
2997     \peek_charcode_remove:NTF * {
2998       % custom op
2999       \__stex_terms_invoke_op_custom:nn {#1}
3000     }{
3001       % op notation
3002       \peek_charcode:NTF [ {
3003         \__stex_terms_invoke_op_notation:nw {#1}
3004       }{
3005         \__stex_terms_invoke_op_notation:nw {#1}[]
3006       }
3007     }
3008   }{
3009     \peek_charcode_remove:NTF * {
3010       \__stex_terms_invoke_custom:nn {#1}
3011       % custom
3012     }{
3013       % normal
3014       \peek_charcode:NTF [ {
3015         \__stex_terms_invoke_notation:nw {#1}
3016       }{
3017         \__stex_terms_invoke_notation:nw {#1}[]
3018       }
3019     }
3020   }
3021 }
3022

```

```

3023
3024 \cs_new_protected:Nn \__stex_terms_invoke_op_custom:nn {
3025   \exp_args:Nnx \use:nn {
3026     \def\comp{\_comp}
3027     \str_set:Nn \l_stex_current_symbol_str { #1 }
3028     \bool_set_false:N \l_stex_allow_semantic_bool
3029     \stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
3030       \comp{ #2 }
3031     }
3032   }{
3033     \stex_reset:N \comp
3034     \stex_reset:N \l_stex_current_symbol_str
3035     \bool_set_true:N \l_stex_allow_semantic_bool
3036   }
3037 }
3038
3039 \keys_define:nn { stex / terms } {
3040   lang .tl_set_x:N = \l_stex_notation_lang_str ,
3041   variant .tl_set_x:N = \l_stex_notation_variant_str ,
3042   unknown .code:n = \str_set:Nx
3043     \l_stex_notation_variant_str \l_keys_key_str
3044 }
3045
3046 \cs_new_protected:Nn \__stex_terms_args:n {
3047   \str_clear:N \l_stex_notation_lang_str
3048   \str_clear:N \l_stex_notation_variant_str
3049
3050   \keys_set:nn { stex / terms } { #1 }
3051 }
3052
3053 \cs_new_protected:Nn \stex_find_notation:nn {
3054   \__stex_terms_args:n { #2 }
3055   \seq_if_empty:cTF {
3056     \l_stex_symdecl_#1 _notations
3057   } {
3058     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
3059   } {
3060     \bool_lazy_all:nTF {
3061       {\str_if_empty_p:N \l_stex_notation_variant_str}
3062       {\str_if_empty_p:N \l_stex_notation_lang_str}
3063     }{
3064       \seq_get_left:cN {\l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
3065     }{
3066       \seq_if_in:cxTF {\l_stex_symdecl_#1_notations}{
3067         \l_stex_notation_variant_str \c_hash_str \l_stex_notation_lang_str
3068       }{
3069         \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3070       }{
3071         \msg_error:nnxx{stex}{error/nonotation}{#1}{
3072           ~\l_stex_notation_variant_str \c_hash_str \l_stex_notation_lang_str
3073         }
3074       }
3075     }
3076   }

```

```

3077 }
3078
3079 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3080   \exp_args:Nnx \use:nn {
3081     \def\comp{\_comp}
3082     \str_set:Nn \l_stex_current_symbol_str { #1 }
3083     \stex_find_notation:nn { #1 }{ #2 }
3084     \bool_set_false:N \l_stex_allow_semantic_bool
3085     \cs_if_exist:cTF {
3086       stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3087     }{
3088       \_stex_term_oms:nnn {
3089         #1 \c_hash_str \l_stex_notation_variant_str
3090       }{ #1 }{
3091         \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3092       }
3093     }{
3094       \int_compare:nNnTF {\prop_item:cn {l_stex_symdecl_#1_prop}{arity}} = 0{
3095         \cs_if_exist:cTF {
3096           stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3097         }{
3098           \tl_set:Nx \stex_symbol_after_invokation_tl {
3099             \_stex_reset:N \comp
3100             \_stex_reset:N \stex_symbol_after_invokation_tl
3101             \_stex_reset:N \l_stex_current_symbol_str
3102             \bool_set_true:N \l_stex_allow_semantic_bool
3103           }
3104           \def\comp{\_comp}
3105           \str_set:Nn \l_stex_current_symbol_str { #1 }
3106           \bool_set_false:N \l_stex_allow_semantic_bool
3107           \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3108         }{
3109           \msg_error:nnxx{stex}{error/nonotation}{#1}{
3110             ~\l_stex_notation_variant_str
3111           }
3112         }
3113       }{
3114         \msg_error:nnxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3115       }
3116     }
3117   }{
3118     \_stex_reset:N \comp
3119     \_stex_reset:N \l_stex_current_symbol_str
3120     \bool_set_true:N \l_stex_allow_semantic_bool
3121   }
3122 }
3123
3124 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {
3125   \stex_find_notation:nn { #1 }{ #2 }
3126   \cs_if_exist:cTF {
3127     stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3128   }{
3129     \tl_set:Nx \stex_symbol_after_invokation_tl {
3130       \_stex_reset:N \comp

```

```

3131     \stex_reset:N \stex_symbol_after_invokation_tl
3132     \stex_reset:N \l_stex_current_symbol_str
3133     \bool_set_true:N \l_stex_allow_semantic_bool
3134   }
3135   \def\comp{\_comp}
3136   \str_set:Nn \l_stex_current_symbol_str { #1 }
3137   \bool_set_false:N \l_stex_allow_semantic_bool
3138   \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3139 }{
3140   \msg_error:nnxx{stex}{error/nonotation}{#1}{
3141     ~\l_stex_notation_variant_str
3142   }
3143 }
3144 }
3145
3146 \prop_new:N \l__stex_terms_custom_args_prop
3147
3148 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3149   \exp_args:Nnx \use:nn {
3150     \bool_set_false:N \l_stex_allow_semantic_bool
3151     \def\comp{\_comp}
3152     \str_set:Nn \l_stex_current_symbol_str { #1 }
3153     \prop_clear:N \l__stex_terms_custom_args_prop
3154     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3155     \prop_get:cnN {
3156       l_stex_symdecl_#1 _prop
3157     }{ args } \l_tmpa_str
3158     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3159     \tl_set:Nn \arg { \__stex_terms_arg: }
3160     \str_if_empty:NTF \l_tmpa_str {
3161       \stex_term oms:nnn {#1}{#1}{#2}
3162     }{
3163       \str_if_in:NnTF \l_tmpa_str b {
3164         \stex_term ombind:nnn {#1}{#1}{#2}
3165       }{
3166         \str_if_in:NnTF \l_tmpa_str B {
3167           \stex_term ombind:nnn {#1}{#1}{#2}
3168         }{
3169           \stex_term oma:nnn {#1}{#1}{#2}
3170         }
3171       }
3172     }
3173     % TODO check that all arguments exist
3174   }{
3175     \stex_reset:N \l_stex_current_symbol_str
3176     \stex_reset:N \arg
3177     \stex_reset:N \comp
3178     \stex_reset:N \l__stex_terms_custom_args_prop
3179     \bool_set_true:N \l_stex_allow_semantic_bool
3180   }
3181 }
3182
3183 \NewDocumentCommand \__stex_terms_arg: { s O{} m }{
3184   \tl_if_empty:nTF {#2}{

```

```

3185 \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
3186 \bool_set_true:N \l_tmpa_bool
3187 \bool_do_while:Nn \l_tmpa_bool {
3188   \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int}
3189   \int_incr:N \l_tmpa_int
3190   }{
3191     \bool_set_false:N \l_tmpa_bool
3192   }
3193 }
3194 ){
3195   \int_set:Nn \l_tmpa_int { #2 }
3196   \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
3197     % TODO throw error
3198   }
3199 }
3200 \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
3201 \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3202   % TODO throw error
3203 }
3204 \bool_set_true:N \l_stex_allow_semantic_bool
3205 \IfBooleanTF#1{
3206   \stex_annotate_invisible:n {
3207     \exp_args:No \_stex_term_arg:nn {\l_stex_current_symbol_str}{#3}
3208   }
3209 }{
3210   \exp_args:No \_stex_term_arg:nn {\l_stex_current_symbol_str}{#3}
3211 }
3212 \bool_set_false:N \l_stex_allow_semantic_bool
3213 }
3214
3215
3216 \cs_new_protected:Nn \_stex_term_arg:nn {
3217   \bool_set_true:N \l_stex_allow_semantic_bool
3218   \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3219   \bool_set_false:N \l_stex_allow_semantic_bool
3220 }
3221
3222 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
3223   \exp_args:Nnx \use:nn
3224   { \int_set:Nn \l__stex_terms_downprec { #2 }
3225     \_stex_term_arg:nn { #1 }{ #3 }
3226   }
3227   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3228 }

```

(End definition for `\stex_invoke_symbol:n`. This function is documented on page 60.)

`\_stex_term_math_assoc_arg:nnnn`

```

3229 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
3230   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3231   \tl_set:Nn \l_tmpb_tl {\_stex_term_math_arg:nnn{#1}{#2}}
3232   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 }}{
3233     \expandafter\if\expandafter\relax\noexpand#3
3234     \expandafter\__stex_terms_math_assoc_arg_maybe_sequence:N\expandafter#3

```

```

3235 \else\expandafter\__stex_terms_math_assoc_arg_simple:n\expandafter#3\fi
3236 }{
3237 \__stex_terms_math_assoc_arg_simple:n{#3}
3238 }
3239 }
3240
3241 \cs_new_protected:Nn \__stex_terms_math_assoc_arg_maybe_sequence:N {
3242 \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
3243 \str_if_empty:NTF \l_tmpa_str {
3244 \exp_args:Nx \cs_if_eq:NNTF {
3245 \tl_head:N #1
3246 } \stex_invoke_sequence:n {
3247 \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
3248 \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
3249 \tl_set:Nx \l_tmpa_tl {\prop_item:cn {stex_varseq \l_tmpa_str _prop}{notation}}
3250 \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
3251 \tl_set:Nx \l_tmpa_tl {\exp_not:N \exp_not:n{
3252 \exp_not:n{\exp_args:Nnx \use:nn} {
3253 \exp_not:n {
3254 \def\comp{\_varcomp}
3255 \str_set:Nn \l_stex_current_symbol_str
3256 } {varseq://\l_tmpa_str}
3257 \exp_not:n{ ##1 }
3258 }{
3259 \exp_not:n {
3260 \_stex_reset:N \comp
3261 \_stex_reset:N \l_stex_current_symbol_str
3262 }
3263 }
3264 }}}
3265 \exp_args:Nno \use:nn {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
3266 \seq_reverse:N \l_tmpa_seq
3267 \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
3268 \seq_map_inline:Nn \l_tmpa_seq {
3269 \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3270 \exp_args:Nno
3271 \l_tmpa_cs { ##1 } \l_tmpa_tl
3272 }
3273 }
3274 \tl_set:Nx \l_tmpa_tl {
3275 \_stex_term_omv:nn {varseq://\l_tmpa_str}{
3276 \exp_args:No \exp_not:n \l_tmpa_tl
3277 }
3278 }
3279 \exp_args:No\l_tmpb_tl\l_tmpa_tl
3280 }{
3281 \__stex_terms_math_assoc_arg_simple:n { #1 }
3282 }
3283 } {
3284 \__stex_terms_math_assoc_arg_simple:n { #1 }
3285 }
3286
3287 }
3288

```

```

3289 \cs_new_protected:Nn \__stex_terms_math_assoc_arg_simple:n {
3290   \clist_set:Nn \l_tmpa_clist{ #1 }
3291   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3292     \tl_set:Nn \l_tmpa_tl { #1 }
3293   }{
3294     \clist_reverse:N \l_tmpa_clist
3295     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3296
3297     \clist_map_inline:Nn \l_tmpa_clist {
3298       \exp_args:NNNo \exp_args:NNNo \tl_set:No \l_tmpa_tl {
3299         \exp_args:Nno
3300         \l_tmpa_cs { ##1 } \l_tmpa_tl
3301       }
3302     }
3303   }
3304   \exp_args:No\l_tmpb_tl\l_tmpa_tl
3305 }

```

(End definition for `\stex_term_math_assoc_arg:nnnn`. This function is documented on page 60.)

## 30.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
3306 \tl_const:Nx \infprec {\int_use:N \c_max_int}
3307 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
3308 \int_new:N \l__stex_terms_downprec
3309 \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 61.)

Bracketing:

```

\l__stex_terms_left_bracket_str
\l__stex_terms_right_bracket_str
3310 \tl_set:Nn \l__stex_terms_left_bracket_str (
3311 \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

```

3312 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
3313   \bool_if:NTF \l__stex_terms_brackets_done_bool {
3314     \bool_set_false:N \l__stex_terms_brackets_done_bool
3315     #2
3316   } {
3317     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
3318       \bool_if:NTF \l_stex_inarray_bool { #2 }{
3319         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
3320         \dobrackets { #2 }
3321       }
3322     }{ #2 }
3323   }
3324 }

```



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

### `\dobrackets`

```

3325 \bool_new:N \l__stex_terms_brackets_done_bool
3326 %\RequirePackage{scalerel}
3327 \cs_new_protected:Npn \dobrackets #1 {
3328   %\ThisStyle{\if D\m@switch
3329   %   \exp_args:Nnx \use:nn
3330   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
3331   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
3332   % \else
3333   \exp_args:Nnx \use:nn
3334   {
3335     \bool_set_true:N \l__stex_terms_brackets_done_bool
3336     \int_set:Nn \l__stex_terms_downprec \infpref
3337     \l__stex_terms_left_bracket_str
3338     #1
3339   }
3340   {
3341     \bool_set_false:N \l__stex_terms_brackets_done_bool
3342     \l__stex_terms_right_bracket_str
3343     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3344   }
3345   %\fi}
3346 }
```

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

### `\withbrackets`

```

3347 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
3348   \exp_args:Nnx \use:nn
3349   {
3350     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
3351     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
3352     #3
3353   }
3354   {
3355     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
3356     {\l__stex_terms_left_bracket_str}
3357     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
3358     {\l__stex_terms_right_bracket_str}
3359   }
3360 }
```

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

### `\STEXinvisible`

```

3361 \cs_new_protected:Npn \STEXinvisible #1 {
3362   \stex_annotate_invisible:n { #1 }
3363 }
```

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

OMDoc terms:

`\_stex_term_math_oms:nnnn`

```
3364 \cs_new_protected:Nn \_stex_term_oms:nnn {
3365   \stex_annotate:nnn{ OMID }{ #2 }{
3366     \stex_highlight_term:nn { #1 } { #3 }
3367   }
3368 }
3369
3370 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
3371   \__stex_terms_maybe_brackets:nn { #3 }{
3372     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3373   }
3374 }
```

*(End definition for \\_stex\_term\_math\_oms:nnnn. This function is documented on page 60.)*

`\_stex_term_math_omv:nn`

```
3375 \cs_new_protected:Nn \_stex_term_omv:nn {
3376   \stex_annotate:nnn{ OMV }{ #1 }{
3377     \stex_highlight_term:nn { #1 } { #2 }
3378   }
3379 }
```

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

`\_stex_term_math_oma:nnnn`

```
3380 \cs_new_protected:Nn \_stex_term_oma:nnn {
3381   \stex_annotate:nnn{ OMA }{ #2 }{
3382     \stex_highlight_term:nn { #1 } { #3 }
3383   }
3384 }
3385
3386 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
3387   \__stex_terms_maybe_brackets:nn { #3 }{
3388     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3389   }
3390 }
```

*(End definition for \\_stex\_term\_math\_oma:nnnn. This function is documented on page 60.)*

`\_stex_term_math_omb:nnnn`

```
3391 \cs_new_protected:Nn \_stex_term_ombind:nnn {
3392   \stex_annotate:nnn{ OMBIND }{ #2 }{
3393     \stex_highlight_term:nn { #1 } { #3 }
3394   }
3395 }
3396
3397 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
3398   \__stex_terms_maybe_brackets:nn { #3 }{
3399     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3400   }
3401 }
```

*(End definition for \\_stex\_term\_math\_omb:nnnn. This function is documented on page 60.)*

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

```

3402 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3403
3404 \keys_define:nn { stex / symname } {
3405   pre      .tl_set_x:N = \l__stex_terms_pre_tl ,
3406   post     .tl_set_x:N = \l__stex_terms_post_tl ,
3407   root     .tl_set_x:N = \l__stex_terms_root_tl
3408 }
3409
3410 \cs_new_protected:Nn \stex_symname_args:n {
3411   \tl_clear:N \l__stex_terms_post_tl
3412   \tl_clear:N \l__stex_terms_pre_tl
3413   \tl_clear:N \l__stex_terms_root_str
3414   \keys_set:nn { stex / symname } { #1 }
3415 }
3416
3417 \NewDocumentCommand \symref { m m }{
3418   \let\compemph_uri_prev:\compemph@uri
3419   \let\compemph@uri\symrefemph@uri
3420   \STEXsymbol{#1}!\{ #2 }
3421   \let\compemph@uri\compemph_uri_prev:
3422 }
3423
3424 \NewDocumentCommand \synonym { 0{} m m }{
3425   \stex_symname_args:n { #1 }
3426   \let\compemph_uri_prev:\compemph@uri
3427   \let\compemph@uri\symrefemph@uri
3428   % TODO
3429   \STEXsymbol{#2}!\{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
3430   \let\compemph@uri\compemph_uri_prev:
3431 }
3432
3433 \NewDocumentCommand \symname { 0{} m }{
3434   \stex_symname_args:n { #1 }
3435   \stex_get_symbol:n { #2 }
3436   \str_set:Nx \l_tmpa_str {
3437     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3438   }
3439   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3440
3441   \let\compemph_uri_prev:\compemph@uri
3442   \let\compemph@uri\symrefemph@uri
3443   \exp_args:NNx \use:nn
3444   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\{
3445     \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
3446   } }
3447   \let\compemph@uri\compemph_uri_prev:
3448 }
3449
3450 \NewDocumentCommand \Symname { 0{} m }{
3451   \stex_symname_args:n { #1 }
3452   \stex_get_symbol:n { #2 }
3453   \str_set:Nx \l_tmpa_str {
3454     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }

```

```

3455 }
3456 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {-}
3457 \let\compemph_uri_prev:\compemph@uri
3458 \let\compemph@uri\symrefemph@uri
3459 \exp_args:NNx \use:nn
3460 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!{
3461   \exp_after:wN \stex_capitalize:n \l_tmpa_str
3462   \l__stex_terms_post_tl
3463 } }
3464 \let\compemph@uri\compemph_uri_prev:
3465 }

```

(End definition for `\symref` and `\symname`. These functions are documented on page 60.)

### 30.3 Notation Components

```

3466 <@@=stex_notationcomps>

```

`\stex_highlight_term:nn`

```

3467 \cs_new_protected:Nn \stex_highlight_term:nn {
3468   #2
3469 }
3470
3471 \cs_new_protected:Nn \stex_unhighlight_term:n {
3472   % \latexml_if:TF {
3473   %   #1
3474   % } {
3475   %   \rustex_if:TF {
3476   %     #1
3477   %   } {
3478   %     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
3479   %   }
3480   % }
3481 }

```

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

```

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

3482 \cs_new_protected:Npn \_comp #1 {
3483   \str_if_empty:NF \l_stex_current_symbol_str {
3484     \rustex_if:TF {
3485       \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
3486     }{
3487       \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3488     }
3489   }
3490 }
3491
3492 \cs_new_protected:Npn \_varcomp #1 {
3493   \str_if_empty:NF \l_stex_current_symbol_str {
3494     \rustex_if:TF {
3495       \stex_annotate:nnn { varcomp }{ \l_stex_current_symbol_str }{ #1 }
3496     }{
3497       \exp_args:Nnx \varemp@uri { #1 } { \l_stex_current_symbol_str }

```

```

3498     }
3499   }
3500 }
3501
3502 \def\comp{\_comp}
3503
3504 \cs_new_protected:Npn \compemph@uri #1 #2 {
3505   \compemph{ #1 }
3506 }
3507
3508
3509 \cs_new_protected:Npn \compemph #1 {
3510   #1
3511 }
3512
3513 \cs_new_protected:Npn \defemph@uri #1 #2 {
3514   \defemph{#1}
3515 }
3516
3517 \cs_new_protected:Npn \defemph #1 {
3518   \textbf{#1}
3519 }
3520
3521 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3522   \symrefemph{#1}
3523 }
3524
3525 \cs_new_protected:Npn \symrefemph #1 {
3526   \textbf{#1}
3527 }
3528
3529 \cs_new_protected:Npn \varempemph@uri #1 #2 {
3530   \varempemph{#1}
3531 }
3532
3533 \cs_new_protected:Npn \varempemph #1 {
3534   #1
3535 }

```

(End definition for `\comp` and others. These functions are documented on page 61.)

## **\ellipses**

```

3536 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix 3537 \bool_new:N \l_stex_inarray_bool
\parrayline 3538 \bool_set_false:N \l_stex_inarray_bool
\parraylineh 3539 \NewDocumentCommand \parray { m m } {
\parraycell 3540   \begingroup
3541   \bool_set_true:N \l_stex_inarray_bool
3542   \begin{array}{#1}
3543     #2
3544   \end{array}

```

```

3545 \endgroup
3546 }
3547
3548 \NewDocumentCommand \prmatrix { m } {
3549   \begingroup
3550   \bool_set_true:N \l_stex_inarray_bool
3551   \begin{matrix}
3552     #1
3553   \end{matrix}
3554   \endgroup
3555 }
3556
3557 \def \maybepline {
3558   \bool_if:NT \l_stex_inarray_bool {\hline}
3559 }
3560
3561 \def \parrayline #1 #2 {
3562   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3563 }
3564
3565 \def \pmrow #1 { \parrayline{}{ #1 } }
3566
3567 \def \parraylineh #1 #2 {
3568   #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3569 }
3570
3571 \def \parraycell #1 {
3572   #1 \bool_if:NT \l_stex_inarray_bool {&}
3573 }

```

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

## 30.4 Variables

```

3574 <@@=stex_variables>

```

\stex\_invoke\_variable:n Invokes a variable

```

3575 \cs_new_protected:Nn \stex_invoke_variable:n {
3576   \if_mode_math:
3577     \exp_after:wN \__stex_variables_invoke_math:n
3578   \else:
3579     \exp_after:wN \__stex_variables_invoke_text:n
3580   \fi: {#1}
3581 }
3582
3583 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
3584   %TODO
3585 }
3586
3587
3588 \cs_new_protected:Nn \__stex_variables_invoke_math:n {
3589   \peek_charcode_remove:NTF ! {
3590     \peek_charcode_remove:NTF ! {
3591       \peek_charcode:NTF [ {

```

```

3592     \__stex_variables_invoke_op_custom:nw
3593   }{
3594     % TODO throw error
3595   }
3596 }{
3597   \__stex_variables_invoke_op:n { #1 }
3598 }
3599 }{
3600   \peek_charcode_remove:NTF * {
3601     \__stex_variables_invoke_text:n { #1 }
3602   }{
3603     \__stex_variables_invoke_math_ii:n { #1 }
3604   }
3605 }
3606 }
3607
3608 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
3609   \cs_if_exist:cTF {
3610     stex_var_op_notation_ #1 _cs
3611   }{
3612     \exp_args:Nnx \use:nn {
3613       \def\comp{\_varcomp}
3614       \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3615       \_stex_term_omv:nn { var://#1 }{
3616         \use:c{stex_var_op_notation_ #1 _cs }
3617       }
3618     }{
3619       \_stex_reset:N \comp
3620       \_stex_reset:N \l_stex_current_symbol_str
3621     }
3622   }{
3623     \int_compare:nNnTF {\prop_item:cn {\l_stex_variable_#1_prop}{arity}} = 0{
3624       \__stex_variables_invoke_math_ii:n {#1}
3625     }{
3626       \msg_error:nnxx{stex}{error/noop}{variable~#1}{-}
3627     }
3628   }
3629 }
3630
3631 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n #1 {
3632   \cs_if_exist:cTF {
3633     stex_var_notation_#1_cs
3634   }{
3635     \tl_set:Nx \stex_symbol_after_invokation_tl {
3636       \_stex_reset:N \comp
3637       \_stex_reset:N \stex_symbol_after_invokation_tl
3638       \_stex_reset:N \l_stex_current_symbol_str
3639       \bool_set_true:N \l_stex_allow_semantic_bool
3640     }
3641     \def\comp{\_varcomp}
3642     \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3643     \bool_set_false:N \l_stex_allow_semantic_bool
3644     \use:c{stex_var_notation_#1_cs}
3645   }{

```

```

3646 \msg_error:nxx{stex}{error/nonotation}{variable~#1}{s}
3647 }
3648 }

```

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

## 30.5 Sequences

```

3649 <@@=stex_sequences>
3650
3651 \cs_new_protected:Nn \stex_invoke_sequence:n {
3652   \peek_charcode_remove:NTF ! {
3653     \stex_term_omv:nn {varseq://#1}{
3654       \exp_args:Nnx \use:nn {
3655         \def\comp{\_varcomp}
3656         \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3657         \prop_item:cn{stex_varseq_#1_prop}{notation}
3658       }{
3659         \stex_reset:N \comp
3660         \stex_reset:N \l_stex_current_symbol_str
3661       }
3662     }
3663   }{
3664     \bool_set_false:N \l_stex_allow_semantic_bool
3665     \def\comp{\_varcomp}
3666     \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3667     \tl_set:Nx \stex_symbol_after_invokation_tl {
3668       \stex_reset:N \comp
3669       \stex_reset:N \stex_symbol_after_invokation_tl
3670       \stex_reset:N \l_stex_current_symbol_str
3671       \bool_set_true:N \l_stex_allow_semantic_bool
3672     }
3673     \use:c { stex_varseq_#1_cs }
3674   }
3675 }
3676 </package>

```



## Chapter 31

# STEX -Structural Features Implementation

```
3677 ⟨*package⟩
3678
3679 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3680
3681 Warnings and error messages
3682 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3683   Symbol~#1~can~not~be~assigned~in~copymodule~#2
3684 }
3685 \msg_new:nnn{stex}{error/interpretmodule/nodefiniens}{
3686   Symbol~#1~not~assigned~in~interpretmodule~#2
3687 }
3688 \msg_new:nnn{stex}{error/unknownstructure}{
3689   No~structure~#1~found!
3690 }
3691
3692 \msg_new:nnn{stex}{error/unknownfield}{
3693   No~field~#1~in~instance~#2~found!\#3
3694 }
3695
3696 \msg_new:nnn{stex}{error/keyval}{
3697   Invalid~key=value~pair~#1
3698 }
3699 \msg_new:nnn{stex}{error/instantiate/missing}{
3700   Assignments~missing~in~instantiate:~#1
3701 }
3702 \msg_new:nnn{stex}{error/incompatible}{
3703   Incompatible~signature:~#1~(#2)~and~#3~(#4)
3704 }
3705
```

## 31.1 Imports with modification

```

3706 <@@=stex_copymodule>
3707 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
3708   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3709     \tl_set:Nn \l_tmpa_tl { #1 }
3710     \__stex_copymodule_get_symbol_from_cs:
3711   }{
3712     % argument is a string
3713     % is it a command name?
3714     \cs_if_exist:cTF { #1 }{
3715       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3716       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3717       \str_if_empty:NNTF \l_tmpa_str {
3718         \exp_args:Nx \cs_if_eq:NNTF {
3719           \tl_head:N \l_tmpa_tl
3720         } \stex_invoke_symbol:n {
3721           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
3722         }{
3723           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3724         }
3725       } {
3726         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3727       }
3728     }{
3729       % argument is not a command name
3730       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3731       % \l_stex_all_symbols_seq
3732     }
3733   }
3734 }
3735
3736 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
3737   \str_set:Nn \l_tmpa_str { #1 }
3738   \bool_set_false:N \l_tmpa_bool
3739   \bool_if:NF \l_tmpa_bool {
3740     \tl_set:Nn \l_tmpa_tl {
3741       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
3742     }
3743     \str_set:Nn \l_tmpa_str { #1 }
3744     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3745     \seq_map_inline:Nn #2 {
3746       \str_set:Nn \l_tmpb_str { ##1 }
3747       \str_if_eq:eeT { \l_tmpa_str } {
3748         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3749       } {
3750         \seq_map_break:n {
3751           \tl_set:Nn \l_tmpa_tl {
3752             \str_set:Nn \l_stex_get_symbol_uri_str {
3753               ##1
3754             }
3755           }
3756         }
3757       }

```

```

3758     }
3759     \l_tmpa_tl
3760   }
3761 }
3762
3763 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
3764   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3765     { \tl_tail:N \l_tmpa_tl }
3766   \tl_if_single:NTF \l_tmpa_tl {
3767     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3768       \exp_after:wN \str_set:Nn \exp_after:wN
3769         \l_stex_get_symbol_uri_str \l_tmpa_tl
3770       \__stex_copymodule_get_symbol_check:n { #1 }
3771     }{
3772       % TODO
3773       % tail is not a single group
3774     }
3775   }{
3776     % TODO
3777     % tail is not a single group
3778   }
3779 }
3780
3781 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
3782   \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
3783     \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3784       :~\seq_use:Nn #1 {,~}
3785     }
3786   }
3787 }
3788
3789 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
3790   \stex_import_module_uri:nn { #1 } { #2 }
3791   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3792   \stex_import_require_module:nnnn
3793     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3794     { \l_stex_import_path_str } { \l_stex_import_name_str }
3795   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3796   \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
3797   \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
3798   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
3799     \seq_map_inline:cn {c_stex_module_###1_constants}{
3800       \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
3801         ##1 ? ####1
3802       }
3803     }
3804   }
3805   \seq_clear:N \l_tmpa_seq
3806   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3807     name      = \l_stex_current_copymodule_name_str ,
3808     module    = \l_stex_current_module_str ,
3809     from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3810     includes  = \l_tmpa_seq ,
3811     fields    = \l_tmpa_seq

```

```

3812 }
3813 \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3814   as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3815   \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_seq {,
3816 \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
3817 \stex_if_smsmode:F {
3818   \begin{stex_annotate_env} {#4} {
3819     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3820   }
3821   \stex_annotate_invisible:nnn{from}{\l_stex_import_ns_str ?\l_stex_import_name_str}{}
3822 }
3823 \bool_set_eq:NN \l__stex_copymodule_oldhtml_bool \_stex_html_do_output_bool
3824 \bool_set_false:N \_stex_html_do_output_bool
3825 }
3826 \cs_new_protected:Nn \stex_copymodule_end:n {
3827   \def \l_tmpa_cs ##1 ##2 {#1}
3828   \bool_set_eq:NN \_stex_html_do_output_bool \l__stex_copymodule_oldhtml_bool
3829   \tl_clear:N \l_tmpa_tl
3830   \tl_clear:N \l_tmpb_tl
3831   \prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3832   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
3833     \seq_map_inline:cn {c_stex_module_##1_constants}{
3834       \tl_clear:N \l_tmpc_tl
3835       \l_tmpa_cs{##1}{####1}
3836       \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
3837         \tl_put_right:Nx \l_tmpa_tl {
3838           \prop_set_from_keyval:cn {
3839             l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_name_str}
3840           }{
3841             \exp_after:wN \prop_to_keyval:N \csname
3842               l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_name_str}
3843             \endcsname
3844           }
3845           \seq_clear:c {
3846             l_stex_symdecl_
3847             \l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_name_str}
3848             _notations
3849           }
3850         }
3851       }
3852       \tl_put_right:Nx \l_tmpc_tl {
3853         \stex_copy_notations:nn {\l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_name_str}
3854         \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?####1_name_str}
3855       }
3856       \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_name_str}
3857       \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
3858         \tl_put_right:Nx \l_tmpc_tl {
3859           \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1?####1_name_str}
3860         }
3861         \tl_put_right:Nx \l_tmpa_tl {
3862           \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
3863             \stex_invoke_symbol:n {
3864               \l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_name_str}
3865             }
3866           }
3867         }
3868       }
3869     }
3870   }

```

```

3866     }
3867   }
3868 }{
3869   \tl_put_right:Nx \l_tmpc_tl {
3870     \stex_copy_notations:nn {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str}
3871   }
3872   \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
3873   \prop_put:Nnx \l_tmpa_prop { name }{ \l_stex_current_copymodule_name_str / ####1 }
3874   \prop_put:Nnx \l_tmpa_prop { module }{ \l_stex_current_module_str }
3875   \tl_put_right:Nx \l_tmpa_tl {
3876     \prop_set_from_keyval:cn {
3877       l_stex_symdecl_ \l_stex_current_module_str ? \l_stex_current_copymodule_name_str
3878     }{
3879       \prop_to_keyval:N \l_tmpa_prop
3880     }
3881     \seq_clear:c {
3882       l_stex_symdecl_
3883       \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3884       _notations
3885     }
3886   }
3887   \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str}
3888   \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
3889     \tl_put_right:Nx \l_tmpc_tl {
3890       \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}
3891     }
3892     \tl_put_right:Nx \l_tmpa_tl {
3893       \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
3894         \stex_invoke_symbol:n {
3895           \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3896         }
3897       }
3898     }
3899   }
3900 }
3901 \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
3902   \tl_put_right:Nx \l_tmpc_tl {
3903     \stex_annotate_invisible:nnn{definiens}{\use:c{l__stex_copymodule_copymodule_##1?####1_def_tl}}
3904   }
3905 }
3906 \tl_put_right:Nx \l_tmpb_tl {
3907   \stex_annotate:nnn{assignment} {##1?####1} { \l_tmpc_tl }
3908 }
3909 }
3910 }
3911 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3912 \tl_put_left:Nx \l_tmpa_tl {
3913   \prop_set_from_keyval:cn {
3914     l_stex_copymodule_ \l_stex_current_module_str? \l_stex_current_copymodule_name_str _prop
3915   }{
3916     \prop_to_keyval:N \l_stex_current_copymodule_prop
3917   }
3918 }
3919 \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl

```

```

3920 \stex_debug:nn{copymodule}{result:\meaning \l_tmpa_tl}
3921 \exp_args:Nx \stex_do_up_to_module:n {
3922   \exp_args:No \exp_not:n \l_tmpa_tl
3923 }
3924 \l_tmpb_tl
3925 \stex_if_smsmode:F {
3926   \end{stex_annotate_env}
3927 }
3928 }
3929
3930 \NewDocumentEnvironment {copymodule} { 0{} m m}{
3931   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ structure }
3932   \stex_deactivate_macro:Nn \symdecl {module~environments}
3933   \stex_deactivate_macro:Nn \symdef {module~environments}
3934   \stex_deactivate_macro:Nn \notation {module~environments}
3935   \stex_reactivate_macro:N \assign
3936   \stex_reactivate_macro:N \renamedekl
3937   \stex_reactivate_macro:N \donotcopy
3938   \stex_smsmode_do:
3939 }{
3940   \stex_copymodule_end:n {}
3941 }
3942
3943 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
3944   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ realization }
3945   \stex_deactivate_macro:Nn \symdecl {module~environments}
3946   \stex_deactivate_macro:Nn \symdef {module~environments}
3947   \stex_deactivate_macro:Nn \notation {module~environments}
3948   \stex_reactivate_macro:N \assign
3949   \stex_reactivate_macro:N \renamedekl
3950   \stex_reactivate_macro:N \donotcopy
3951   \stex_smsmode_do:
3952 }{
3953   \stex_copymodule_end:n {
3954     \tl_if_exist:cF {
3955       l__stex_copymodule_copymodule_##1?##2_def_tl
3956     }{
3957       \str_if_eq:eeF {
3958         \prop_item:cn{
3959           l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
3960         }{ true }{
3961           \msg_error:nxxx{stex}{error/interpretmodule/nodéfiniens}{
3962             ##1?##2
3963           }{\l_stex_current_copymodule_name_str}
3964         }
3965       }
3966     }
3967   }
3968
3969 \NewDocumentCommand \donotcopy { 0{} m}{
3970   \stex_import_module_uri:nn { #1 } { #2 }
3971   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3972   \seq_map_inline:Nn \l_stex_collect_imports_seq {
3973     \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }

```

```

3974 \seq_map_inline:cn {c_stex_module_##1_constants}{
3975 \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? #####1 }
3976 \bool_lazy_any_p:nT {
3977   { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_name_str}}
3978   { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_macroname_str}}
3979   { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_def_tl}}
3980 }{
3981   % TODO throw error
3982 }
3983 }
3984 }
3985
3986 \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
3987 \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_ns_str ?\l_stex_import_name_str }
3988 \prop_put:Nnx \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
3989 }
3990
3991 \NewDocumentCommand \assign { m m }{
3992 \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
3993 \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
3994 \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _def_tl}{#2}
3995 }
3996
3997 \keys_define:nn { stex / renamedec1 } {
3998   name .str_set_x:N = \l_stex_renamedec1_name_str
3999 }
4000 \cs_new_protected:Nn \__stex_copymodule_renamedec1_args:n {
4001 \str_clear:N \l_stex_renamedec1_name_str
4002 \keys_set:nn { stex / renamedec1 } { #1 }
4003 }
4004
4005 \NewDocumentCommand \renamedec1 { 0{} m m }{
4006 \__stex_copymodule_renamedec1_args:n { #1 }
4007 \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
4008 \stex_debug:nn{renamedec1}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
4009 \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
4010 \str_if_empty:NTF \l_stex_renamedec1_name_str {
4011 \tl_set:cx { #3 }{\stex_invoke_symbol:n {
4012 \l_stex_get_symbol_uri_str
4013 } }
4014 } {
4015 \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex
4016 \stex_debug:nn{renamedec1}{@~\l_stex_current_module_str ? \l_stex_renamedec1_name_str}
4017 \prop_set_eq:cc {l_stex_symdec1_
4018 \l_stex_current_module_str ? \l_stex_renamedec1_name_str
4019 _prop
4020 }{l_stex_symdec1_ \l_stex_get_symbol_uri_str _prop}
4021 \seq_set_eq:cc {l_stex_symdec1_
4022 \l_stex_current_module_str ? \l_stex_renamedec1_name_str
4023 _notations
4024 }{l_stex_symdec1_ \l_stex_get_symbol_uri_str _notations}
4025 \prop_put:cnx {l_stex_symdec1_
4026 \l_stex_current_module_str ? \l_stex_renamedec1_name_str
4027 _prop

```

```

4028     }{ name }{ \l_stex_renamedekl_name_str }
4029     \prop_put:cnx {l_stex_symdecl_
4030       \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4031       _prop
4032     }{ module }{ \l_stex_current_module_str }
4033     \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
4034       \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4035     }
4036     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4037       \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4038     } }
4039   }
4040 }
4041
4042 \stex_deactivate_macro:Nn \assign {copymodules}
4043 \stex_deactivate_macro:Nn \renamedekl {copymodules}
4044 \stex_deactivate_macro:Nn \donotcopy {copymodules}
4045
4046
4047 \seq_new:N \l_stex_implicit_morphisms_seq
4048 \NewDocumentCommand \implicitmorphism { 0{} m m }{
4049   \stex_import_module_uri:nn { #1 } { #2 }
4050   \stex_debug:nn{implicits}{
4051     Implicit~morphism:~
4052     \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4053   }
4054   \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
4055     \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4056   }{
4057     \msg_error:nnn{stex}{error/conflictingmodules}{
4058       \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4059     }
4060   }
4061 }
4062 % TODO
4063
4064
4065
4066 \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
4067   \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4068 }
4069 }
4070

```

## 31.2 The feature environment

structural@feature

```

4071 <@@=stex_features>
4072
4073 \NewDocumentEnvironment{structural_feature_module}{ m m m }{
4074   \stex_if_in_module:F {
4075     \msg_set:nnn{stex}{error/nomodule}{
4076       Structural~Feature~has~to~occur~in~a~module:\\

```



```

4077     Feature~#2~of~type~#1\\
4078     In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
4079   }
4080   \msg_error:nn{stex}{error/nomodule}
4081 }
4082
4083 \stex_module_setup:nn{meta=NONE}{#2 - #1}
4084
4085 \stex_if_smsmode:F {
4086   \begin{stex_annotate_env}{ feature:#1 }{}
4087   \stex_annotate_invisible:nnn{header}{}{ #3 }
4088 }
4089 }{
4090   \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
4091   \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
4092   \stex_debug:nn{features}{
4093     Feature: \l_stex_last_feature_str
4094   }
4095   \stex_if_smsmode:F {
4096     \end{stex_annotate_env}
4097   }
4098 }

```

### 31.3 Structure

structure

```

4099 <@@=stex_structures>
4100 \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
4101   \prop_if_exist:cF {c_stex_module_ \l_stex_current_module_str _structures}{
4102     \prop_new:c {c_stex_module_ \l_stex_current_module_str _structures}
4103   }
4104   \prop_gput:cxx{c_stex_module_ \l_stex_current_module_str _structures}
4105   {#1}{#2}
4106 }
4107
4108 \keys_define:nn { stex / features / structure } {
4109   name .str_set_x:N = \l__stex_structures_name_str ,
4110 }
4111
4112 \cs_new_protected:Nn \__stex_structures_structure_args:n {
4113   \str_clear:N \l__stex_structures_name_str
4114   \keys_set:nn { stex / features / structure } { #1 }
4115 }
4116
4117 \NewDocumentEnvironment{mathstructure}{m O{}}{
4118   \__stex_structures_structure_args:n { #2 }
4119   \str_if_empty:NT \l__stex_structures_name_str {
4120     \str_set:Nx \l__stex_structures_name_str { #1 }
4121   }
4122   \exp_args:Nnnx
4123   \begin{structural_feature_module}{ structure }
4124     { \l__stex_structures_name_str }{}
4125   \stex_smsmode_do:

```

```

4126 }{
4127   \end{structural_feature_module}
4128   \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
4129   \seq_clear:N \l_tmpa_seq
4130   \seq_map_inline:Nn \l_stex_collect_imports_seq {
4131     \seq_map_inline:cn{c_stex_module_##1_constants}{
4132       \seq_put_right:Nn \l_tmpa_seq { ##1 ? ####1 }
4133     }
4134   }
4135   \exp_args:Nnno
4136   \prop_gput:cnm {c_stex_module_ \l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
4137   \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
4138   \stex_add_structure_to_current_module:nn
4139     \l__stex_structures_name_str
4140     \l_stex_last_feature_str
4141   \exp_args:Nx \stex_symdecl_do:nn {
4142     name = \l__stex_structures_name_str ,
4143     type = \metacollection ,
4144     def = {\STEXsymbol{module-type}{
4145       \stex_term_math_oms:nnnn { \l_stex_last_feature_str }{}{0}{}}
4146     }}
4147   }{ #1 }
4148   \exp_args:Nx
4149   \stex_add_to_current_module:n {
4150     \tl_set:cn { #1 }{
4151       \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures
4152     }
4153   }
4154   \exp_args:Nx
4155   \stex_do_up_to_module:n {
4156     \tl_set:cn { #1 }{
4157       \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures
4158     }
4159   }
4160 }
4161 \seq_put_right:Nx \g_stex_smsmode_allowedenvs_seq { \tl_to_str:n {mathstructure}}
4162
4163 \cs_new:Nn \stex_invoke_structure:nn {
4164   \stex_invoke_symbol:n { #1?#2 }
4165 }
4166
4167 \cs_new_protected:Nn \stex_get_structure:n {
4168   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4169     \tl_set:Nn \l_tmpa_tl { #1 }
4170     \__stex_structures_get_from_cs:
4171   }{
4172     \cs_if_exist:cTF { #1 }{
4173       \cs_set_eq:Nc \l_tmpa_cs { #1 }
4174       \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
4175       \str_if_empty:NTF \l_tmpa_str {
4176         \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs } \stex_invoke_structure:nn {
4177           \__stex_structures_get_from_cs:
4178         }{
4179           \__stex_structures_get_from_string:n { #1 }

```

```

4180     }
4181   }{
4182     \__stex_structures_get_from_string:n { #1 }
4183   }
4184   }{
4185     \__stex_structures_get_from_string:n { #1 }
4186   }
4187 }
4188 }
4189
4190 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
4191   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4192     { \tl_tail:N \l_tmpa_tl }
4193   \str_set:Nx \l_tmpa_str {
4194     \exp_after:wN \use_i:nn \l_tmpa_tl
4195   }
4196   \str_set:Nx \l_tmpb_str {
4197     \exp_after:wN \use_ii:nn \l_tmpa_tl
4198   }
4199   \str_set:Nx \l_stex_get_structure_str {
4200     \l_tmpa_str ? \l_tmpb_str
4201   }
4202   \str_set:Nx \l_stex_get_structure_module_str {
4203     \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
4204   }
4205 }
4206
4207 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
4208   \tl_set:Nn \l_tmpa_tl {
4209     \msg_error:nnn{stex}{error/unknownstructure}{#1}
4210   }
4211   \str_set:Nn \l_tmpa_str { #1 }
4212   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4213
4214   \seq_map_inline:Nn \l_stex_all_modules_seq {
4215     \prop_if_exist:cT {c_stex_module_##1_structures} {
4216       \prop_map_inline:cn {c_stex_module_##1_structures} {
4217         \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
4218           \prop_map_break:n{ \seq_map_break:n{
4219             \tl_set:Nn \l_tmpa_tl {
4220               \str_set:Nn \l_stex_get_structure_str {##1?####1}
4221               \str_set:Nn \l_stex_get_structure_module_str {####2}
4222             }
4223           }}
4224         }
4225       }
4226     }
4227   }
4228   \l_tmpa_tl
4229 }

```

\instantiate

```

4230
4231 \keys_define:nn { stex / instantiate } {

```

```

4232     name .str_set_x:N = \l__stex_structures_name_str
4233 }
4234 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
4235     \str_clear:N \l__stex_structures_name_str
4236     \keys_set:nn { stex / instantiate } { #1 }
4237 }
4238
4239 \NewDocumentCommand \instantiate {m O{} m m m}{
4240     \begin{group}
4241         \stex_get_structure:n {#4}
4242         \__stex_structures_instantiate_args:n { #2 }
4243         \str_if_empty:NT \l__stex_structures_name_str {
4244             \str_set:Nn \l__stex_structures_name_str { #1 }
4245         }
4246         \seq_clear:N \l__stex_structures_fields_seq
4247         \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4248         \seq_map_inline:Nn \l_stex_collect_imports_seq {
4249             \seq_map_inline:cn {c_stex_module_##1_constants}{
4250                 \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? #####1 }
4251             }
4252         }
4253         \seq_set_split:Nnn \l_tmpa_seq , {#3}
4254         \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4255         \prop_clear:N \l_tmpa_prop
4256         \seq_map_inline:Nn \l_tmpa_seq {
4257             \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4258             \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4259                 \msg_error:nnn{stex}{error/keyval}{##1}
4260             }
4261             \exp_args:Nx \stex_get_symbol_in_seq:nn { \seq_item:Nn \l_tmpb_seq 1 } \l__stex_structures_name_str
4262             \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4263             \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_uri_str
4264             \exp_args:Nx \stex_get_symbol:n { \seq_item:Nn \l_tmpb_seq 2 }
4265             \exp_args:Nxx \str_if_eq:nnF
4266                 { \prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args} }
4267                 { \prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args} } {
4268                 \msg_error:nnxxx{stex}{error/incompatible}
4269                 { \l__stex_structures_dom_str }
4270                 { \prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args} }
4271                 { \l_stex_get_symbol_uri_str }
4272                 { \prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args} }
4273             }
4274             \prop_put:Nxx \l_tmpa_prop { \seq_item:Nn \l_tmpb_seq 1 } \l_stex_get_symbol_uri_str
4275         }
4276         \seq_if_empty:NF \l__stex_structures_fields_seq {
4277             \msg_error:nnx{stex}{error/instantiate/missing}{ \seq_use:Nn \l__stex_structures_fields_seq }
4278         }
4279         \exp_args:Nx
4280         \stex_add_to_current_module:n {
4281             \prop_set_from_keyval:cn {l_stex_instance\l_stex_current_module_str?\l__stex_structures_name_str}
4282             domain = \l_stex_get_structure_module_str ,
4283             \prop_to_keyval:N \l_tmpa_prop
4284         }
4285         \tl_set:cn{ #1 }{ \stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structures_name_str }

```

```

4286     }
4287     \exp_args:Nx
4288     \stex_do_up_to_module:n {
4289         \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
4290             domain = \l_stex_get_structure_module_str ,
4291             \prop_to_keyval:N \l_tmpa_prop
4292         }
4293         \tl_set:cn{ #1 }{\stex_invoke_instance:n{\l_stex_current_module_str?\l__stex_structur
4294     }
4295     \stex_debug:nn{instantiate}{
4296         Instance~\l_stex_current_module_str?\l__stex_structures_name_str \
4297         \prop_to_keyval:N \l_tmpa_prop
4298     }
4299     \exp_args:Nxx \stex_symdecl_do:nn {
4300         type={\STEXsymbol{module-type}}{
4301             \_stex_term_math_oms:nnnn {
4302                 \l_stex_get_structure_module_str
4303             }{}{0}{}
4304         }}
4305     }{\l__stex_structures_name_str}
4306     \exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}
4307     \endgroup
4308     \stex_smsmode_do:\ignorespacesandpars
4309 }
4310 \tl_put_right:Nx \g_stex_smsmode_allowedmacros_escape_tl {\instantiate}
4311
4312 \cs_new_protected:Nn \stex_symbol_or_var:n {
4313     \cs_if_exist:cTF{#1}{
4314         \cs_set_eq:Nc \l_tmpa_tl { #1 }
4315         \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4316         \str_if_empty:NTF \l_tmpa_str {
4317             \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
4318                 \stex_invoke_variable:n {
4319                     \bool_set_true:N \l_stex_symbol_or_var_bool
4320                     \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4321                     \str_set:Nx \l_stex_get_symbol_uri_str {
4322                         \exp_after:wN \use:n \l_tmpa_tl
4323                     }
4324                 }{
4325                     \bool_set_false:N \l_stex_symbol_or_var_bool
4326                     \stex_get_symbol:n{#1}
4327                 }
4328         }{
4329             \__stex_structures_symbolorvar_from_string:n{ #1 }
4330         }
4331     }{
4332         \__stex_structures_symbolorvar_from_string:n{ #1 }
4333     }
4334 }
4335
4336 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
4337     \prop_if_exist:cTF {l_stex_variable_#1 _prop}{
4338         \bool_set_true:N \l_stex_symbol_or_var_bool
4339         \str_set:Nn \l_stex_get_symbol_uri_str { #1 }

```

```

4340 }{
4341   \bool_set_false:N \l_stex_symbol_or_var_bool
4342   \stex_get_symbol:n{#1}
4343 }
4344 }
4345
4346
4347 \NewDocumentCommand \varinstantiate {m O{} m m m}{
4348   \beginngroup
4349     \stex_get_structure:n {#4}
4350     \__stex_structures_instantiate_args:n { #2 }
4351     \str_if_empty:NT \l__stex_structures_name_str {
4352       \str_set:Nn \l__stex_structures_name_str { #1 }
4353     }
4354     \seq_clear:N \l__stex_structures_fields_seq
4355     \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4356     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4357       \seq_map_inline:cn {c_stex_module_##1_constants}{
4358         \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? #####1 }
4359       }
4360     }
4361     \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4362     \prop_clear:N \l_tmpa_prop
4363     \tl_if_empty:nF {#3} {
4364       \seq_set_split:Nnn \l_tmpa_seq , {#3}
4365       \seq_map_inline:Nn \l_tmpa_seq {
4366         \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4367         \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4368           \msg_error:nnn{stex}{error/keyval}{##1}
4369         }
4370         \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
4371         \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4372         \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
4373         \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
4374         \bool_if:NTF \l_stex_symbol_or_var_bool {
4375           \exp_args:Nxx \str_if_eq:nnF
4376             {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4377             {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}{
4378             \msg_error:nnxxx{stex}{error/incompatible}
4379             {\l__stex_structures_dom_str
4380              {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4381              {\l_stex_get_symbol_uri_str
4382               {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}}
4383           }
4384           \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n {
4385         }{
4386           \exp_args:Nxx \str_if_eq:nnF
4387             {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4388             {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
4389             \msg_error:nnxxx{stex}{error/incompatible}
4390             {\l__stex_structures_dom_str
4391              {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4392              {\l_stex_get_symbol_uri_str
4393               {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}}

```

```

4394     }
4395     \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n {\l
4396   }
4397 }
4398 }
4399 \tl_gclear:N \g__stex_structures_aftergroup_tl
4400 \seq_map_inline:Nn \l__stex_structures_fields_seq {
4401   \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdecl
4402   \stex_find_notation:nn{##1}{}}
4403   \cs_gset_eq:cc{g__stex_structures_tmpa_\l_tmpa_str_cs}
4404     {stex_notation_##1\c_hash_str \l_stex_notation_variant_str_cs}
4405   \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str_cs}{
4406     \cs_gset_eq:cc {g__stex_structures_tmpa_op_\l_tmpa_str_cs}
4407     {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str_cs}
4408 }
4409
4410 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4411   \prop_set_from_keyval:cn { l_stex_variable_ \l_tmpa_str_prop}{
4412     name   = \l_tmpa_str ,
4413     args   = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4414     arity  = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4415     assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4416   }
4417   \cs_set_eq:cc {stex_var_notation_\l_tmpa_str_cs}
4418   {g__stex_structures_tmpa_\l_tmpa_str_cs}
4419   \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str_cs}
4420   {g__stex_structures_tmpa_op_\l_tmpa_str_cs}
4421 }
4422 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_invo
4423 }
4424 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4425   \prop_set_from_keyval:cn {l_stex_varinstance_\l__stex_structures_name_str_prop }{
4426     domain = \l_stex_get_structure_module_str ,
4427     \prop_to_keyval:N \l_tmpa_prop
4428   }
4429   \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
4430   \tl_set:cn {l_stex_varinstance_\l__stex_structures_name_str_op_tl}{
4431     \exp_args:Nnx \exp_not:N \use:nn {
4432       \str_set:Nn \exp_not:N \l_stex_current_symbol_str {var://\l__stex_structures_name
4433       \_stex_term_omv:nn {var://\l__stex_structures_name_str}{
4434         \exp_not:n{
4435           \_varcomp{#5}
4436         }
4437       }
4438     }{
4439       \exp_not:n{\_stex_reset:N \l_stex_current_symbol_str}
4440     }
4441   }
4442 }
4443 \aftergroup\g__stex_structures_aftergroup_tl
4444 \endgroup
4445 \stex_smsmode_do:\ignorespacesandpars
4446 }
4447

```

```

4448 \cs_new_protected:Nn \stex_invoke_instance:n {
4449   \peek_charcode_remove:NTF ! {
4450     \stex_invoke_symbol:n{#1}
4451   }{
4452     \stex_invoke_instance:nn {#1}
4453   }
4454 }
4455
4456
4457 \cs_new_protected:Nn \stex_invoke_varinstance:n {
4458   \peek_charcode_remove:NTF ! {
4459     \use:c{l_stex_varinstance_#1_op_tl}
4460   }{
4461     \stex_invoke_varinstance:nn {#1}
4462   }
4463 }
4464
4465 \cs_new_protected:Nn \stex_invoke_instance:nn {
4466   \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
4467     \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
4468   }{
4469     \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
4470     \msg_error:nnnn{stex}{error/unknownfield}{#2}{#1}{
4471       \prop_to_keyval:N \l_tmpa_prop
4472     }
4473   }
4474 }
4475
4476 \cs_new_protected:Nn \stex_invoke_varinstance:nn {
4477   \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
4478     \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
4479     \l_tmpa_tl
4480   }{
4481     \msg_error:nnnn{stex}{error/unknownfield}{#2}{#1}{
4482     }
4483   }

```

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

\stex\_invoke\_structure:nnn

```

4484 % #1: URI of the instance
4485 % #2: URI of the instantiated module
4486 \cs_new_protected:Nn \stex_invoke_structure:nnn {
4487   \tl_if_empty:nTF{ #3 }{
4488     \prop_set_eq:Nc \l__stex_structures_structure_prop {
4489       c_stex_feature_ #2 _prop
4490     }
4491     \tl_clear:N \l_tmpa_tl
4492     \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
4493     \seq_map_inline:Nn \l_tmpa_seq {
4494       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
4495       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
4496       \cs_if_exist:cT {
4497         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs

```



```

4498     }{
4499       \tl_if_empty:NF \l_tmpa_tl {
4500         \tl_put_right:Nn \l_tmpa_tl {,}
4501       }
4502       \tl_put_right:Nx \l_tmpa_tl {
4503         \stex_invoke_symbol:n {#1/\l_tmpa_str}!
4504       }
4505     }
4506   }
4507   \exp_args:No \mathstruct \l_tmpa_tl
4508   }{
4509     \stex_invoke_symbol:n{#1/#3}
4510   }
4511 }

```

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

```

4512 </package>

```

## Chapter 32

# STEX -Statements Implementation

```
4513 <*package>
4514
4515 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4516
4517 <@@=stex_statements>
    Warnings and error messages
4518
\titleemph
4519 \def\titleemph#1{\textbf{#1}}
    (End definition for \titleemph. This function is documented on page ??.)
```

### 32.1 Definitions

#### definiendum

```
4520 \keys_define:nn {stex / definiendum }{
4521   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
4522   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
4523   root     .str_set_x:N    = \l__stex_statements_definiendum_root_str,
4524   gfa      .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
4525 }
4526 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
4527   \str_clear:N \l__stex_statements_definiendum_root_str
4528   \tl_clear:N \l__stex_statements_definiendum_post_tl
4529   \str_clear:N \l__stex_statements_definiendum_gfa_str
4530   \keys_set:nn { stex / definiendum }{ #1 }
4531 }
4532 \NewDocumentCommand \definiendum { O{} m m } {
4533   \__stex_statements_definiendum_args:n { #1 }
4534   \stex_get_symbol:n { #2 }
4535   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4536   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
4537     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```

4538     \tl_set:Nn \l_tmpa_tl { #3 }
4539   } {
4540     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
4541     \tl_set:Nn \l_tmpa_tl {
4542       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
4543     }
4544   }
4545 } {
4546   \tl_set:Nn \l_tmpa_tl { #3 }
4547 }
4548
4549 % TODO root
4550 \rustex_if:TF {
4551   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
4552 } {
4553   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
4554 }
4555 }
4556 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

*(End definition for definiendum. This function is documented on page 6.)*

#### definame

```

4557
4558 \NewDocumentCommand \definame { 0{ } m } {
4559   \__stex_statements_definiendum_args:n { #1 }
4560   % TODO: root
4561   \stex_get_symbol:n { #2 }
4562   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4563   \str_set:Nx \l_tmpa_str {
4564     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4565   }
4566   \str_replace_all:Nnn \l_tmpa_str {-} {~}
4567   \rustex_if:TF {
4568     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4569       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4570     }
4571   } {
4572     \exp_args:Nnx \defemph@uri {
4573       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4574     } { \l_stex_get_symbol_uri_str }
4575   }
4576 }
4577 \stex_deactivate_macro:Nn \definame {definition~environments}
4578
4579 \NewDocumentCommand \Definame { 0{ } m } {
4580   \__stex_statements_definiendum_args:n { #1 }
4581   \stex_get_symbol:n { #2 }
4582   \str_set:Nx \l_tmpa_str {
4583     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4584   }
4585   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4586   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4587   \rustex_if:TF {

```

```

4588 \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4589 \l_tmpa_str\l__stex_statements_definiendum_post_tl
4590 }
4591 } {
4592 \exp_args:Nnx \defemph@uri {
4593 \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
4594 } { \l_stex_get_symbol_uri_str }
4595 }
4596 }
4597 \stex_deactivate_macro:Nn \Definame {definition~environments}
4598
4599 \NewDocumentCommand \premise { m }{
4600 \stex_annotate:nnn{ premise }{}{ #1 }
4601 }
4602 \NewDocumentCommand \conclusion { m }{
4603 \stex_annotate:nnn{ conclusion }{}{ #1 }
4604 }
4605 \NewDocumentCommand \definiens { m }{
4606 \stex_annotate:nnn{ definiens }{}{ #1 }
4607 }
4608
4609 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
4610 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
4611 \stex_deactivate_macro:Nn \definiens {definition~environments}
4612

```

(End definition for `definame`. This function is documented on page 6.)

## sdefinition

```

4613
4614 \keys_define:nn {stex / sdefinition }{
4615 type .str_set_x:N = \sdefinitiontype,
4616 id .str_set_x:N = \sdefinitionid,
4617 name .str_set_x:N = \sdefinitionname,
4618 for .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4619 title .tl_set:N = \sdefinitiontitle
4620 }
4621 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
4622 \str_clear:N \sdefinitiontype
4623 \str_clear:N \sdefinitionid
4624 \str_clear:N \sdefinitionname
4625 \clist_clear:N \l__stex_statements_sdefinition_for_clist
4626 \tl_clear:N \sdefinitiontitle
4627 \keys_set:nn { stex / sdefinition }{}{ #1 }
4628 }
4629
4630 \NewDocumentEnvironment{sdefinition}{0{}}{
4631 \__stex_statements_sdefinition_args:n{ #1 }
4632 \stex_reactivate_macro:N \definiendum
4633 \stex_reactivate_macro:N \definame
4634 \stex_reactivate_macro:N \Definame
4635 \stex_reactivate_macro:N \premise
4636 \stex_reactivate_macro:N \definiens
4637 \stex_if_smsmode:F{

```

```

4638 \seq_clear:N \l_tmpa_seq
4639 \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4640   \tl_if_empty:nF{ ##1 }{
4641     \stex_get_symbol:n { ##1 }
4642     \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4643       \l_stex_get_symbol_uri_str
4644     }
4645   }
4646 }
4647 \exp_args:Nnnx
4648 \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {,}}
4649 \str_if_empty:NF \sdefinitiontype {
4650   \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{}
4651 }
4652 \clist_set:No \l_tmpa_clist \sdefinitiontype
4653 \tl_clear:N \l_tmpa_tl
4654 \clist_map_inline:Nn \l_tmpa_clist {
4655   \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
4656     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
4657   }
4658 }
4659 \tl_if_empty:NTF \l_tmpa_tl {
4660   \__stex_statements_sdefinition_start:
4661 }{
4662   \l_tmpa_tl
4663 }
4664 }
4665 \stex_ref_new_doc_target:n \sdefinitionid
4666 \stex_smsmode_do:
4667 }{
4668   \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
4669   \stex_if_smsmode:F {
4670     \clist_set:No \l_tmpa_clist \sdefinitiontype
4671     \tl_clear:N \l_tmpa_tl
4672     \clist_map_inline:Nn \l_tmpa_clist {
4673       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
4674         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
4675       }
4676     }
4677     \tl_if_empty:NTF \l_tmpa_tl {
4678       \__stex_statements_sdefinition_end:
4679     }{
4680       \l_tmpa_tl
4681     }
4682     \end{stex_annotate_env}
4683   }
4684 }

```

\stexpatchdefinition

```

4685 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
4686   \par\noindent\titllemph{Definition\tl_if_empty:NF \sdefinitiontitle {
4687     ~(\sdefinitiontitle)
4688   }~}
4689 }

```

```

4690 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\par\medskip}
4691
4692 \newcommand\stexpatchdefinition[3] [] {
4693   \str_set:Nx \l_tmpa_str{ #1 }
4694   \str_if_empty:NTF \l_tmpa_str {
4695     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
4696     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
4697   }{
4698     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
4699     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
4700   }
4701 }

```

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

\inlinedef inline:

```

4702 \keys_define:nn {stex / inlinedef }{
4703   type      .str_set_x:N = \sdefinitiontype,
4704   id        .str_set_x:N = \sdefinitionid,
4705   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4706   name      .str_set_x:N = \sdefinitionname
4707 }
4708 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
4709   \str_clear:N \sdefinitiontype
4710   \str_clear:N \sdefinitionid
4711   \str_clear:N \sdefinitionname
4712   \clist_clear:N \l__stex_statements_sdefinition_for_clist
4713   \keys_set:nn { stex / inlinedef }{ #1 }
4714 }
4715 \NewDocumentCommand \inlinedef { 0{} m } {
4716   \beginngroup
4717   \__stex_statements_inlinedef_args:n{ #1 }
4718   \stex_reactivate_macro:N \definiendum
4719   \stex_reactivate_macro:N \definame
4720   \stex_reactivate_macro:N \Definame
4721   \stex_reactivate_macro:N \premise
4722   \stex_reactivate_macro:N \definiens
4723   \stex_ref_new_doc_target:n \sdefinitionid
4724   \stex_if_smsmode:TF{
4725     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
4726   }{
4727     \seq_clear:N \l_tmpa_seq
4728     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4729       \tl_if_empty:nF{ ##1 }{
4730         \stex_get_symbol:n { ##1 }
4731         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4732           \l_stex_get_symbol_uri_str
4733         }
4734       }
4735     }
4736     \exp_args:Nnx
4737     \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
4738       \str_if_empty:NF \sdefinitiontype {
4739         \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{

```

```

4740     }
4741     #2
4742     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{ }\sdefinitionname } }
4743   }
4744 }
4745 \endgroup
4746 \stex_smsmode_do:
4747 }

```

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

## 32.2 Assertions

**sassertion**

```

4748
4749 \keys_define:nn {stex / sassertion }{
4750   type      .str_set_x:N = \sassertiontype,
4751   id        .str_set_x:N = \sassertionid,
4752   title     .tl_set:N    = \sassertiontitle ,
4753   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4754   name      .str_set_x:N = \sassertionname
4755 }
4756 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
4757   \str_clear:N \sassertiontype
4758   \str_clear:N \sassertionid
4759   \str_clear:N \sassertionname
4760   \clist_clear:N \l__stex_statements_sassertion_for_clist
4761   \tl_clear:N \sassertiontitle
4762   \keys_set:nn { stex / sassertion }{ #1 }
4763 }
4764
4765 %\tl_new:N \g__stex_statements_aftergroup_tl
4766
4767 \NewDocumentEnvironment{sassertion}{0{}}{
4768   \__stex_statements_sassertion_args:n{ #1 }
4769   \stex_reactivate_macro:N \premise
4770   \stex_reactivate_macro:N \conclusion
4771   \stex_if_smsmode:F {
4772     \seq_clear:N \l_tmpa_seq
4773     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4774       \tl_if_empty:nF{ ##1 }{
4775         \stex_get_symbol:n { ##1 }
4776         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4777           \l_stex_get_symbol_uri_str
4778         }
4779       }
4780     }
4781     \exp_args:Nnnx
4782     \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}
4783     \str_if_empty:NF \sassertiontype {
4784       \stex_annotate_invisible:nnn{type}{\sassertiontype}{ }
4785     }
4786     \clist_set:Nn \l_tmpa_clist \sassertiontype

```

```

4787 \tl_clear:N \l_tmpa_tl
4788 \clist_map_inline:Nn \l_tmpa_clist {
4789   \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
4790     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
4791   }
4792 }
4793 \tl_if_empty:NTF \l_tmpa_tl {
4794   \__stex_statements_sassertion_start:
4795 }{
4796   \l_tmpa_tl
4797 }
4798 }
4799 \str_if_empty:NTF \sassertionid {
4800   \str_if_empty:NF \sassertionname {
4801     \stex_ref_new_doc_target:n {}
4802   }
4803 } {
4804   \stex_ref_new_doc_target:n \sassertionid
4805 }
4806 \stex_smsmode_do:
4807 ){
4808   \str_if_empty:NF \sassertionname {
4809     \stex_symdecl_do:nn{ }\sassertionname}
4810   \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4811 }
4812 \stex_if_smsmode:F {
4813   \clist_set:Nn \l_tmpa_clist \sassertiontype
4814   \tl_clear:N \l_tmpa_tl
4815   \clist_map_inline:Nn \l_tmpa_clist {
4816     \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
4817       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
4818     }
4819   }
4820   \tl_if_empty:NTF \l_tmpa_tl {
4821     \__stex_statements_sassertion_end:
4822   }{
4823     \l_tmpa_tl
4824   }
4825   \end{stex_annotate_env}
4826 }
4827 }

```

\stexpatchassertion

```

4828
4829 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
4830   \par\noindent\titllemph{Assertion~\tl_if_empty:NF \sassertiontitle {
4831     (\sassertiontitle)
4832   }~}
4833 }
4834 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
4835
4836 \newcommand\stexpatchassertion[3] [] {
4837   \str_set:Nx \l_tmpa_str{ #1 }
4838   \str_if_empty:NTF \l_tmpa_str {

```



```

4839     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
4840     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
4841   }{
4842     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
4843     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
4844   }
4845 }

```

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

\inlineass inline:

```

4846 \keys_define:nn {stex / inlineass }{
4847   type      .str_set_x:N = \sassertiontype,
4848   id        .str_set_x:N = \sassertionid,
4849   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4850   name      .str_set_x:N = \sassertionname
4851 }
4852 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
4853   \str_clear:N \sassertiontype
4854   \str_clear:N \sassertionid
4855   \str_clear:N \sassertionname
4856   \clist_clear:N \l__stex_statements_sassertion_for_clist
4857   \keys_set:nn { stex / inlineass }{ #1 }
4858 }
4859 \NewDocumentCommand \inlineass { 0{} m } {
4860   \begin_group
4861     \stex_reactivate_macro:N \premise
4862     \stex_reactivate_macro:N \conclusion
4863     \__stex_statements_inlineass_args:n{ #1 }
4864     \str_if_empty:NTF \sassertionid {
4865       \str_if_empty:NF \sassertionname {
4866         \stex_ref_new_doc_target:n { }
4867       }
4868     } {
4869       \stex_ref_new_doc_target:n \sassertionid
4870     }
4871
4872     \stex_if_smsmode:TF{
4873       \str_if_empty:NF \sassertionname {
4874         \stex_symdecl_do:nn{}{\sassertionname}
4875         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4876       }
4877     }{
4878       \seq_clear:N \l_tmpa_seq
4879       \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4880         \tl_if_empty:nF{ ##1 }{
4881           \stex_get_symbol:n { ##1 }
4882           \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4883             \l_stex_get_symbol_uri_str
4884           }
4885         }
4886       }
4887       \exp_args:Nnx
4888       \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{

```

```

4889     \str_if_empty:NF \sassertiontype {
4890       \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
4891     }
4892     #2
4893     \str_if_empty:NF \sassertionname {
4894       \stex_symdecl_do:nn{}{\sassertionname}
4895       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4896     }
4897   }
4898 }
4899 \endgroup
4900 \stex_smsmode_do:
4901 }

```

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

## 32.3 Examples

`sexample`

```

4902
4903 \keys_define:nn {stex / sexample }{
4904   type      .str_set_x:N = \exampletype,
4905   id        .str_set_x:N = \sexampleid,
4906   title     .tl_set:N   = \sexampletitle,
4907   for       .clist_set:N = \l__stex_statements_sexample_for_clist,
4908 }
4909 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
4910   \str_clear:N \sexampletype
4911   \str_clear:N \sexampleid
4912   \tl_clear:N \sexampletitle
4913   \clist_clear:N \l__stex_statements_sexample_for_clist
4914   \keys_set:nn { stex / sexample }{ #1 }
4915 }
4916
4917 \NewDocumentEnvironment{sexample}{0{}}{
4918   \__stex_statements_sexample_args:n{ #1 }
4919   \stex_reactivate_macro:N \premise
4920   \stex_reactivate_macro:N \conclusion
4921   \stex_if_smsmode:F {
4922     \seq_clear:N \l_tmpa_seq
4923     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4924       \tl_if_empty:nF{ ##1 }{
4925         \stex_get_symbol:n { ##1 }
4926         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4927           \l_stex_get_symbol_uri_str
4928         }
4929       }
4930     }
4931     \exp_args:Nnnx
4932     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
4933     \str_if_empty:NF \sexampletype {
4934       \stex_annotate_invisible:nnn{type}{\sexampletype}{}
4935     }

```

```

4936 \clist_set:No \l_tmpa_clist \sexamplotype
4937 \tl_clear:N \l_tmpa_tl
4938 \clist_map_inline:Nn \l_tmpa_clist {
4939   \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
4940     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
4941   }
4942 }
4943 \tl_if_empty:NTF \l_tmpa_tl {
4944   \__stex_statements_sexample_start:
4945 }{
4946   \l_tmpa_tl
4947 }
4948 }
4949 \str_if_empty:NF \sexampleid {
4950   \stex_ref_new_doc_target:n \sexampleid
4951 }
4952 \stex_smsmode_do:
4953 }{
4954   \str_if_empty:NF \sexamplename { \stex_symdecl_do:nn{}{\sexamplename} }
4955   \stex_if_smsmode:F {
4956     \clist_set:No \l_tmpa_clist \sexamplotype
4957     \tl_clear:N \l_tmpa_tl
4958     \clist_map_inline:Nn \l_tmpa_clist {
4959       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
4960         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
4961       }
4962     }
4963     \tl_if_empty:NTF \l_tmpa_tl {
4964       \__stex_statements_sexample_end:
4965     }{
4966       \l_tmpa_tl
4967     }
4968     \end{stex_annotate_env}
4969   }
4970 }

```

\stexpatchexample

```

4971 \cs_new_protected:Nn \__stex_statements_sexample_start: {
4972   \par\noindent\titllemph{Example~\tl_if_empty:NF \sexamplename {
4973     (\sexamplename)
4974   }~}
4975 }
4976 }
4977 \cs_new_protected:Nn \__stex_statements_sexample_end: { \par\medskip}
4978
4979 \newcommand\stexpatchexample[3]{} {
4980   \str_set:Nx \l_tmpa_str{ #1 }
4981   \str_if_empty:NTF \l_tmpa_str {
4982     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
4983     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
4984   }{
4985     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
4986     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
4987   }

```

4988 }

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

`\inlineex` inline:

```

4989 \keys_define:nn {stex / inlineex }{
4990   type      .str_set_x:N = \sexamplotype,
4991   id        .str_set_x:N = \sexampleid,
4992   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
4993   name      .str_set_x:N = \sexamplename
4994 }
4995 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
4996   \str_clear:N \sexamplotype
4997   \str_clear:N \sexampleid
4998   \str_clear:N \sexamplename
4999   \clist_clear:N \l__stex_statements_sexample_for_clist
5000   \keys_set:nn { stex / inlineex }{ #1 }
5001 }
5002 \NewDocumentCommand \inlineex { 0{ } m } {
5003   \begingroup
5004   \stex_reactivate_macro:N \premise
5005   \stex_reactivate_macro:N \conclusion
5006   \__stex_statements_inlineex_args:n{ #1 }
5007   \str_if_empty:NF \sexampleid {
5008     \stex_ref_new_doc_target:n \sexampleid
5009   }
5010   \stex_if_smsmode:TF{
5011     \str_if_empty:NF \sexamplename { \stex_symdecl_do:nn{}{\sexamplename} }
5012   }{
5013     \seq_clear:N \l_tmpa_seq
5014     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5015       \tl_if_empty:nF{ ##1 }{
5016         \stex_get_symbol:n { ##1 }
5017         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5018           \l_stex_get_symbol_uri_str
5019         }
5020       }
5021     }
5022     \exp_args:Nnx
5023     \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpa_seq {},}{
5024       \str_if_empty:NF \sexamplotype {
5025         \stex_annotate_invisible:nnn{type}{\sexamplotype}{}
5026       }
5027       #2
5028       \str_if_empty:NF \sexamplename { \stex_symdecl_do:nn{}{\sexamplename} }
5029     }
5030   }
5031   \endgroup
5032   \stex_smsmode_do:
5033 }

```

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

## 32.4 Logical Paragraphs

sparagraph

```

5034 \keys_define:nn { stex / sparagraph } {
5035   id       .str_set_x:N = \sparagraphid ,
5036   title    .tl_set:N    = \l_stex_sparagraph_title_tl ,
5037   type     .str_set_x:N = \sparagraphtype ,
5038   for      .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5039   from     .tl_set:N    = \sparagraphfrom ,
5040   to       .tl_set:N    = \sparagraphto ,
5041   start    .tl_set:N    = \l_stex_sparagraph_start_tl ,
5042   name     .str_set:N   = \sparagraphname
5043 }
5044
5045 \cs_new_protected:Nn \stex_sparagraph_args:n {
5046   \tl_clear:N \l_stex_sparagraph_title_tl
5047   \tl_clear:N \sparagraphfrom
5048   \tl_clear:N \sparagraphto
5049   \tl_clear:N \l_stex_sparagraph_start_tl
5050   \str_clear:N \sparagraphid
5051   \str_clear:N \sparagraphtype
5052   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5053   \str_clear:N \sparagraphname
5054   \keys_set:nn { stex / sparagraph } { #1 }
5055 }
5056 \newif\if@in@omtext\@in@omtextfalse
5057
5058 \NewDocumentEnvironment {sparagraph} { 0{} } {
5059   \stex_sparagraph_args:n { #1 }
5060   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5061     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
5062   }{
5063     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
5064   }
5065   \@in@omtexttrue
5066   \stex_if_smsmode:F {
5067     \seq_clear:N \l_tmpa_seq
5068     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5069       \tl_if_empty:NF{ ##1 }{
5070         \stex_get_symbol:n { ##1 }
5071         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5072           \l_stex_get_symbol_uri_str
5073         }
5074       }
5075     }
5076     \exp_args:Nnnx
5077     \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
5078     \str_if_empty:NF \sparagraphtype {
5079       \stex_annotate_invisible:nnn{type}{\sparagraphtype}{ }
5080     }
5081     \str_if_empty:NF \sparagraphfrom {
5082       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{ }
5083     }
5084     \str_if_empty:NF \sparagraphto {

```

```

5085     \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5086   }
5087   \clist_set:No \l_tmpa_clist \sparagraphtype
5088   \tl_clear:N \l_tmpa_tl
5089   \clist_map_inline:Nn \sparagraphtype {
5090     \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
5091       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
5092     }
5093   }
5094   \tl_if_empty:NTF \l_tmpa_tl {
5095     \__stex_statements_sparagraph_start:
5096   }{
5097     \l_tmpa_tl
5098   }
5099 }
5100 \clist_set:No \l_tmpa_clist \sparagraphtype
5101 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5102 {
5103   \stex_reactivate_macro:N \definiendum
5104   \stex_reactivate_macro:N \definame
5105   \stex_reactivate_macro:N \Definame
5106   \stex_reactivate_macro:N \premise
5107   \stex_reactivate_macro:N \definiens
5108 }
5109 \str_if_empty:NTF \sparagraphid {
5110   \str_if_empty:NTF \sparagraphname {
5111     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5112       \stex_ref_new_doc_target:n {}
5113     }
5114   } {
5115     \stex_ref_new_doc_target:n {}
5116   }
5117 } {
5118   \stex_ref_new_doc_target:n \sparagraphid
5119 }
5120 \exp_args:NNx
5121 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5122   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5123     \tl_if_empty:nF{ ##1 }{
5124       \stex_get_symbol:n { ##1 }
5125       \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5126     }
5127   }
5128 }
5129 \stex_smsmode_do:
5130 \ignorespacesandpars
5131 }{
5132   \str_if_empty:NF \sparagraphname {
5133     \stex_symdecl_do:nn{}{\sparagraphname}
5134     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5135   }
5136   \stex_if_smsmode:F {
5137     \clist_set:No \l_tmpa_clist \sparagraphtype
5138     \tl_clear:N \l_tmpa_tl

```

```

5139 \clist_map_inline:Nn \l_tmpa_clist {
5140   \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
5141     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
5142   }
5143 }
5144 \tl_if_empty:NTF \l_tmpa_tl {
5145   \__stex_statements_sparagraph_end:
5146 }{
5147   \l_tmpa_tl
5148 }
5149 \end{stex_annotate_env}
5150 }
5151 }

```

# \stexpatchparagraph

```

5152
5153 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
5154   \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5155     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
5156       \titleemph{\l_stex_sparagraph_title_tl}:~
5157     }
5158   }{
5159     \titleemph{\l_stex_sparagraph_start_tl}~
5160   }
5161 }
5162 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
5163
5164 \newcommand\stexpatchparagraph[3]{} {
5165   \str_set:Nx \l_tmpa_str{ #1 }
5166   \str_if_empty:NTF \l_tmpa_str {
5167     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
5168     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
5169   }{
5170     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2 }
5171     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
5172   }
5173 }
5174
5175 \keys_define:nn { stex / inlinepara } {
5176   id      .str_set_x:N = \sparagraphid ,
5177   type    .str_set_x:N = \sparagraphtype ,
5178   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5179   from    .tl_set:N    = \sparagraphfrom ,
5180   to      .tl_set:N    = \sparagraphto ,
5181   name    .str_set:N    = \sparagraphname
5182 }
5183 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
5184   \tl_clear:N \sparagraphfrom
5185   \tl_clear:N \sparagraphto
5186   \str_clear:N \sparagraphid
5187   \str_clear:N \sparagraphtype
5188   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5189   \str_clear:N \sparagraphname
5190   \keys_set:nn { stex / inlinepara }{ #1 }

```

```

5191 }
5192 \NewDocumentCommand \inlinepara { 0{} m } {
5193   \begingroup
5194   \_stex_statements_inlinepara_args:n{ #1 }
5195   \clist_set:No \l_tmpa_clist \sparagraphtype
5196   \str_if_empty:NTF \sparaagraphid {
5197     \str_if_empty:NTF \sparaagraphname {
5198       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5199         \stex_ref_new_doc_target:n {}
5200       }
5201     } {
5202       \stex_ref_new_doc_target:n {}
5203     }
5204   } {
5205     \stex_ref_new_doc_target:n \sparaagraphid
5206   }
5207   \stex_if_smsmode:TF{
5208     \str_if_empty:NF \sparaagraphname {
5209       \stex_symdecl_do:nn{}{\sparaagraphname}
5210       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparaagraphname}
5211     }
5212   }{
5213     \seq_clear:N \l_tmpa_seq
5214     \clist_map_inline:Nn \l__stex_statements_sparaagraph_for_clist {
5215       \tl_if_empty:nF{ ##1 }{
5216         \stex_get_symbol:n { ##1 }
5217         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5218           \l_stex_get_symbol_uri_str
5219         }
5220       }
5221     }
5222     \exp_args:Nnx
5223     \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {},,}{
5224       \str_if_empty:NF \sparaagraphtype {
5225         \stex_annotate_invisible:nnn{type}{\sparaagraphtype}{}
5226       }
5227       \str_if_empty:NF \sparaagraphfrom {
5228         \stex_annotate_invisible:nnn{from}{\sparaagraphfrom}{}
5229       }
5230       \str_if_empty:NF \sparaagraphto {
5231         \stex_annotate_invisible:nnn{to}{\sparaagraphto}{}
5232       }
5233       \str_if_empty:NF \sparaagraphname {
5234         \stex_symdecl_do:nn{}{\sparaagraphname}
5235         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparaagraphname}
5236       }
5237       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5238         \clist_map_inline:Nn \l_tmpa_seq {
5239           \stex_ref_new_sym_target:n {##1}
5240         }
5241       }
5242     } #2
5243   }
5244 }

```



```

5245 \endgroup
5246 \stex_smsmode_do:
5247 }
5248
(End definition for \stexpatchparagraph. This function is documented on page ??.)
5249 </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>11</sup>

```
5250 <*package>
5251 <@@=stex_sproof>
5252
5253 %%%%%%%%%%% sproof.dtx %%%%%%%%%%%
5254
```

### 33.2 Proofs

We first define some keys for the proof environment.

```
5255 \keys_define:nn { stex / spf } {
5256   id          .str_set_x:N = \spfid,
5257   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,
5258   from        .tl_set:N    = \l__stex_sproof_spf_from_tl ,
5259   proofend    .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
5260   type        .str_set_x:N = \spftype,
5261   title       .tl_set:N    = \spftitle,
5262   continues   .tl_set:N    = \l__stex_sproof_spf_continues_tl,
5263   functions   .tl_set:N    = \l__stex_sproof_spf_functions_tl,
5264   method      .tl_set:N    = \l__stex_sproof_spf_method_tl
5265 }
5266 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
5267   \str_clear:N \spfid
5268   \tl_clear:N \l__stex_sproof_spf_for_tl
5269   \tl_clear:N \l__stex_sproof_spf_from_tl
5270   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
5271   \str_clear:N \spftype
5272   \tl_clear:N \spftitle
5273   \tl_clear:N \l__stex_sproof_spf_continues_tl
5274   \tl_clear:N \l__stex_sproof_spf_functions_tl
```

---

<sup>11</sup>EdNOTE: need an implementation for L<sup>A</sup>T<sub>E</sub>XML

```

5275 \tl_clear:N \l__stex_sproof_spf_method_tl
5276 \bool_set_false:N \l__stex_sproof_inc_counter_bool
5277 \keys_set:nn { stex / spf }{ #1 }
5278 }

```

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

```

5279 \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>6</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.

```

5280 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
5281 \cs_new_protected:Npn \sproofnumber {
5282   \int_set:Nn \l_tmpa_int {1}
5283   \bool_while_do:nn {
5284     \int_compare_p:nNn {
5285       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5286     } > 0
5287   }{
5288     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
5289     \int_incr:N \l_tmpa_int
5290   }
5291 }
5292 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
5293   \int_set:Nn \l_tmpa_int {1}
5294   \bool_while_do:nn {
5295     \int_compare_p:nNn {
5296       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5297     } > 0
5298   }{
5299     \int_incr:N \l_tmpa_int
5300   }
5301   \int_compare:nNnF \l_tmpa_int = 1 {
5302     \int_decr:N \l_tmpa_int
5303   }
5304   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
5305     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1

```

---

<sup>6</sup>This gets the labeling right but only works 8 levels deep



```

5350 }
5351 \clist_if_in:NnT \l_tmpa_clist {finnish}{
5352   \input{sproof-finnish.ldf}
5353 }
5354 \clist_if_in:NnT \l_tmpa_clist {french}{
5355   \input{sproof-french.ldf}
5356 }
5357 \clist_if_in:NnT \l_tmpa_clist {russian}{
5358   \input{sproof-russian.ldf}
5359 }
5360 \makeatother
5361 }{}
5362 }

```

spfsketch

```

5363 \newcommand\spfsketch[2] [] {
5364   \beginingroup
5365   \let \premise \stex_proof_premise:
5366   \__stex_sproof_spf_args:n{#1}
5367   \stex_if_smsmode:TF {
5368     \str_if_empty:NF \spfid {
5369       \stex_ref_new_doc_target:n \spfid
5370     }
5371   }{
5372     \seq_clear:N \l_tmpa_seq
5373     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5374       \tl_if_empty:nF{ ##1 }{
5375         \stex_get_symbol:n { ##1 }
5376         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5377           \l_stex_get_symbol_uri_str
5378         }
5379       }
5380     }
5381     \exp_args:Nnx
5382     \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
5383       \str_if_empty:NF \spftype {
5384         \stex_annotate_invisible:nnn{type}{\spftype}{-}
5385       }
5386       \clist_set:No \l_tmpa_clist \spftype
5387       \tl_set:Nn \l_tmpa_tl {
5388         \titleemph{
5389           \tl_if_empty:NTF \spftitle {
5390             \spf@proofsketch@kw
5391           }{
5392             \spftitle
5393           }
5394         }::~
5395       }
5396       \clist_map_inline:Nn \l_tmpa_clist {
5397         \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5398           \tl_clear:N \l_tmpa_tl
5399         }
5400       }
5401       \str_if_empty:NF \spfid {

```

```

5402         \stex_ref_new_doc_target:n \spfid
5403     }
5404     \l_tmpa_tl #2 \sproofend
5405 }
5406 }
5407 \endgroup
5408 \stex_smsmode_do:
5409 }
5410

```

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

**spfeq** This is very similar to \spfsketch, but uses a computation array<sup>1213</sup>

```

5411 \newenvironment{spfeq}[2][]{
5412   \__stex_sproof_spf_args:n{#1}
5413   \let \premise \stex_proof_premise:
5414   \stex_if_smsmode:TF {
5415     \str_if_empty:NF \spfid {
5416       \stex_ref_new_doc_target:n \spfid
5417     }
5418   }{
5419     \seq_clear:N \l_tmpa_seq
5420     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5421       \tl_if_empty:NF{ ##1 }{
5422         \stex_get_symbol:n { ##1 }
5423         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5424           \l_stex_get_symbol_uri_str
5425         }
5426       }
5427     }
5428     \exp_args:Nnnx
5429     \begin{stex_annotate_env}{spfeq}{\seq_use:Nn \l_tmpa_seq {,}}
5430     \str_if_empty:NF \spftype {
5431       \stex_annotate_invisible:nnn{type}{\spftype}{ }
5432     }
5433
5434     \clist_set:No \l_tmpa_clist \spftype
5435     \tl_clear:N \l_tmpa_tl
5436     \clist_map_inline:Nn \l_tmpa_clist {
5437       \tl_if_exist:cT {__stex_sproof_spfeq_##1_start:}{
5438         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_start:}}
5439       }
5440       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5441         \tl_set:Nn \l_tmpa_tl {\use:n{}}
5442       }
5443     }
5444     \tl_if_empty:NTF \l_tmpa_tl {
5445       \__stex_sproof_spfeq_start:
5446     }{
5447       \l_tmpa_tl
5448     }{-#2}

```

<sup>12</sup>EDNOTE: This should really be more like a tabular with an ensuremath in it. or invoke text on the last column

<sup>13</sup>EDNOTE: document above

```

5449 \str_if_empty:NF \spfid {
5450 \stex_ref_new_doc_target:n \spfid
5451 }
5452 \begin{displaymath}\begin{array}{rcll}
5453 }
5454 \stex_smsmode_do:
5455 }{
5456 \stex_if_smsmode:F {
5457 \end{array}\end{displaymath}
5458 \clist_set:No \l_tmpa_clist \spftype
5459 \tl_clear:N \l_tmpa_tl
5460 \clist_map_inline:Nn \l_tmpa_clist {
5461 \tl_if_exist:cT {__stex_sproof_spfeq_##1_end:}{
5462 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_end:}}
5463 }
5464 }
5465 \tl_if_empty:NTF \l_tmpa_tl {
5466 \__stex_sproof_spfeq_end:
5467 }{
5468 \l_tmpa_tl
5469 }
5470 \end{stex_annotate_env}
5471 }
5472 }
5473
5474 \cs_new_protected:Nn \__stex_sproof_spfeq_start: {
5475 \titleemph{
5476 \tl_if_empty:NTF \spftitle {
5477 \spf@proof@kw
5478 }{
5479 \spftitle
5480 }
5481 }:
5482 }
5483 \cs_new_protected:Nn \__stex_sproof_spfeq_end: {\sproofend}
5484
5485 \newcommand\stexpatchspfeq[3] [] {
5486 \str_set:Nx \l_tmpa_str{ #1 }
5487 \str_if_empty:NTF \l_tmpa_str {
5488 \tl_set:Nn \__stex_sproof_spfeq_start: { #2 }
5489 \tl_set:Nn \__stex_sproof_spfeq_end: { #3 }
5490 }{
5491 \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_start:\endcsname{ #2 }
5492 \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_end:\endcsname{ #3 }
5493 }
5494 }
5495

```

(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.

```

5496 \newenvironment{sproof}[2] []{

```

```

5497 \let \premise \stex_proof_premise:
5498 \intarray_gzero:N \l__stex_sproof_counter_intarray
5499 \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
5500 \__stex_sproof_spf_args:n{#1}
5501 \stex_if_smsmode:TF {
5502   \str_if_empty:NF \spfid {
5503     \stex_ref_new_doc_target:n \spfid
5504   }
5505 }{
5506   \seq_clear:N \l_tmpa_seq
5507   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5508     \tl_if_empty:NF{ ##1 }{
5509       \stex_get_symbol:n { ##1 }
5510       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5511         \l_stex_get_symbol_uri_str
5512       }
5513     }
5514   }
5515   \exp_args:Nnnx
5516   \begin{stex_annotate_env}{sproof}{\seq_use:Nn \l_tmpa_seq {},}}
5517   \str_if_empty:NF \spftype {
5518     \stex_annotate_invisible:nnn{type}{\spftype}{}
5519   }
5520
5521   \clist_set:No \l_tmpa_clist \spftype
5522   \tl_clear:N \l_tmpa_tl
5523   \clist_map_inline:Nn \l_tmpa_clist {
5524     \tl_if_exist:cT {__stex_sproof_sproof_##1_start:}{
5525       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_start:}}
5526     }
5527     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5528       \tl_set:Nn \l_tmpa_tl {\use:n{}}
5529     }
5530   }
5531   \tl_if_empty:NTF \l_tmpa_tl {
5532     \__stex_sproof_sproof_start:
5533   }{
5534     \l_tmpa_tl
5535   }{~#2}
5536   \str_if_empty:NF \spfid {
5537     \stex_ref_new_doc_target:n \spfid
5538   }
5539   \begin{description}
5540 }
5541 \stex_smsmode_do:
5542 }{
5543   \stex_if_smsmode:F{
5544     \end{description}
5545     \clist_set:No \l_tmpa_clist \spftype
5546     \tl_clear:N \l_tmpa_tl
5547     \clist_map_inline:Nn \l_tmpa_clist {
5548       \tl_if_exist:cT {__stex_sproof_sproof_##1_end:}{
5549         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_end:}}
5550       }

```



```

5551 }
5552 \tl_if_empty:NTF \l_tmpa_tl {
5553   \__stex_sproof_sproof_end:
5554 }{
5555   \l_tmpa_tl
5556 }
5557 \end{stex_annotate_env}
5558 }
5559 }
5560
5561 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
5562   \par\noindent\titleemph{
5563     \tl_if_empty:NTF \spftype {
5564       \spf@proof@kw
5565     }{
5566       \spftype
5567     }
5568   }:
5569 }
5570 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
5571
5572 \newcommand\stexpatchsproof[3] [] {
5573   \str_set:Nx \l_tmpa_str{ #1 }
5574   \str_if_empty:NTF \l_tmpa_str {
5575     \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
5576     \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
5577   }{
5578     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
5579     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
5580   }
5581 }

```

\spfidea

```

5582 \newcommand\spfidea[2] []{
5583   \__stex_sproof_spf_args:n{#1}
5584   \titleemph{
5585     \tl_if_empty:NTF \spftype {Proof~Idea}{
5586       \spftype
5587     }:
5588   }~#2
5589   \sproofend
5590 }

```

(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

```

5591 \newenvironment{spfstep}[1] []{
5592   \__stex_sproof_spf_args:n{#1}
5593   \stex_if_smsmode:TF {

```

```

5594 \str_if_empty:NF \spfid {
5595   \stex_ref_new_doc_target:n \spfid
5596 }
5597 }{
5598   \@in@omtexttrue
5599   \seq_clear:N \l_tmpa_seq
5600   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5601     \tl_if_empty:nF{ ##1 }{
5602       \stex_get_symbol:n { ##1 }
5603       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5604         \l_stex_get_symbol_uri_str
5605       }
5606     }
5607   }
5608   \exp_args:Nnnx
5609   \begin{stex_annotate_env}{spfstep}{\seq_use:Nn \l_tmpa_seq {,}}
5610   \str_if_empty:NF \spftype {
5611     \stex_annotate_invisible:nnn{type}{\spftype}{}
5612   }
5613   \clist_set:No \l_tmpa_clist \spftype
5614   \tl_set:Nn \l_tmpa_tl {
5615     \item[\sproofnumber]
5616     \bool_set_true:N \l__stex_sproof_inc_counter_bool
5617   }
5618   \clist_map_inline:Nn \l_tmpa_clist {
5619     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5620       \tl_clear:N \l_tmpa_tl
5621     }
5622   }
5623   \l_tmpa_tl
5624   \tl_if_empty:NF \spftitle {
5625     {(\titleemph{\spftitle})\enspace}
5626   }
5627   \str_if_empty:NF \spfid {
5628     \stex_ref_new_doc_target:n \spfid
5629   }
5630 }
5631 \stex_smsmode_do:
5632 \ignorespacesandpars
5633 }{
5634   \bool_if:NT \l__stex_sproof_inc_counter_bool {
5635     \__stex_sproof_inc_counter:
5636   }
5637   \stex_if_smsmode:F {
5638     \end{stex_annotate_env}
5639   }
5640 }

```

sproofcomment

```

5641 \newenvironment{sproofcomment}[1][]{
5642   \__stex_sproof_spf_args:n{#1}
5643   \clist_set:No \l_tmpa_clist \spftype
5644   \tl_set:Nn \l_tmpa_tl {
5645     \item[\sproofnumber]

```

```

5646 \bool_set_true:N \l__stex_sproof_inc_counter_bool
5647 }
5648 \clist_map_inline:Nn \l_tmpa_clist {
5649 \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5650 \tl_clear:N \l_tmpa_tl
5651 }
5652 }
5653 \l_tmpa_tl
5654 }{
5655 \bool_if:NT \l__stex_sproof_inc_counter_bool {
5656 \__stex_sproof_inc_counter:
5657 }
5658 }

```

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.

```

5659 \newenvironment{subproof}[2][]{
5660 \__stex_sproof_spf_args:n{#1}
5661 \stex_if_smsmode:TF{
5662 \str_if_empty:NF \spfid {
5663 \stex_ref_new_doc_target:n \spfid
5664 }
5665 }{
5666 \seq_clear:N \l_tmpa_seq
5667 \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5668 \tl_if_empty:nF{ ##1 }{
5669 \stex_get_symbol:n { ##1 }
5670 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5671 \l_stex_get_symbol_uri_str
5672 }
5673 }
5674 }
5675 \exp_args:Nnnx
5676 \begin{stex_annotate_env}{subproof}{\seq_use:Nn \l_tmpa_seq {,}}
5677 \str_if_empty:NF \spftype {
5678 \stex_annotate_invisible:nnn{type}{\spftype}{}}
5679 }
5680
5681 \clist_set:No \l_tmpa_clist \spftype
5682 \tl_set:Nn \l_tmpa_tl {
5683 \item[\sproofnumber]
5684 \bool_set_true:N \l__stex_sproof_inc_counter_bool
5685 }
5686 \clist_map_inline:Nn \l_tmpa_clist {
5687 \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5688 \tl_clear:N \l_tmpa_tl
5689 }
5690 }
5691 \l_tmpa_tl
5692 \tl_if_empty:NF \spftitle {
5693 {(\titleemph{\spftitle})\enspace}
5694 }

```

```

5695     {~#2}
5696     \str_if_empty:NF \spfid {
5697       \stex_ref_new_doc_target:n \spfid
5698     }
5699   }
5700   \__stex_sproof_add_counter:
5701   \stex_smsmode_do:
5702 }{
5703   \__stex_sproof_remove_counter:
5704   \bool_if:NT \l__stex_sproof_inc_counter_bool {
5705     \__stex_sproof_inc_counter:
5706   }
5707   \stex_if_smsmode:F{
5708     \end{stex_annotate_env}
5709   }
5710 }

```

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

```

5711 \newenvironment{spfcases}[2][]{
5712   \tl_if_empty:nTF{#1}{
5713     \begin{subproof}[method=by-cases]{#2}
5714   }{
5715     \begin{subproof}[#1,method=by-cases]{#2}
5716   }
5717 }{
5718   \end{subproof}
5719 }

```

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

```

5720 \newenvironment{spfcase}[2][]{
5721   \__stex_sproof_spf_args:n{#1}
5722   \stex_if_smsmode:TF {
5723     \str_if_empty:NF \spfid {
5724       \stex_ref_new_doc_target:n \spfid
5725     }
5726   }{
5727     \seq_clear:N \l_tmpa_seq
5728     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5729       \tl_if_empty:nF{ ##1 }{
5730         \stex_get_symbol:n { ##1 }
5731         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5732           \l_stex_get_symbol_uri_str
5733         }
5734       }
5735     }
5736     \exp_args:Nnnx
5737     \begin{stex_annotate_env}{spfcase}{\seq_use:Nn \l_tmpa_seq {,}}
5738     \str_if_empty:NF \spftype {
5739       \stex_annotate_invisible:nnn{type}{\spftype}{}}
5740   }
5741   \clist_set:Nn \l_tmpa_clist \spftype
5742   \tl_set:Nn \l_tmpa_tl {
5743     \item[\sproofnumber]

```

```

5744     \bool_set_true:N \l__stex_sproof_inc_counter_bool
5745   }
5746   \clist_map_inline:Nn \l_tmpa_clist {
5747     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5748       \tl_clear:N \l_tmpa_tl
5749     }
5750   }
5751   \l_tmpa_tl
5752   \tl_if_empty:nF{#2}{
5753     \titleemph{#2}:~
5754   }
5755 }
5756 \__stex_sproof_add_counter:
5757 \stex_smsmode_do:
5758 ){
5759   \__stex_sproof_remove_counter:
5760   \bool_if:NT \l__stex_sproof_inc_counter_bool {
5761     \__stex_sproof_inc_counter:
5762   }
5763   \stex_if_smsmode:F{
5764     \clist_set:No \l_tmpa_clist \spftype
5765     \tl_set:Nn \l_tmpa_tl{\sproofend}
5766     \clist_map_inline:Nn \l_tmpa_clist {
5767       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5768         \tl_clear:N \l_tmpa_tl
5769       }
5770     }
5771     \l_tmpa_tl
5772     \end{stex_annotate_env}
5773   }
5774 }

```

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

```

5775 \newcommand\spfcasesketch[3][]{
5776   \begin{spfcase}[#1]{#2}#3\end{spfcase}
5777 }

```

### 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.

```

5778 \keys_define:nn { stex / just }{
5779   id      .str_set:x:N = \l__stex_sproof_just_id_str,
5780   method  .tl_set:N    = \l__stex_sproof_just_method_tl,
5781   premises .tl_set:N    = \l__stex_sproof_just_premises_tl,
5782   args     .tl_set:N    = \l__stex_sproof_just_args_tl
5783 }

```

The next three environments and macros are purely semantic, so we ignore the keyval arguments for now and only display the content.<sup>14</sup>

<sup>14</sup>EdNOTE: need to do something about the premise in draft mode.

**justification**

```
5784 \newenvironment{justification}[1] [] {}{}
```

**\premise**

```
5785 \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.

```
5786 \newcommand\justarg[2] [] {#2}
```

```
5787 \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

```
5788 <*package>
5789
5790 %%%%%%%%%%% others.dtx %%%%%%%%%%%
5791
5792 <@@=stex_others>
    Warnings and error messages
5793 % None

\MSC Math subject classifier

5794 \NewDocumentCommand \MSC {m} {
5795 % TODO
5796 }

(End definition for \MSC. This function is documented on page ??.)
    Patching tikzinput, if loaded
5797 \@ifpackageloaded{tikzinput}{
5798 \RequirePackage{stex-tikzinput}
5799 }{}
5800 </package>
```

## Chapter 35

# STEX -Metatheory Implementation

```
5801 <*package>
5802 <@@=stex_modules>
5803
5804 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
5805
5806 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
5807 \begingroup
5808 \stex_module_setup:nn{
5809   ns=\c_stex_metatheory_ns_str,
5810   meta=NONE
5811 }{Metatheory}
5812 \stex_reactivate_macro:N \symdecl
5813 \stex_reactivate_macro:N \notation
5814 \stex_reactivate_macro:N \symdef
5815 \ExplSyntaxOff
5816 \csname stex_suppress_html:n\endcsname{
5817   % is-a (a:A, a \in A, a is an A, etc.)
5818   \symdecl{isa}[args=ai]
5819   \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
5820   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
5821   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
5822
5823   % bind (\forall, \Pi, \lambda etc.)
5824   \symdecl{bind}[args=Bi]
5825   \notation{bind}[forall]{\comp\forall #1.;#2}{##1 \comp, ##2}
5826   \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
5827   \notation{bind}[depfun]{\comp( #1 \comp{}\;\to\;)}{##1 \comp, ##2}
5828
5829   % implicit bind
5830   \symdef{implicitbind}[args=Bi]{\comp\prod_{#1}#2}{##1\comp,##2}
5831
5832   % dummy variable
5833   \symdecl{dummyvar}
5834   \notation{dummyvar}[underscore]{\comp\_}
5835   \notation{dummyvar}[dot]{\comp\cdot}
```



```

5836 \notation{dummyvar}[dash]{\comp{\rm --}}
5837
5838 %fromto (function space, Hom-set, implication etc.)
5839 \symdecl{fromto}[args=ai]
5840 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
5841 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
5842
5843 % mapto (lambda etc.)
5844 \symdecl{mapto}[args=Bi]
5845 \notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
5846 \notation{mapto}[lambda]{\comp\lambda #1 \comp.; #2}{#1 \comp, #2}
5847 \notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.; #2}{#1 \comp, #2}
5848
5849 % function/operator application
5850 \symdecl{apply}[args=ia]
5851 \notation{apply}[prec=0;0x\infpres,parens]{#1 \comp( #2 \comp)}{##1 \comp, ##2}
5852 \notation{apply}[prec=0;0x\infpres,lambda]{#1 \; #2 }{##1 \; ; ##2}
5853
5854 % ‘‘type’’ of all collections (sets,classes,types,kinds)
5855 \symdecl{metacollection}
5856 \notation{metacollection}[U]{\comp{\mathcal{U}}}
5857 \notation{metacollection}[set]{\comp{\textsf{Set}}}
5858
5859 % collection of propositions/booleans/truth values
5860 \symdecl{prop}[name=proposition]
5861 \notation{prop}[prop]{\comp{\rm prop}}
5862 \notation{prop}[BOOL]{\comp{\rm BOOL}}
5863
5864 % sequences
5865 \symdecl{seqtype}[args=1]
5866 \notation{seqtype}[kleene]{#1^{\comp\ast}}
5867
5868 \symdef{sequence-index}[args=2,li,prec=nobrackets]{#{1}_#{2}}
5869 \notation{sequence-index}[ui,prec=nobrackets]{#{1}^#{2}}
5870
5871 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{,\ellipses}}{##1\comp,##2}
5872 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{,\ellipses,}#2}{##1\comp,##2}
5873 \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{,\ellipses,}#2\comp{,\ellipses,}#3}
5874
5875 % letin (‘‘let’’, local definitions, variable substitution)
5876 \symdecl{letin}[args=bii]
5877 \notation{letin}[let]{\comp{\rm let}}\;#1\comp{=}#2\; \comp{\rm in}}\;#3}
5878 \notation{letin}[subst]{#3 \comp[ #1 \comp/ #2 \comp]}
5879 \notation{letin}[frac]{#3 \comp[ \frac{#2}{#1} \comp]}
5880
5881 % structures
5882 \symdecl*{module-type}[args=1]
5883 \notation{module-type}{\mathtt{MOD} #1}
5884 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
5885 \notation{mathstruct}[angle,prec=nobrackets]{\comp\angle #1 \comp\rangle}{##1 \comp, ##2}
5886
5887 }
5888 \ExplSyntaxOn
5889 \stex_add_to_current_module:n{

```

```

5890 \let\nappa\apply
5891 \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
5892 \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
5893 \def\livar{\csname sequence-index\endcsname[li]}
5894 \def\uivar{\csname sequence-index\endcsname[ui]}
5895 \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
5896 \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
5897 \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
5898 }
5899 \__stex_modules_end_module:
5900 \endgroup
5901 </package>

```

## Chapter 36

# Tikzinput Implementation

```
5902 <*package>
5903
5904 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
5905
5906 \ProvidesExplPackage{tikzinput}{2022/02/26}{3.0.1}{tikzinput package}
5907 \RequirePackage{l3keys2e}
5908
5909 \keys_define:nn { tikzinput } {
5910   image .bool_set:N = \c_tikzinput_image_bool,
5911   image .default:n = false ,
5912   unknown .code:n = {}
5913 }
5914
5915 \ProcessKeysOptions { tikzinput }
5916
5917 \bool_if:NTF \c_tikzinput_image_bool {
5918   \RequirePackage{graphicx}
5919
5920   \providecommand\usetikzlibrary[]{}
5921   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
5922 }{
5923   \RequirePackage{tikz}
5924   \RequirePackage{standalone}
5925
5926   \newcommand \tikzinput [2] [] {
5927     \setkeys{Gin}{#1}
5928     \ifx \Gin@ewidth \Gin@exclamation
5929       \ifx \Gin@eheight \Gin@exclamation
5930         \input { #2 }
5931       \else
5932         \resizebox{!}{ \Gin@eheight }{
5933           \input { #2 }
5934         }
5935       \fi
5936     \else
5937       \ifx \Gin@eheight \Gin@exclamation
5938         \resizebox{ \Gin@ewidth }{!}{
5939           \input { #2 }
```

```

5940     }
5941     \else
5942         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
5943             \input { #2 }
5944         }
5945     \fi
5946 \fi
5947 }
5948 }
5949
5950 \newcommand \ctikzinput [2] [] {
5951     \begin{center}
5952         \tikzinput [ #1 ] { #2 }
5953     \end{center}
5954 }
5955
5956 \@ifpackageloaded{stex}{
5957     \RequirePackage{stex-tikzinput}
5958 }{}
5959
5960 </package>
5961 <*stex>
5962 \ProvidesExplPackage{stex-tikzinput}{2022/02/26}{3.0.1}{stex-tikzinput}
5963 \RequirePackage{stex}
5964 \RequirePackage{tikzinput}
5965
5966 \newcommand\mhtikzinput [2] [] {%
5967     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
5968     \stex_in_repository:nn\Gin@mhrepos{
5969         \tikzinput [ #1 ] {\mhp{##1}{#2}}
5970     }
5971 }
5972 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [ #1 ] { #2 }\end{center}}
5973 </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.

```
5974 \*cls)
5975 \@@=document_structure)
5976 \ProvidesExplClass{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure Class}
5977 \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`

```
5978 \keys_define:nn{ document-structure / pkg }{
5979   class      .str_set_x:N = \c_document_structure_class_str,
5980   minimal    .bool_set:N = \c_document_structure_minimal_bool,
5981   report     .code:n      = {
5982     \ClassWarning{document-structure}{the option 'report' is deprecated, use 'class=report',
5983     \str_set:Nn \c_document_structure_class_str {report}
5984   },
5985   book       .code:n      = {
5986     \ClassWarning{document-structure}{the option 'book' is deprecated, use 'class=book', ins
5987     \str_set:Nn \c_document_structure_class_str {book}
5988   },
5989   bookpart   .code:n      = {
5990     \ClassWarning{document-structure}{the option 'bookpart' is deprecated, use 'class=book,t
5991     \str_set:Nn \c_document_structure_class_str {book}
5992     \str_set:Nn \c_document_structure_topsect_str {chapter}
5993   },
```

```

5994 docopt      .str_set_x:N = \c_document_structure_docopt_str,
5995 unknown     .code:n      = {
5996   \PassOptionsToPackage{ \CurrentOption }{ document-structure }
5997 }
5998 }
5999 \ProcessKeysOptions{ document-structure / pkg }
6000 \str_if_empty:NT \c_document_structure_class_str {
6001   \str_set:Nn \c_document_structure_class_str {article}
6002 }
6003 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
6004   {\c_document_structure_class_str}
6005

```

### 37.3 Beefing up the document environment

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

```

6006 \RequirePackage{document-structure}
6007 \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>15</sup>

```

6008 \keys_define:nn { document-structure / document }{
6009   id .str_set_x:N = \c_document_structure_document_id_str
6010 }
6011 \let\__document_structure_orig_document=\document
6012 \renewcommand{\document}[1][]{
6013   \keys_set:nn{ document-structure / document }{ #1 }
6014   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
6015   \__document_structure_orig_document
6016 }

```

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

```

6017 }
6018 \</cls>

```

### 37.4 Implementation: document-structure Package

```

6019 <*package>
6020 \ProvidesExplPackage{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure}
6021 \RequirePackage{13keys2e}

```

### 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>15</sup>EDNOTE: faking documentkeys for now. @HANG, please implement

```

6022
6023 \keys_define:nn{ document-structure / pkg }{
6024   class      .str_set_x:N = \c_document_structure_class_str,
6025   topsect    .str_set_x:N = \c_document_structure_topsect_str,
6026   % showignores .bool_set:N = \c_document_structure_showignores_bool,
6027 }
6028 \ProcessKeysOptions{ document-structure / pkg }
6029 \str_if_empty:NT \c_document_structure_class_str {
6030   \str_set:Nn \c_document_structure_class_str {article}
6031 }
6032 \str_if_empty:NT \c_document_structure_topsect_str {
6033   \str_set:Nn \c_document_structure_topsect_str {section}
6034 }

```

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

```

6035 \RequirePackage{xspace}
6036 \RequirePackage{comment}
6037 \AddToHook{begindocument}{
6038   \ltx@ifpackageloaded{babel}{
6039     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6040     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6041       \makeatletter\input{document-structure-ngerman.ldf}\makeatother
6042     }
6043   }{}
6044 }

```

`\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.

```

6045 \int_new:N \l_document_structure_section_level_int
6046 \str_case:VnF \c_document_structure_topsect_str {
6047   {part}}{
6048     \int_set:Nn \l_document_structure_section_level_int {0}
6049   }
6050   {chapter}}{
6051     \int_set:Nn \l_document_structure_section_level_int {1}
6052   }
6053 }{
6054   \str_case:VnF \c_document_structure_class_str {
6055     {book}}{
6056       \int_set:Nn \l_document_structure_section_level_int {0}
6057     }
6058     {report}}{
6059       \int_set:Nn \l_document_structure_section_level_int {0}
6060     }
6061   }{
6062     \int_set:Nn \l_document_structure_section_level_int {2}
6063   }
6064 }

```

## 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>16</sup>

EdN:16

```
6065 \def\current@section@level{document}%
6066 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
6067 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
6068 \cs_new_protected:Npn \skipomgroup {
6069   \ifcase\l_document_structure_section_level_int
6070   \or\stepcounter{part}
6071   \or\stepcounter{chapter}
6072   \or\stepcounter{section}
6073   \or\stepcounter{subsection}
6074   \or\stepcounter{subsubsection}
6075   \or\stepcounter{paragraph}
6076   \or\stepcounter{subparagraph}
6077   \fi
6078 }
```

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

`blindfragment`

```
6079 \newcommand\at@begin@blindomgroup[1]{%
6080 \newenvironment{blindfragment}
6081 {
6082   \int_incr:N\l_document_structure_section_level_int
6083   \at@begin@blindomgroup\l_document_structure_section_level_int
6084 }{}}
```

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

```
6085 \newcommand\omgroup@nonum[2]{
6086   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
6087   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
6088 }
```

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

`\omgroup@num` convenience macro: `\omgroup@nonum{<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.

```
6089 \newcommand\omgroup@num[2]{
```

<sup>16</sup>EDNOTE: MK: we may have to experiment with the more powerful uppercasing macro from `mfirstuc.sty` once we internationalize.



```

6090 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
6091   \@nameuse{#1}{#2}
6092 }{
6093   \cs_if_exist:NTF\rdfmata@sectioning{
6094     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
6095   }{
6096     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
6097   }
6098 }
6099 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
6100 }

```

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

**sfragment**

```

6101 \keys_define:nn { document-structure / omgroup }{
6102   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
6103   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
6104   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
6105   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
6106   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
6107   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
6108   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
6109   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
6110   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
6111   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
6112 }
6113 \cs_new_protected:Nn \__document_structure_omgroup_args:n {
6114   \str_clear:N \l__document_structure_omgroup_id_str
6115   \str_clear:N \l__document_structure_omgroup_date_str
6116   \clist_clear:N \l__document_structure_omgroup_creators_clist
6117   \clist_clear:N \l__document_structure_omgroup_contributors_clist
6118   \tl_clear:N \l__document_structure_omgroup_srccite_tl
6119   \tl_clear:N \l__document_structure_omgroup_type_tl
6120   \tl_clear:N \l__document_structure_omgroup_short_tl
6121   \tl_clear:N \l__document_structure_omgroup_display_tl
6122   \tl_clear:N \l__document_structure_omgroup_intro_tl
6123   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
6124   \keys_set:nn { document-structure / omgroup } { #1 }
6125 }

```

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.

```

6126 \newif\if@mainmatter\@mainmattertrue
6127 \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.

```

6128 \keys_define:nn { document-structure / sectioning }{
6129   name .str_set_x:N = \l__document_structure_sect_name_str ,
6130   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
6131   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
6132   clear .default:n = {true} ,
6133   num .bool_set:N = \l__document_structure_sect_num_bool ,

```

```

6134   num      .default:n      = {true}
6135 }
6136 \cs_new_protected:Nn \__document_structure_sect_args:n {
6137   \str_clear:N \l__document_structure_sect_name_str
6138   \str_clear:N \l__document_structure_sect_ref_str
6139   \bool_set_false:N \l__document_structure_sect_clear_bool
6140   \bool_set_false:N \l__document_structure_sect_num_bool
6141   \keys_set:nn { document-structure / sectioning } { #1 }
6142 }
6143 \newcommand\omdoc@sectioning[3][]{
6144   \__document_structure_sect_args:n {#1}
6145   \let\omdoc@ssect@name\l__document_structure_sect_name_str
6146   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
6147   \if@mainmatter% numbering not overridden by frontmatter, etc.
6148     \bool_if:NTF \l__document_structure_sect_num_bool {
6149       \omgroup@num{#2}{#3}
6150     }{
6151       \omgroup@nonum{#2}{#3}
6152     }
6153     \def\current@section@level{\omdoc@ssect@name}
6154   \else
6155     \omgroup@nonum{#2}{#3}
6156   \fi
6157 }% 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.

```

6158 \newcommand\omgroup@redefine@addtocontents[1]{%
6159 %\edef\__document_structureimport{#1}%
6160 %\@for\@I:=\__document_structureimport\do{%
6161 %\edef\@path{\csname module@\@I @path\endcsname}%
6162 %\@ifundefined{tf@toc}\relax%
6163 %   {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
6164 %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
6165 %\def\addcontentsline##1##2##3{%
6166 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6167 %\else% hyperref.sty not loaded
6168 %\def\addcontentsline##1##2##3{%
6169 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6170 %\fi
6171 }% 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.

```

6172 \newenvironment{sfragment}[2][]{% keys, title
6173 {
6174   \__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.

```

6175   \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
6176     \omgroup@redefine@addtocontents{
6177       \@ifundefined{module@id}\used@modules%

```

```

6178     %{\@ifundefined{module@}\module@id @path}{\used@modules}\module@id}
6179   }
6180 }

```

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

```

6181 \int_incr:N\l_document_structure_section_level_int
6182 \ifcase\l_document_structure_section_level_int
6183   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
6184   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
6185   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
6186   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
6187   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
6188   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
6189   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
6190 \fi
6191 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
6192 \str_if_empty:NF \l__document_structure_omgroup_id_str {
6193   \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
6194 }
6195 }% for customization
6196 {}

```

and finally, we localize the sections

```

6197 \newcommand\omdoc@part@kw{Part}
6198 \newcommand\omdoc@chapter@kw{Chapter}
6199 \newcommand\omdoc@section@kw{Section}
6200 \newcommand\omdoc@subsection@kw{Subsection}
6201 \newcommand\omdoc@subsubsection@kw{Subsubsection}
6202 \newcommand\omdoc@paragraph@kw{paragraph}
6203 \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`

```

6204 \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).

```

6205 \cs_if_exist:NTF\frontmatter{
6206   \let\__document_structure_orig_frontmatter\frontmatter
6207   \let\frontmatter\relax
6208 }{
6209   \tl_set:Nn\__document_structure_orig_frontmatter{
6210     \clearpage
6211     \@mainmatterfalse
6212     \pagenumbering{roman}

```

```

6213 }
6214 }
6215 \cs_if_exist:NTF\backmatter{
6216   \let\__document_structure_orig_backmatter\backmatter
6217   \let\backmatter\relax
6218 }{
6219   \tl_set:Nn\__document_structure_orig_backmatter{
6220     \clearpage
6221     \@mainmatterfalse
6222     \pagenumbering{roman}
6223   }
6224 }

```

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.

```

6225 \newenvironment{frontmatter}{
6226   \__document_structure_orig_frontmatter
6227 }{
6228   \cs_if_exist:NTF\mainmatter{
6229     \mainmatter
6230   }{
6231     \clearpage
6232     \@mainmattertrue
6233     \pagenumbering{arabic}
6234   }
6235 }

```

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

```

6236 \newenvironment{backmatter}{
6237   \__document_structure_orig_backmatter
6238 }{
6239   \cs_if_exist:NTF\mainmatter{
6240     \mainmatter
6241   }{
6242     \clearpage
6243     \@mainmattertrue
6244     \pagenumbering{arabic}
6245   }
6246 }

```

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

```

6247 \@mainmattertrue\pagenumbering{arabic}

```

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

```

6248 \def \c__document_structure_document_str{document}
6249 \newcommand\afterprematurestop{}
6250 \def\prematurestop@endomgroup{
6251   \unless\ifx\@currenvir\c__document_structure_document_str
6252     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
6253       \expandafter\prematurestop@endomgroup

```

```

6254 \fi
6255 }
6256 \providecommand\prematurestop{
6257 \message{Stopping~sTeX~processing~prematurely}
6258 \prematurestop@endomgroup
6259 \afterprematurestop
6260 \end{document}
6261 }

```

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

## 37.8 Global Variables

**\setSGvar** set a global variable

```

6262 \RequirePackage{etoolbox}
6263 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

**\useSGvar** use a global variable

```

6264 \newrobustcmd\useSGvar[1]{%
6265 \@ifundefined{sTeX@Gvar@#1}
6266 {\PackageError{document-structure}
6267 {The sTeX Global variable #1 is undefined}
6268 {set it with \protect\setSGvar}}
6269 \@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.

```

6270 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
6271 \@ifundefined{sTeX@Gvar@#1}
6272 {\PackageError{document-structure}
6273 {The sTeX Global variable #1 is undefined}
6274 {set it with \protect\setSGvar}}
6275 {\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.

```
6276 \*cls)
6277 \@@=notesslides)
6278 \ProvidesExplClass{notesslides}{2022/02/28}{3.1.0}{notesslides Class}
6279 \RequirePackage{13keys2e}
6280
6281 \keys_define:nn{notesslides / cls}{
6282   class .code:n = {
6283     \PassOptionsToClass{\CurrentOption}{document-structure}
6284     \str_if_eq:nnT{#1}{book}{
6285       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
6286     }
6287     \str_if_eq:nnT{#1}{report}{
6288       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
6289     }
6290   },
6291   notes .bool_set:N = \c__notesslides_notes_bool ,
6292   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
6293   unknown .code:n = {
6294     \PassOptionsToClass{\CurrentOption}{document-structure}
6295     \PassOptionsToClass{\CurrentOption}{beamer}
6296     \PassOptionsToPackage{\CurrentOption}{notesslides}
6297   }
6298 }
6299 \ProcessKeysOptions{ notesslides / cls }
6300 \bool_if:NTF \c__notesslides_notes_bool {
6301   \PassOptionsToPackage{notes=true}{notesslides}
6302 }{
6303   \PassOptionsToPackage{notes=false}{notesslides}
6304 }
6305 \</cls)
```

now we do the same for the notesslides package.

```

6306 <*package>
6307 \ProvidesExplPackage{notesslides}{2022/02/28}{3.1.0}{notesslides Package}
6308 \RequirePackage{13keys2e}
6309
6310 \keys_define:nn{notesslides / pkg}{
6311   topsect      .str_set_x:N = \c_notesslides_topsect_str,
6312   defaulttopsect .str_set_x:N = \c_notesslides_defaulttopsec_str,
6313   notes        .bool_set:N = \c_notesslides_notes_bool ,
6314   slides       .code:n      = { \bool_set_false:N \c_notesslides_notes_bool },
6315   sectocframes .bool_set:N = \c_notesslides_sectocframes_bool ,
6316   frameimages  .bool_set:N = \c_notesslides_frameimages_bool ,
6317   fiboxed      .bool_set:N = \c_notesslides_fiboxed_bool ,
6318   nopproblems  .bool_set:N = \c_notesslides_nopproblems_bool,
6319   unknown      .code:n      = {
6320     \PassOptionsToClass{\CurrentOption}{stex}
6321     \PassOptionsToClass{\CurrentOption}{tikzinput}
6322   }
6323 }
6324 \ProcessKeysOptions{ notesslides / pkg }
6325 \newif\ifnotes
6326 \bool_if:NTF \c_notesslides_notes_bool {
6327   \notesttrue
6328 }{
6329   \notesfalse
6330 }
6331

```

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

```

6332 \str_if_empty:NTF \c_notesslides_topsect_str {
6333   \str_set_eq:NN \__notesslides_topsect \c_notesslides_defaulttopsec_str
6334 }{
6335   \str_set_eq:NN \__notesslides_topsect \c_notesslides_topsect_str
6336 }
6337 </package>

```

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

```

6338 <*cls>
6339 \bool_if:NTF \c_notesslides_notes_bool {
6340   \LoadClass{document-structure}
6341 }{
6342   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
6343   \newcounter{Item}
6344   \newcounter{paragraph}
6345   \newcounter{subparagraph}
6346   \newcounter{Hfootnote}
6347   \RequirePackage{document-structure}
6348 }

```

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

```

6349 \RequirePackage{notesslides}
6350 </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.

```

6351 <*package>
6352 \bool_if:NT \c__notesslides_notes_bool {
6353   \RequirePackage{a4wide}
6354   \RequirePackage{marginnote}
6355   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
6356   \RequirePackage{mdframed}
6357   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
6358   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
6359 }
6360 \RequirePackage{stex-tikzinput}
6361 \RequirePackage{etoolbox}
6362 \RequirePackage{amssymb}
6363 \RequirePackage{amsmath}
6364 \RequirePackage{comment}
6365 \RequirePackage{textcomp}
6366 \RequirePackage{url}
6367 \RequirePackage{graphicx}
6368 \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>17</sup>

```

6369 \bool_if:NT \c__notesslides_notes_bool {
6370   \renewcommand\usetheme[2][\usepackage[#1]{beamernotestheme#2}]
6371 }
6372
6373
6374 \NewDocumentCommand \libusetheme {0{} m} {
6375   \bool_if:NTF \c__notesslides_notes_bool {
6376     \libusepackage[#1]{beamernotestheme#2}
6377   }{
6378     \libusepackage[#1]{beamertheme#2}
6379   }
6380 }

```

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

```

6381 \newcounter{slide}
6382 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
6383 \newlength{\slideheight}\setlength{\slideheight}{9cm}

```

---

<sup>17</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.

```

6384 \bool_if:NTF \c__notesslides_notes_bool {
6385   \renewenvironment{note}{\ignorespaces}{}
6386 }{
6387   \excludecomment{note}
6388 }

```

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.

```

6389 \bool_if:NT \c__notesslides_notes_bool {
6390   \newlength{\slideframewidth}
6391   \setlength{\slideframewidth}{1.5pt}

```

**frame** We first define the keys.

```

6392 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
6393   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
6394     \bool_set_true:N #1
6395   }{
6396     \bool_set_false:N #1
6397   }
6398 }
6399 \keys_define:nn{notesslides / frame}{
6400   label .str_set_x:N = \l__notesslides_frame_label_str,
6401   allowframebreaks .code:n = {
6402     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
6403   },
6404   allowdisplaybreaks .code:n = {
6405     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
6406   },
6407   fragile .code:n = {
6408     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
6409   },
6410   shrink .code:n = {
6411     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
6412   },
6413   squeeze .code:n = {
6414     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
6415   },
6416   t .code:n = {
6417     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
6418   },
6419 }
6420 \cs_new_protected:Nn \__notesslides_frame_args:n {
6421   \str_clear:N \l__notesslides_frame_label_str
6422   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
6423   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
6424   \bool_set_true:N \l__notesslides_frame_fragile_bool
6425   \bool_set_true:N \l__notesslides_frame_shrink_bool
6426   \bool_set_true:N \l__notesslides_frame_squeeze_bool
6427   \bool_set_true:N \l__notesslides_frame_t_bool

```

```

6428 \keys_set:nn { notesslides / frame }{ #1 }
6429 }

```

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

```

6430 \renewenvironment{frame}[1][]{
6431   \__notesslides_frame_args:n{#1}
6432   \sffamily
6433   \stepcounter{slide}
6434   \def\@currentlabel{\theslide}
6435   \str_if_empty:NF \l__notesslides_frame_label_str {
6436     \label{\l__notesslides_frame_label_str}
6437   }

```

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

```

6438 \def\itemize@level{outer}
6439 \def\itemize@outer{outer}
6440 \def\itemize@inner{inner}
6441 \renewcommand\newpage{\addtocounter{framenumber}{1}}
6442 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
6443 \renewenvironment{itemize}{
6444   \ifx\itemize@level\itemize@outer
6445     \def\itemize@label{$\rhd$}
6446   \fi
6447   \ifx\itemize@level\itemize@inner
6448     \def\itemize@label{$\scriptstyle\rhd$}
6449   \fi
6450   \begin{list}
6451     {\itemize@label}
6452     {\setlength{\labelsep}{.3em}
6453      \setlength{\labelwidth}{.5em}
6454      \setlength{\leftmargin}{1.5em}
6455     }
6456   \edef\itemize@level{\itemize@inner}
6457 }{
6458   \end{list}
6459 }

```

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

```

6460 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth]
6461 }{
6462   \medskip\miko@slidelabel\end{mdframed}
6463 }

```

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

`\frametitle`

```

6464 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
6465 }

```

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

EdN:18

`\pause`

```

18
6466 \bool_if:NT \c__notesslides_notes_bool {
6467   \newcommand\pause{}
6468 }

```

---

<sup>18</sup>EdNOTE: MK: fake it in notes mode for now

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

**nparagraph**

```
6469 \bool_if:NTF \c__notesslides_notes_bool {  
6470   \newenvironment{nparagraph}[1] [] {\begin{sparagraph}[#1]}\end{sparagraph}}  
6471 }{  
6472   \excludecomment{nparagraph}  
6473 }
```

**nfragment**

```
6474 \bool_if:NTF \c__notesslides_notes_bool {  
6475   \newenvironment{nfragment}[2] [] {\begin{sfragment}[#1]{#2}}\end{sfragment}}  
6476 }{  
6477   \excludecomment{nfragment}  
6478 }
```

**ndefinition**

```
6479 \bool_if:NTF \c__notesslides_notes_bool {  
6480   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}\end{sdefinition}}  
6481 }{  
6482   \excludecomment{ndefinition}  
6483 }
```

**nassertion**

```
6484 \bool_if:NTF \c__notesslides_notes_bool {  
6485   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}\end{sassertion}}  
6486 }{  
6487   \excludecomment{nassertion}  
6488 }
```

**nsproof**

```
6489 \bool_if:NTF \c__notesslides_notes_bool {  
6490   \newenvironment{nsproof}[2] [] {\begin{sproof}[#1]{#2}}\end{sproof}}  
6491 }{  
6492   \excludecomment{nsproof}  
6493 }
```

**nexample**

```
6494 \bool_if:NTF \c__notesslides_notes_bool {  
6495   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}\end{sexample}}  
6496 }{  
6497   \excludecomment{nexample}  
6498 }
```

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

```
6499 \def\inputref@preskip{\smallskip}  
6500 \def\inputref@postskip{\medskip}
```

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

`\inputref*`

```
6501 \let\orig@inputref\inputref
6502 \def\inputref{\@ifstar\ninputref\orig@inputref}
6503 \newcommand\ninputref[2][] {
6504   \bool_if:NT \c__notesslides_notes_bool {
6505     \orig@inputref[#1]{#2}
6506   }
6507 }
```

(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>}`.

```
6508 \newlength{\slidelogoheight}
6509
6510 \bool_if:NTF \c__notesslides_notes_bool {
6511   \setlength{\slidelogoheight}{.4cm}
6512 }{
6513   \setlength{\slidelogoheight}{1cm}
6514 }
6515 \newsavebox{\slidelogo}
6516 \sbox{\slidelogo}{\text{\TeX}}
6517 \newrobustcmd{\setslidelogo}[1]{
6518   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
6519 }
```

(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.

```
6520 \def\source{Michael Kohlhase}% customize locally
6521 \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.

```
6522 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
6523 \newsavebox{\cclogo}
6524 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
6525 \newif\ifcchref\cchreffalse
6526 \AtBeginDocument{
6527   \@ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
6528 }
6529 \def\licensing{
6530   \ifcchref
```

```

6531     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
6532 \else
6533     {\usebox{\cclogo}}
6534 \fi
6535 }
6536 \newrobustcmd{\setlicensing}[2][]{
6537   \def\@url{#1}
6538   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
6539   \ifx\@url\@empty
6540     \def\licensing{\usebox{\cclogo}}
6541   \else
6542     \def\licensing{
6543       \ifcchref
6544         \href{#1}{\usebox{\cclogo}}
6545       \else
6546         {\usebox{\cclogo}}
6547       \fi
6548     }
6549   \fi
6550 }

```

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

EdN:19

\slidelabel Now, we set up the slide label for the article mode.<sup>19</sup>

```

6551 \newrobustcmd\miko@slidelabel{
6552   \vbox to \slidelogoheight{
6553     \vss\hbox to \slidewidth
6554     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
6555   }
6556 }

```

(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.

```

6557 \def\Gin@mhrepos{}
6558 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
6559 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
6560 \newrobustcmd\frameimage[2][]{
6561   \stepcounter{slide}
6562   \bool_if:NT \c__notesslides_frameimages_bool {
6563     \def\Gin@ewidth{}\setkeys{Gin}{#1}
6564     \bool_if:NF \c__notesslides_notes_bool { \vfill }
6565     \begin{center}
6566       \bool_if:NTF \c__notesslides_fboxed_bool {
6567         \fbox{
6568           \ifx\Gin@ewidth\@empty
6569             \ifx\Gin@mhrepos\@empty
6570               \mhgraphics[width=\slidewidth,#1]{#2}
6571             \else

```

---

<sup>19</sup>EdNOTE: see that we can use the themes for the slides some day. This is all fake.

```

6572         \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
6573     \fi
6574 \else% Gin@ewidth empty
6575     \ifx\Gin@mhrepos\@empty
6576         \mhgraphics[#1]{#2}
6577     \else
6578         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
6579     \fi
6580 \fi% Gin@ewidth empty
6581 }
6582 }{
6583     \ifx\Gin@ewidth\@empty
6584         \ifx\Gin@mhrepos\@empty
6585             \mhgraphics[width=\slidewidth,#1]{#2}
6586         \else
6587             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
6588         \fi
6589         \ifx\Gin@mhrepos\@empty
6590             \mhgraphics[#1]{#2}
6591         \else
6592             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
6593         \fi
6594     \fi% Gin@ewidth empty
6595 }
6596 \end{center}
6597 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
6598 \bool_if:NF \c__notesslides_notes_bool { \vfill }
6599 }
6600 } % 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.

```

6601 \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

```

6602 \AddToHook{begindocument}{
6603     \definecolor{green}{rgb}{0,.5,0}
6604     \definecolor{purple}{cmyk}{.3,1,0,.17}
6605 }

```

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`.

```

6606 % \def\STpresent#1{\textcolor{blue}{#1}}
6607 \def\defemph#1{\textcolor{magenta}{#1}}
6608 \def\symrefemph#1{\textcolor{cyan}{#1}}
6609 \def\compemph#1{\textcolor{blue}{#1}}
6610 \def\titleemph#1{\textcolor{blue}{#1}}
6611 \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.

```

6612 \pgfdeclareimage[width=.8em]{miko@small@dbend}{stex-dangerous-bend}
6613 \def\smalltextwarning{
6614   \pgfuseimage{miko@small@dbend}
6615   \xspace
6616 }
6617 \pgfdeclareimage[width=1.2em]{miko@dbend}{stex-dangerous-bend}
6618 \newrobustcmd\textwarning{
6619   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
6620   \xspace
6621 }
6622 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{stex-dangerous-bend}
6623 \newrobustcmd\bigtextwarning{
6624   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
6625   \xspace
6626 }

```

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

```

6627 \newrobustcmd\putgraphicsat[3]{
6628   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
6629 }
6630 \newrobustcmd\putat[2]{
6631   \begin{picture}(0,0)\put(#1){#2}\end{picture}
6632 }

```

## 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`.

```

6633 \bool_if:NT \c__notesslides_sectocframes_bool {
6634   \str_if_eq:VnTF \__notesslidesstopsect{part}{
6635     \newcounter{chapter}\counterwithin*{section}{chapter}
6636   }{
6637     \str_if_eq:VnT \__notesslidesstopsect{chapter}{
6638       \newcounter{chapter}\counterwithin*{section}{chapter}
6639     }
6640   }
6641 }

```

`\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
6642 \def\part@prefix{}
6643 \@ifpackageloaded{document-structure}{\{
6644   \str_case:VnF \__notesslidesstopsect {
6645     {part}{
6646       \int_set:Nn \l_document_structure_section_level_int {0}
6647       \def\thesection{\arabic{chapter}.\arabic{section}}

```

```

6648     \def\part@prefix{\arabic{chapter}.}
6649   }
6650   {chapter}{
6651     \int_set:Nn \l_document_structure_section_level_int {1}
6652     \def\thesection{\arabic{chapter}.\arabic{section}}
6653     \def\part@prefix{\arabic{chapter}.}
6654   }
6655   }{
6656     \int_set:Nn \l_document_structure_section_level_int {2}
6657     \def\part@prefix{}
6658   }
6659 }
6660
6661 \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

```

6662 \renewenvironment{sfragment}[2][]{
6663   \_document_structure_omgroup_args:n { #1 }
6664   \int_incr:N \l_document_structure_section_level_int
6665   \bool_if:NT \c__notesslides_sectocframes_bool {
6666     \stepcounter{slide}
6667     \begin{frame}[noframenumbering]
6668       \vfill\Large\centering
6669     \red{
6670       \ifcase\l_document_structure_section_level_int\or
6671         \stepcounter{part}
6672         \def\_notesslideslabel{\omdoc@part@kw~\Roman{part}}
6673         \def\currentsectionlevel{\omdoc@part@kw}
6674       \or
6675         \stepcounter{chapter}
6676         \def\_notesslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
6677         \def\currentsectionlevel{\omdoc@chapter@kw}
6678       \or
6679         \stepcounter{section}
6680         \def\_notesslideslabel{\part@prefix\arabic{section}}
6681         \def\currentsectionlevel{\omdoc@section@kw}
6682       \or
6683         \stepcounter{subsection}
6684         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
6685         \def\currentsectionlevel{\omdoc@subsection@kw}
6686       \or
6687         \stepcounter{subsubsection}
6688         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
6689         \def\currentsectionlevel{\omdoc@subsubsection@kw}
6690       \or
6691         \stepcounter{paragraph}
6692         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
6693         \def\currentsectionlevel{\omdoc@paragraph@kw}
6694       \else
6695         \def\_notesslideslabel{}

```



```

6696         \def\currentsectionlevel{\omdoc@paragraph@kw}
6697         \fi% end ifcase
6698         \_notesslideslabel%\sref@label@id\_notesslideslabel
6699         \quad #2%
6700     }%
6701     \vfill%
6702     \end{frame}%
6703 }
6704 \str_if_empty:NF \l__document_structure_omgroup_id_str {
6705     \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
6706 }
6707 }{}
6708 }

```

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

```

6709 \def\inserttheorembodyfont{\normalfont}
6710 %\bool_if:NF \c__notesslides_notes_bool {
6711 % \defbeamertemplate{theorem begin}{miko}
6712 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
6713 % \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
6714 % \inserttheorempunctuation\inserttheorembodyfont\xspace}
6715 % \defbeamertemplate{theorem end}{miko}{}}

```

and we set it as the default one.

```

6716 % \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.

```

6717 % \expandafter\def\csname Parent2\endcsname{}
6718 %}
6719
6720 \AddToHook{begindocument}{% this does not work for some reason
6721     \setbeamertemplate{theorems}[ams style]
6722 }
6723 \bool_if:NT \c__notesslides_notes_bool {
6724     \renewenvironment{columns}[1][{}]{%
6725         \par\noindent%
6726         \begin{minipage}%
6727             \slidewidth\centering\leavevmode%
6728     }{}%
6729     \end{minipage}\par\noindent%
6730 }%
6731 \newsavebox\columnbox%
6732 \renewenvironment<>{column}[2][{}]{%
6733     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
6734 }{}%
6735     \end{minipage}\end{lrbox}\usebox\columnbox%
6736 }%
6737 }
6738 \bool_if:NTF \c__notesslides_noproblems_bool {
6739     \newenvironment{problems}{}{}
6740 }{
6741     \excludecomment{problems}
6742 }

```

## 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.

```

6743 \gdef\printexcursions{}
6744 \newcommand\excursionref[2]{% label, text
6745   \bool_if:NT \c__notesslides_notes_bool {
6746     \begin{sparagraph}[title=Excursion]
6747       #2 \sref[fallback=the appendix]{#1}.
6748     \end{sparagraph}
6749   }
6750 }
6751 \newcommand\activate@excursion[2][]{
6752   \gappto\printexcursions{\inputref{#1}{#2}}
6753 }
6754 \newcommand\excursion[4][]{% repos, label, path, text
6755   \bool_if:NT \c__notesslides_notes_bool {
6756     \activate@excursion[#1]{#3}\excursionref{#2}{#4}
6757   }
6758 }

```

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

`\excursiongroup`

```

6759 \keys_define:nn{notesslides / excursiongroup }{
6760   id          .str_set_x:N = \l__notesslides_excursion_id_str,
6761   intro       .tl_set:N   = \l__notesslides_excursion_intro_tl,
6762   mhrepos     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
6763 }
6764 \cs_new_protected:Nn \__notesslides_excursion_args:n {
6765   \tl_clear:N \l__notesslides_excursion_intro_tl
6766   \str_clear:N \l__notesslides_excursion_id_str
6767   \str_clear:N \l__notesslides_excursion_mhrepos_str
6768   \keys_set:nn {notesslides / excursiongroup }{ #1 }
6769 }
6770 \newcommand\excursiongroup[1][]{
6771   \__notesslides_excursion_args:n{ #1 }
6772   \ifdefempty\printexcursions{}% only if there are excursions
6773   {\begin{note}
6774     \begin{sfragment}[#1]{Excursions}%
6775     \ifdefempty\l__notesslides_excursion_intro_tl{\
6776       \inputref[\l__notesslides_excursion_mhrepos_str]{
6777         \l__notesslides_excursion_intro_tl
6778       }
6779     }
6780     \printexcursions%
6781     \end{sfragment}
6782     \end{note}}
6783 }
6784 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{\fi
6785 \</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.

```
6786 <*package>
6787 <@@=problems>
6788 \ProvidesExplPackage{problem}{2022/02/26}{3.0.1}{Semantic Markup for Problems}
6789 \RequirePackage{l3keys2e,stex}
6790
6791 \keys_define:nn { problem / pkg }{
6792   notes      .default:n    = { true },
6793   notes      .bool_set:N   = \c__problems_notes_bool,
6794   gnotes     .default:n    = { true },
6795   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
6796   hints      .default:n    = { true },
6797   hints      .bool_set:N   = \c__problems_hints_bool,
6798   solutions  .default:n    = { true },
6799   solutions  .bool_set:N   = \c__problems_solutions_bool,
6800   pts        .default:n    = { true },
6801   pts        .bool_set:N   = \c__problems_pts_bool,
6802   min        .default:n    = { true },
6803   min        .bool_set:N   = \c__problems_min_bool,
6804   boxed      .default:n    = { true },
6805   boxed      .bool_set:N   = \c__problems_boxed_bool,
6806   unknown    .code:n       = {}
6807 }
6808 \newif\ifsolutions
6809
6810 \ProcessKeysOptions{ problem / pkg }
6811 \bool_if:NTF \c__problems_solutions_bool {
6812   \solutionstrue
6813 }{
6814   \solutionsfalse
6815 }
```

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

```
6816 \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.

```
6817 \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.

```
6818 \def\prob@problem@kw{Problem}
6819 \def\prob@solution@kw{Solution}
6820 \def\prob@hint@kw{Hint}
6821 \def\prob@note@kw{Note}
6822 \def\prob@gnote@kw{Grading}
6823 \def\prob@pt@kw{pt}
6824 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
6825 \AddToHook{begindocument}{
6826   \ltx@ifpackageloaded{babel}{
6827     \makeatletter
6828     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6829     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6830       \input{problem-ngerman.ldf}
6831     }
6832     \clist_if_in:NnT \l_tmpa_clist {finnish}{
6833       \input{problem-finnish.ldf}
6834     }
6835     \clist_if_in:NnT \l_tmpa_clist {french}{
6836       \input{problem-french.ldf}
6837     }
6838     \clist_if_in:NnT \l_tmpa_clist {russian}{
6839       \input{problem-russian.ldf}
6840     }
6841     \makeatother
6842   }{}
6843 }
```

## 39.2 Problems and Solutions

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

```
6844 \keys_define:nn{ problem / problem }{
6845   id      .str_set_x:N = \l__problems_prob_id_str,
6846   pts     .tl_set:N    = \l__problems_prob_pts_tl,
6847   min     .tl_set:N    = \l__problems_prob_min_tl,
6848   title   .tl_set:N    = \l__problems_prob_title_tl,
6849   type    .tl_set:N    = \l__problems_prob_type_tl,
6850   refnum  .int_set:N   = \l__problems_prob_refnum_int
6851 }
6852 \cs_new_protected:Nn \__problems_prob_args:n {
```

```

6853 \str_clear:N \l__problems_prob_id_str
6854 \tl_clear:N \l__problems_prob_pts_tl
6855 \tl_clear:N \l__problems_prob_min_tl
6856 \tl_clear:N \l__problems_prob_title_tl
6857 \tl_clear:N \l__problems_prob_type_tl
6858 \int_zero_new:N \l__problems_prob_refnum_int
6859 \keys_set:nn { problem / problem }{ #1 }
6860 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
6861   \let\l__problems_prob_refnum_int\undefined
6862 }
6863 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

6864 \newcounter{problem}
6865 \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.

```

6866 \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

```

6867 \newcommand\prob@number{
6868   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
6869     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
6870   }{
6871     \int_if_exist:NTF \l__problems_prob_refnum_int {
6872       \prob@label{\int_use:N \l__problems_prob_refnum_int }
6873     }{
6874       \prob@label\theproblem
6875     }
6876   }
6877 }

```

(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.

```

6878 \newcommand\prob@title[3]{%
6879   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
6880     #2 \l__problems_inclprob_title_tl #3
6881   }{
6882     \tl_if_exist:NTF \l__problems_prob_title_tl {
6883       #2 \l__problems_prob_title_tl #3
6884     }{
6885       #1
6886     }
6887   }
6888 }

```

(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.

```

6889 \def\prob@heading{
6890   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
6891   %\sref@label{id}\prob@problem@kw~\prob@number}{~}
6892 }

```

(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`

```

6893 \newenvironment{sproblem}[1][{}]{
6894   \__problems_prob_args:n{#1}%\sref@target%
6895   \@in@omtexttrue% we are in a statement (for inline definitions)
6896   \stepcounter{problem}\record@problem
6897   \def\current@section@level{\prob@problem@kw}
6898   \tl_if_exist:NTF \l__problems_inclprob_type_tl {
6899     \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
6900   }{
6901     \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
6902   }
6903   \str_if_exist:NTF \l__problems_inclprob_id_str {
6904     \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
6905   }{
6906     \str_set_eq:NN \sproblemid \l__problems_prob_id_str
6907   }
6908
6909
6910   \clist_set:No \l_tmpa_clist \sproblemtype
6911   \tl_clear:N \l_tmpa_tl
6912   \clist_map_inline:Nn \l_tmpa_clist {
6913     \tl_if_exist:cT {\__problems_sproblem_##1_start:}{
6914       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_start:}}
6915     }
6916   }
6917   \tl_if_empty:NTF \l_tmpa_tl {
6918     \__problems_sproblem_start:
6919   }{
6920     \l_tmpa_tl
6921   }
6922   \stex_ref_new_doc_target:n \sproblemid
6923 }{
6924   \clist_set:No \l_tmpa_clist \sproblemtype
6925   \tl_clear:N \l_tmpa_tl
6926   \clist_map_inline:Nn \l_tmpa_clist {
6927     \tl_if_exist:cT {\__problems_sproblem_##1_end:}{
6928       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_end:}}
6929     }

```

```

6930 }
6931 \tl_if_empty:NTF \l_tmpa_tl {
6932   \__problems_sproblem_end:
6933 }{
6934   \l_tmpa_tl
6935 }
6936
6937
6938 \smallskip
6939 }
6940
6941
6942 \cs_new_protected:Nn \__problems_sproblem_start: {
6943   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
6944 }
6945 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
6946
6947 \newcommand\stexpatchproblem[3][] {
6948   \str_set:Nx \l_tmpa_str{ #1 }
6949   \str_if_empty:NTF \l_tmpa_str {
6950     \tl_set:Nn \__problems_sproblem_start: { #2 }
6951     \tl_set:Nn \__problems_sproblem_end: { #3 }
6952   }{
6953     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
6954     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
6955   }
6956 }
6957
6958
6959 \bool_if:NT \c__problems_boxed_bool {
6960   \surroundwithmdframed{problem}
6961 }

```

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

```

6962 \def\record@problem{
6963   \protected@write\@auxout{}
6964   {
6965     \string\@problem{\prob@number}
6966     {
6967       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
6968         \l__problems_inclprob_pts_tl
6969       }{
6970         \l__problems_prob_pts_tl
6971       }
6972     }%
6973     {
6974       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
6975         \l__problems_inclprob_min_tl
6976       }{
6977         \l__problems_prob_min_tl
6978       }
6979     }
6980   }
6981 }

```

(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).

```
6982 \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.

```
6983 \keys_define:nn { problem / solution }{
6984   id                .str_set_x:N = \l__problems_solution_id_str ,
6985   for               .tl_set:N    = \l__problems_solution_for_tl ,
6986   height            .dim_set:N   = \l__problems_solution_height_dim ,
6987   creators          .clist_set:N = \l__problems_solution_creators_clist ,
6988   contributors      .clist_set:N = \l__problems_solution_contributors_clist ,
6989   srccite           .tl_set:N    = \l__problems_solution_srccite_tl
6990 }
6991 \cs_new_protected:Nn \__problems_solution_args:n {
6992   \str_clear:N \l__problems_solution_id_str
6993   \tl_clear:N \l__problems_solution_for_tl
6994   \tl_clear:N \l__problems_solution_srccite_tl
6995   \clist_clear:N \l__problems_solution_creators_clist
6996   \clist_clear:N \l__problems_solution_contributors_clist
6997   \dim_zero:N \l__problems_solution_height_dim
6998   \keys_set:nn { problem / solution }{ #1 }
6999 }
```

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

```
7000 \newcommand\@startsolution[1][{}]{
7001   \__problems_solution_args:n { #1 }
7002   \@in@omtexttrue% we are in a statement.
7003   \bool_if:NF \c__problems_boxed_bool { \hrule }
7004   \smallskip\noindent
7005   {\textbf\prob@solution@kw : \enspace}
7006   \begin{small}
7007   \def\current@section@level{\prob@solution@kw}
7008   \ignorespacesandpars
7009 }
```

**\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.

```
7010 \newcommand\startsolutions{
7011   \specialcomment{solution}{\@startsolution}{
7012     \bool_if:NF \c__problems_boxed_bool {
7013       \hrule\medskip
7014     }
7015     \end{small}%
7016   }
7017   \bool_if:NT \c__problems_boxed_bool {
7018     \surroundwithmdframed{solution}
7019   }
7020 }
```



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

\stopsolutions

```
7021 \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.

```
7022 \ifsolutions
7023   \startsolutions
7024 \else
7025   \stopsolutions
7026 \fi
```

exnote

```
7027 \bool_if:NTF \c__problems_notes_bool {
7028   \newenvironment{exnote}[1][]{
7029     \par\smallskip\hrule\smallskip
7030     \noindent\textbf{\prob@note@kw : }\small
7031   }{
7032     \smallskip\hrule
7033   }
7034 }{
7035   \excludecomment{exnote}
7036 }
```

hint

```
7037 \bool_if:NTF \c__problems_notes_bool {
7038   \newenvironment{hint}[1][]{
7039     \par\smallskip\hrule\smallskip
7040     \noindent\textbf{\prob@hint@kw :~ }\small
7041   }{
7042     \smallskip\hrule
7043   }
7044   \newenvironment{exhint}[1][]{
7045     \par\smallskip\hrule\smallskip
7046     \noindent\textbf{\prob@hint@kw :~ }\small
7047   }{
7048     \smallskip\hrule
7049   }
7050 }{
7051   \excludecomment{hint}
7052   \excludecomment{exhint}
7053 }
```

gnote

```
7054 \bool_if:NTF \c__problems_notes_bool {
7055   \newenvironment{gnote}[1][]{
7056     \par\smallskip\hrule\smallskip
7057     \noindent\textbf{\prob@gnote@kw : }\small
7058   }{
7059     \smallskip\hrule
7060   }
7061 }{
7062   \excludecomment{gnote}
7063 }
```

### 39.3 Multiple Choice Blocks

```

7064 \newenvironment{mcb}{
7065   \begin{enumerate}
7066 }{
7067   \end{enumerate}
7068 }

```

we define the keys for the mcc macro

```

7069 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
7070   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
7071     \bool_set_true:N #1
7072   }{
7073     \bool_set_false:N #1
7074   }
7075 }
7076 \keys_define:nn { problem / mcc }{
7077   id          .str_set_x:N = \l__problems_mcc_id_str ,
7078   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
7079   T           .default:n    = { true } ,
7080   T           .bool_set:N   = \l__problems_mcc_t_bool ,
7081   F           .default:n    = { true } ,
7082   F           .bool_set:N   = \l__problems_mcc_f_bool ,
7083   Ttext       .code:n       = {
7084     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
7085   } ,
7086   Ftext       .code:n       = {
7087     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
7088   }
7089 }
7090 \cs_new_protected:Nn \l__problems_mcc_args:n {
7091   \str_clear:N \l__problems_mcc_id_str
7092   \tl_clear:N \l__problems_mcc_feedback_tl
7093   \bool_set_true:N \l__problems_mcc_t_bool
7094   \bool_set_true:N \l__problems_mcc_f_bool
7095   \bool_set_true:N \l__problems_mcc_Ttext_bool
7096   \bool_set_false:N \l__problems_mcc_Ftext_bool
7097   \keys_set:nn { problem / mcc }{ #1 }
7098 }

```

\mcc

```

7099 \newcommand\mcc[2][] {
7100   \l__problems_mcc_args:n{ #1 }
7101   \item #2
7102   \ifsolutions
7103     \l
7104     \bool_if:NT \l__problems_mcc_t_bool {
7105       % TODO!
7106       % \ifcsstring{mcc@T}{T}{ }\{mcc@Ttext}%
7107     }
7108     \bool_if:NT \l__problems_mcc_f_bool {

```

---

<sup>20</sup>EdNOTE: MK: maybe import something better here from a dedicated MC package

```

7109      % TODO!
7110      % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
7111    }
7112    \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
7113      !
7114    }{
7115      \l__problems_mcc_feedback_tl
7116    }
7117    \fi
7118  } %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.

```

7119
7120 \keys_define:nn{ problem / inclproblem }{
7121   id      .str_set:N = \l__problems_inclprob_id_str,
7122   pts     .tl_set:N  = \l__problems_inclprob_pts_tl,
7123   min     .tl_set:N  = \l__problems_inclprob_min_tl,
7124   title   .tl_set:N  = \l__problems_inclprob_title_tl,
7125   refnum  .int_set:N  = \l__problems_inclprob_refnum_int,
7126   type    .tl_set:N  = \l__problems_inclprob_type_tl,
7127   mhrepos .str_set:N = \l__problems_inclprob_mhrepos_str
7128 }
7129 \cs_new_protected:Nn \l__problems_inclprob_args:n {
7130   \str_clear:N \l__problems_prob_id_str
7131   \tl_clear:N \l__problems_inclprob_pts_tl
7132   \tl_clear:N \l__problems_inclprob_min_tl
7133   \tl_clear:N \l__problems_inclprob_title_tl
7134   \tl_clear:N \l__problems_inclprob_type_tl
7135   \int_zero_new:N \l__problems_inclprob_refnum_int
7136   \str_clear:N \l__problems_inclprob_mhrepos_str
7137   \keys_set:nn { problem / inclproblem }{ #1 }
7138   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
7139     \let\l__problems_inclprob_pts_tl\undefined
7140   }
7141   \tl_if_empty:NT \l__problems_inclprob_min_tl {
7142     \let\l__problems_inclprob_min_tl\undefined
7143   }
7144   \tl_if_empty:NT \l__problems_inclprob_title_tl {
7145     \let\l__problems_inclprob_title_tl\undefined
7146   }
7147   \tl_if_empty:NT \l__problems_inclprob_type_tl {
7148     \let\l__problems_inclprob_type_tl\undefined
7149   }
7150   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
7151     \let\l__problems_inclprob_refnum_int\undefined
7152   }
7153 }

```

```

7154
7155 \cs_new_protected:Nn \__problems_inclprob_clear: {
7156   \let\l__problems_inclprob_id_str\undefined
7157   \let\l__problems_inclprob_pts_tl\undefined
7158   \let\l__problems_inclprob_min_tl\undefined
7159   \let\l__problems_inclprob_title_tl\undefined
7160   \let\l__problems_inclprob_type_tl\undefined
7161   \let\l__problems_inclprob_refnum_int\undefined
7162   \let\l__problems_inclprob_mhrepos_str\undefined
7163 }
7164 \__problems_inclprob_clear:
7165
7166 \newcommand\includeproblem[2][ ]{
7167   \__problems_inclprob_args:n{ #1 }
7168   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
7169     \input{#2}
7170   }{
7171     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
7172       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
7173     }
7174   }
7175   \__problems_inclprob_clear:
7176 }

```

(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.

```

7177 \AddToHook{enddocument}{
7178   \bool_if:NT \c__problems_pts_bool {
7179     \message{Total:~\arabic{pts}~points}
7180   }
7181   \bool_if:NT \c__problems_min_bool {
7182     \message{Total:~\arabic{min}~minutes}
7183   }
7184 }

```

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

```

7185 \def\pts#1{
7186   \bool_if:NT \c__problems_pts_bool {
7187     \marginpar{#1~\prob@pt@kw}
7188   }
7189 }
7190 \def\min#1{
7191   \bool_if:NT \c__problems_min_bool {
7192     \marginpar{#1~\prob@min@kw}
7193   }
7194 }

```

`\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.

```

7195 \newcounter{pts}
7196 \def\show@pts{
7197   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7198     \bool_if:NT \c__problems_pts_bool {
7199       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
7200       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
7201     }
7202   }{
7203     \tl_if_exist:NT \l__problems_prob_pts_tl {
7204       \bool_if:NT \c__problems_pts_bool {
7205         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
7206         \addtocounter{pts}{\l__problems_prob_pts_tl}
7207       }
7208     }
7209   }
7210 }

```

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

`\show@min`

```

7211 \newcounter{min}
7212 \def\show@min{
7213   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7214     \bool_if:NT \c__problems_min_bool {
7215       \marginpar{\l__problems_inclprob_min_tl\ min}
7216       \addtocounter{min}{\l__problems_inclprob_min_tl}
7217     }
7218   }{
7219     \tl_if_exist:NT \l__problems_prob_min_tl {
7220       \bool_if:NT \c__problems_min_bool {
7221         \marginpar{\l__problems_prob_min_tl\ min}
7222         \addtocounter{min}{\l__problems_prob_min_tl}
7223       }
7224     }
7225   }
7226 }
7227 \</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.

```
7228 \@@=hwexam>
7229 \*cls>
7230 \ProvidesExplClass{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
7231 \RequirePackage{l3keys2e}
7232 \DeclareOption*{
7233   \PassOptionsToClass{\CurrentOption}{document-structure}
7234   \PassOptionsToPackage{\CurrentOption}{stex}
7235   \PassOptionsToPackage{\CurrentOption}{hwexam}
7236   \PassOptionsToPackage{\CurrentOption}{tikzinput}
7237 }
7238 \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.

```
7239 \LoadClass{document-structure}
7240 \RequirePackage{stex}
7241 \RequirePackage{hwexam}
7242 \RequirePackage{tikzinput}
7243 \RequirePackage{graphicx}
7244 \RequirePackage{a4wide}
7245 \RequirePackage{amssymb}
7246 \RequirePackage{amstext}
7247 \RequirePackage{amsmath}
```

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

```

7248 \newcommand\assig@default@type{\hwexam@assignment@kw}
7249 \def\document@hwexamtype{\assig@default@type}
7250 <@@=document_structure>
7251 \keys_define:nn { document-structure / document }{
7252 id .str_set_x:N = \c_document_structure_document_id_str,
7253 hwexamtype .tl_set:N = \document@hwexamtype
7254 }
7255 <@@=hwexam>
7256 </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.

```
7257 \*package>
7258 \ProvidesExplPackage{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
7259 \RequirePackage{13keys2e}
7260
7261 \newif\iftest\testfalse
7262 \DeclareOption{test}{\testtrue}
7263 \newif\ifmultiple\multiplefalse
7264 \DeclareOption{multiple}{\multipletrue}
7265 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
7266 \ProcessOptions
```

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

```
7267 \RequirePackage{keyval}[1997/11/10]
7268 \RequirePackage{problem}
```

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

```
7269 \newcommand\hwexam@assignment@kw{Assignment}
7270 \newcommand\hwexam@given@kw{Given}
7271 \newcommand\hwexam@due@kw{Due}
7272 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
7273 blank~for~extra~space}
7274 \def\hwexam@minutes@kw{minutes}
7275 \newcommand\correction@probs@kw{prob.}
7276 \newcommand\correction@pts@kw{total}
7277 \newcommand\correction@reached@kw{reached}
7278 \newcommand\correction@sum@kw{Sum}
7279 \newcommand\correction@grade@kw{grade}
7280 \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

```

7281 \AddToHook{begindocument}{
7282 \ltx@ifpackageloaded{babel}{
7283 \makeatletter
7284 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7285 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7286 \input{hwexam-ngerman.ldf}
7287 }
7288 \clist_if_in:NnT \l_tmpa_clist {finnish}{
7289 \input{hwexam-finnish.ldf}
7290 }
7291 \clist_if_in:NnT \l_tmpa_clist {french}{
7292 \input{hwexam-french.ldf}
7293 }
7294 \clist_if_in:NnT \l_tmpa_clist {russian}{
7295 \input{hwexam-russian.ldf}
7296 }
7297 \makeatother
7298 }{}
7299 }
7300

```

## 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.

```

7301 \newcounter{assignment}
7302 \numberproblemsin{assignment}
7303 \renewcommand\prob@label[1]{\assignment@number.#1}

```

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

```

7304 \keys_define:nn { hwexam / assignment } {
7305 id .str_set:x:N = \l__hwexam_assign_id_str,
7306 number .int_set:N = \l__hwexam_assign_number_int,
7307 title .tl_set:N = \l__hwexam_assign_title_tl,
7308 type .tl_set:N = \l__hwexam_assign_type_tl,
7309 given .tl_set:N = \l__hwexam_assign_given_tl,
7310 due .tl_set:N = \l__hwexam_assign_due_tl,
7311 loadmodules .code:n = {
7312 \bool_set_true:N \l__hwexam_assign_loadmodules_bool
7313 }
7314 }
7315 \cs_new_protected:Nn \__hwexam_assignment_args:n {
7316 \str_clear:N \l__hwexam_assign_id_str
7317 \int_set:Nn \l__hwexam_assign_number_int {-1}
7318 \tl_clear:N \l__hwexam_assign_title_tl
7319 \tl_clear:N \l__hwexam_assign_type_tl
7320 \tl_clear:N \l__hwexam_assign_given_tl
7321 \tl_clear:N \l__hwexam_assign_due_tl
7322 \bool_set_false:N \l__hwexam_assign_loadmodules_bool

```

```

7323 \keys_set:nn { hwexam / assignment }{ #1 }
7324 }

```

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.

```

7325 \newcommand\given@due[2]{
7326 \bool_lazy_all:nF {
7327 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
7328 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
7329 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
7330 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
7331 }{ #1 }
7332
7333 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
7334 \tl_if_empty:NF \l__hwexam_assign_given_tl {
7335 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
7336 }
7337 }{
7338 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
7339 }
7340
7341 \bool_lazy_or:nnF {
7342 \bool_lazy_and_p:nn {
7343 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
7344 }{
7345 \tl_if_empty_p:V \l__hwexam_assign_due_tl
7346 }
7347 }{
7348 \bool_lazy_and_p:nn {
7349 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
7350 }{
7351 \tl_if_empty_p:V \l__hwexam_assign_due_tl
7352 }
7353 }{ ,~ }
7354
7355 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
7356 \tl_if_empty:NF \l__hwexam_assign_due_tl {
7357 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
7358 }
7359 }{
7360 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
7361 }
7362
7363 \bool_lazy_all:nF {
7364 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
7365 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
7366 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
7367 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
7368 }{ #2 }
7369 }

```

`\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.

```

7370 \newcommand\assignment@title[3]{
7371 \tl_if_empty:NTF \l__hwexam_inclasssign_title_tl {
7372 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
7373 #1
7374 }{
7375 #2\l__hwexam_assign_title_tl#3
7376 }
7377 }{
7378 #2\l__hwexam_inclasssign_title_tl#3
7379 }
7380 }

```

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

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

```

7381 \newcommand\assignment@number{
7382 \int_compare:nNnTF \l__hwexam_inclasssign_number_int = {-1} {
7383 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
7384 \arabic{assignment}
7385 } {
7386 \int_use:N \l__hwexam_assign_number_int
7387 }
7388 }{
7389 \int_use:N \l__hwexam_inclasssign_number_int
7390 }
7391 }

```

(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.

```

7392 \newenvironment{assignment}[1][ ]{
7393 \__hwexam_assignment_args:n { #1 }
7394 %\sref@target
7395 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
7396 \global\stepcounter{assignment}
7397 }{
7398 \global\setcounter{assignment}{\int_use:N\l__hwexam_assign_number_int}
7399 }
7400 \setcounter{problem}{0}
7401 \def\current@section@level{\document@hwexamtype}
7402 %\sref@label@id{\document@hwexamtype \thesection}
7403 \begin{@assignment}
7404 }{
7405 \end{@assignment}
7406 }

```

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

```

7407 \def\ass@title{
7408 \protect\document@hwexamtype~\arabic{assignment}
7409 \assignment@title{}\;\;{}{}\; -- \given@due{}\}
7410 }
7411 \ifmultiple
7412 \newenvironment{@assignment}{
7413 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
7414 \begin{sfragment}[loadmodules]{\ass@title}
7415 }{
7416 \begin{sfragment}{\ass@title}
7417 }
7418 }{
7419 \end{sfragment}
7420 }

```

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

```

7421 \else
7422 \newenvironment{@assignment}{
7423 \begin{center}\bf
7424 \Large@title\strut\
7425 \document@hwexamtype~\arabic{assignment}\assignment@title{}\;\;{}{}\;
7426 \large\given@due{--\;\;{}{}\;--}
7427 \end{center}
7428 }{}
7429 \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.

```

7430 \keys_define:nn { hwexam / inclassignment } {
7431 %id .str_set_x:N = \l__hwexam_assign_id_str,
7432 number .int_set:N = \l__hwexam_inclassign_number_int,
7433 title .tl_set:N = \l__hwexam_inclassign_title_tl,
7434 type .tl_set:N = \l__hwexam_inclassign_type_tl,
7435 given .tl_set:N = \l__hwexam_inclassign_given_tl,
7436 due .tl_set:N = \l__hwexam_inclassign_due_tl,
7437 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
7438 }
7439 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
7440 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
7441 \tl_clear:N \l__hwexam_inclassign_title_tl
7442 \tl_clear:N \l__hwexam_inclassign_type_tl
7443 \tl_clear:N \l__hwexam_inclassign_given_tl
7444 \tl_clear:N \l__hwexam_inclassign_due_tl
7445 \str_clear:N \l__hwexam_inclassign_mhrepos_str
7446 \keys_set:nn { hwexam / inclassignment }{ #1 }
7447 }
7448 \__hwexam_inclassignment_args:n {}
7449
7450 \newcommand\inputassignment[2][{}]{

```

```

7451 \_hwexam_inclassnment_args:n { #1 }
7452 \str_if_empty:NTF \l__hwexam_inclassn_mhrepos_str {
7453   \input{#2}
7454 }{
7455   \stex_in_repository:nn{\l__hwexam_inclassn_mhrepos_str}{
7456     \input{\mhp{path}\l__hwexam_inclassn_mhrepos_str}{#2}}
7457 }
7458 }
7459 \_hwexam_inclassnment_args:n {}
7460 }
7461 \newcommand\includeassignment[2][ ]{
7462   \newpage
7463   \inputassignment[#1]{#2}
7464 }

```

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

## 41.4 Typesetting Exams

\quizheading

```

7465 \ExplSyntaxOff
7466 \newcommand\quizheading[1]{%
7467   \def\@tas{#1}%
7468   \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
7469   \ifx\@tas\@empty\else%
7470     \noindent TA:~\@for\@I:=\@tas\do{{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
7471   \fi%
7472 }
7473 \ExplSyntaxOn

```

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

\testheading

```

7474
7475 \def\hwexamheader{\input{hwexam-default.header}}
7476
7477 \def\hwexamminutes{
7478   \tl_if_empty:NTF \testheading@duration {
7479     {\testheading@min}~\hwexam@minutes@kw
7480   }{
7481     \testheading@duration
7482   }
7483 }
7484
7485 \keys_define:nn { hwexam / testheading } {
7486   min .tl_set:N = \testheading@min,
7487   duration .tl_set:N = \testheading@duration,
7488   reqpts .tl_set:N = \testheading@reqpts,
7489   tools .tl_set:N = \testheading@tools
7490 }
7491 \cs_new_protected:Nn \_hwexam_testheading_args:n {
7492   \tl_clear:N \testheading@min
7493   \tl_clear:N \testheading@duration

```

```

7494 \tl_clear:N \testheading@reqpts
7495 \tl_clear:N \testheading@tools
7496 \keys_set:nn { hwexam / testheading }{ #1 }
7497 }
7498 \newenvironment{testheading}[1][ ]{
7499 \__hwexam_testheading_args:n{ #1 }
7500 \newcount\check@time\check@time=\testheading@min
7501 \advance\check@time by -\theassignment@totalmin
7502 \newif\if@bonuspoints
7503 \tl_if_empty:NTF \testheading@reqpts {
7504 \@bonuspointsfalse
7505 }{
7506 \newcount\bonus@pts
7507 \bonus@pts=\theassignment@totalpts
7508 \advance\bonus@pts by -\testheading@reqpts
7509 \edef\bonus@pts{\the\bonus@pts}
7510 \@bonuspointstrue
7511 }
7512 \edef\check@time{\the\check@time}
7513
7514 \makeatletter\hwexamheader\makeatother
7515 }{
7516 \newpage
7517 }

```

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

\testspace

```

7518 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

7519 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

7520 \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.

```

7521 <@=problems>
7522 \renewcommand\@problem[3]{
7523 \stepcounter{assignment@probs}
7524 \def\__problemspts{#2}
7525 \ifx\__problemspts\@empty\else
7526 \addtocounter{assignment@totalpts}{#2}
7527 \fi
7528 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
7529 \xdef\correction@probs{\correction@probs & #1}%
7530 \xdef\correction@pts{\correction@pts & #2}
7531 \xdef\correction@reached{\correction@reached &}

```

```

7532 }
7533 <@@=hwexam>

```

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

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

```

7534 \newcounter{assignment@probs}
7535 \newcounter{assignment@totalpts}
7536 \newcounter{assignment@totalmin}
7537 \def\correction@probs{\correction@probs@kw}
7538 \def\correction@pts{\correction@pts@kw}
7539 \def\correction@reached{\correction@reached@kw}
7540 \stepcounter{assignment@probs}
7541 \newcommand\correction@table{
7542 \resizebox{\textwidth}{!}{%
7543 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
7544 &\multicolumn{\theassignment@probs}{c|}|%|
7545 {\footnotesize\correction@forgrading@kw} &\\ \hline
7546 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
7547 \correction@pts & \theassignment@totalpts & \\ \hline
7548 \correction@reached & & \[.7cm]\hline
7549 \end{tabular}}
7550 </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}

```