

# The sTeX3 Package Collection \*

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

2022-09-27

## Abstract

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

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

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

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

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

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

---

\*Version 3.2 (last revised 2022-09-27)

# Contents

<b>I Manual</b>	<b>STEX _annotate:nnn counterpage ,arabic,11</b>
<b>STEX What is TeX?</b>	<b>STEX _annotate:nnn counterpage ,arabic,22</b>
<b>STEX Setup</b>	<b>3</b>
STEX _Setting up the TeX Package ,arabic,22	STEX _ annotate:nnn countersection ,arabic,11
STEX _Minimize Setup of the TeX Package,22	STEX _ annotate:nnn countersection ,arabic,11,ST
STEX _How to use Setup for the TeX Arabic,22	STEX _ annotate:nnn countersection ,arabic,11,ST
STEX _Setting your Math fonts,arabic,22	STEX _ annotate:nnn countersSTEX _amotate:nn
STEX _Setting up the TeX Header,arabic,22	STEX _ annotate:nnn countersectionSTEX _amotate:nn
STEX _The STEX VS Code Extension,arabic,22	STEX _ annotate:nnn countersSTEX _amotate:nn
STEX _Setting up a counterchapter ,arabic,22	STEX _ annotate:nnn countersSTEX _amotate:nn
STEX _Manual Setup counterchapter ,arabic,22	STEX _ annotate:nnn countersectionSTEX _amotate:nn
STEX _TeX Archives of Manual Setup,arabic,22	STEX _ annotate:nnn countersSTEX _amotate:nn
STEX _Manual Setup for Arabic Doc,arabic,22	STEX _annotate:nnn countersection ,arabic,33,ST
Services . . . . .	STEX _annotate:nn
<b>STEX The STEX Header</b>	<b>STEX _annotate:nnn counterpage ,arabic,77</b>
<b>STEX First STEX: Document</b>	<b>8</b>
STEX _AMDoc,arabic,44	STEX _ annotate:nnn countersection ,arabic,11
STEX _Manual Doc counterchapter ,arabic,44	STEX _ annotate:nnn countersectionSTEX _amotate:nn
<b>STEX Creating STEX Content</b>	<b>14</b>
STEX _How Knowledge is Organized ,arabic,55	STEX _ annotate:nnn countersection ,arabic,11
STEX _STEX Archives counterchapter ,arabic,55	STEX _ annotate:nnn countersection ,arabic,22
STEX _The LaTeX Math Header,arabic,55	STEX _ annotate:nnn countersection ,arabic,22,ST
STEX _The Structure of STEX Chapter,arabic,55	STEX _ annotate:nnn countersection ,arabic,22,ST
STEX _MAATHES in the Chapter,arabic,55	STEX _ annotate:nnn countersection ,arabic,22,ST
STEX _Using Files in STEX Chapters,arabic,55	STEX _ annotate:nnn countersection ,arabic,22,ST
STEX _Module, Symbol and Notation,arabic,55	STEX _ annotate:nnn countersection ,arabic,33
STEX _The module Environment,arabic,55	STEX _ annotate:nnn countersection ,arabic,33,ST
STEX _Declaring New Symbols,arabic,55	STEX _ annotate:nnn countersection ,arabic,33,ST
Operator Notations . . . . .	24
STEX _Argument Modes counterchapter ,arabic,55	STEX _ annotate:nnn countersection ,arabic,33,ST
Mode-b Arguments . . . . .	24
Mode-a Arguments . . . . .	25
Mode-B Arguments . . . . .	26
STEX _Typed and Default Commands,arabic,55	STEX _ annotate:nnn countersection ,arabic,33,ST
STEX _Direct commands in a Chapter ,arabic,55	STEX _ annotate:nnn countersection ,arabic,33,ST
STEX _Manual nnn counterchapter ,arabic,55	STEX _ annotate:nnn countersection ,arabic,33,ST
STEX _Manual Sequence counterchapter ,arabic,55	STEX _ annotate:nnn countersection ,arabic,33,ST
STEX _Module Inheritance and Structure,arabic,55	STEX _ annotate:nnn countersection ,arabic,44
STEX _Annotating quality of modules,arabic,55	STEX _ annotate:nnn countersection ,arabic,44,ST
STEX _Simple Inheritance and Spaces,arabic,55	STEX _ annotate:nnn countersection ,arabic,44,ST
STEX _The LaTeX structure Chapter,arabic,55	STEX _ annotate:nnn countersection ,arabic,44,ST
STEX _The LaTeX module Environment,arabic,55	STEX _ annotate:nnn countersection ,arabic,44,ST

<code>\theintermediatechapterarabic</code>	55	<code>\TeX . annotate:nnn counters</code>	<code>\TeX _annotate:nn</code>
<code>\PreambleSymbols</code>	( <code>\thechapter</code> , <code>\thechapter</code> )	<code>\TeX . annotate:nnn countersections</code>	<code>\TeX _bian</code>
<code>\singl\TeX Symbols</code>	chapter ,arabic,66		42
<code>\symoffanditsvariant</code>	chapter ,arabic,66	<code>\TeX . annotate:nnn countersection</code>	42arabic,11
<code>\Marking&amp;ImpExtatChapter</code>	Chapter,English,66	<code>\TeX . annotate:nnn countersection</code>	43arabic,22
<code>\TeX State</code>	nnn counterchapter ,arabic,77		47
<code>\Definitions,illustrates,Examples</code>	Page,77	<code>\TeX . annotate:nnn countersection</code>	47arabic,11
<code>\Proof</code>	state:nnn.counterchapter ,arabic,77	<code>\TeX . annotate:nnn countersection</code>	50arabic,22
<code>\Highlighting and Presentation</code>	Custom,77	<code>\TeX . annotate:nnn countersection</code>	55arabic,33
<code>\CrossReferences</code>	counterchapter ,arabic,88		57
<code>\Additional Packages</code>	counterchapter ,arabic,99		59
<code>\tikzinputintheading&amp;tikzorder</code>	arabic,99	<code>\TeX . annotate:nnn countersection</code>	59arabic,11
<code>\ModulateDocument&amp;Structure</code>	ing,arabic,99	<code>\TeX . annotate:nnn countersection</code>	60arabic,22
<code>\Annotation</code>	counterchapter ,arabic,99	<code>\TeX . annotate:nnn counterse</code>	60on ,arabic,22
<code>\Package Options</code>	counterchapter ,arabic,99	<code>\TeX . annotate:nnn counterse</code>	60on ,arabic,22
<code>\Document Fragments</code>	chapter ,arabic,99	<code>\TeX . annotate:nnn counterse</code>	60on ,arabic,22
<code>\Ending Documents&amp;Chapter</code>	arabic,99	<code>\TeX . annotate:nnn counterse</code>	62on ,arabic,22
<code>\Global Document Variables</code>	er ,arabic,99	<code>\TeX . annotate:nnn counterse</code>	62on ,arabic,22
<code>\Slide&amp;Table&amp;Cons&amp;Notes</code>	apter ,arabic,99	<code>\TeX . annotate:nnn countersection</code>	62arabic,33
<code>\Annotation</code>	counterchapter ,arabic,99	<code>\TeX . annotate:nnn counterse</code>	62on ,arabic,33
<code>\Package Options</code>	counterchapter ,arabic,99	<code>\TeX . annotate:nnn counterse</code>	63on ,arabic,33
<code>\Notes&amp;Table Slides</code>	counterchapter ,arabic,99	<code>\TeX . annotate:nnn counterse</code>	63on ,arabic,33
<code>\Customizing Header&amp;Page</code>	arabic,99	<code>\TeX . annotate:nnn counterse</code>	64on ,arabic,33
<code>\Frontmatter</code>	counterchapter ,arabic,99	<code>\TeX . annotate:nnn counterse</code>	65on ,arabic,33
<code>\Footnotes</code>	nnn.counterchapter ,arabic,99	<code>\TeX . annotate:nnn counterse</code>	66on ,arabic,33
<code>\Representing Problems&amp;Solutions</code>	99	<code>\TeX . annotate:nnn countersection</code>	67arabic,44
<code>\Annotation</code>	counterchapter ,arabic,99	<code>\TeX . annotate:nnn counterse</code>	67on ,arabic,44
<code>\Problems&amp;Solutions</code>	chapter ,arabic,99	<code>\TeX . annotate:nnn counterse</code>	67on ,arabic,44
<code>\Macros for Axiom Values</code>	arabic,99	<code>\TeX . annotate:nnn counterse</code>	69on ,arabic,44
	Multiple Choice Blocks . . . . .		69
	Filling-In Concrete Solutions . . . . .		70
<code>\Inclutign Problems</code>	counterchapter ,arabic,99	<code>\TeX . annotate:nnn counterse</code>	71on ,arabic,44
<code>\Testing and Spacing</code>	chapter ,arabic,99	<code>\TeX . annotate:nnn counterse</code>	72on ,arabic,44
<code>\Homework&amp;Quizzes&amp;Exercises</code>	arabic,99	<code>\TeX . annotate:nnn countersection</code>	72arabic,55
<code>\Annotation</code>	counterchapter ,arabic,99	<code>\TeX . annotate:nnn counterse</code>	72on ,arabic,55
<code>\Package Options</code>	counterchapter ,arabic,99	<code>\TeX . annotate:nnn counterse</code>	72on ,arabic,55
<code>\Assignments</code>	counterchapter ,arabic,99	<code>\TeX . annotate:nnn counterse</code>	73on ,arabic,55
<code>\Inclutign Assignments</code>	chapter ,arabic,99	<code>\TeX . annotate:nnn counterse</code>	73on ,arabic,55
<code>\Typesetting Exams</code>	chapter ,arabic,99	<code>\TeX . annotate:nnn counterse</code>	73on ,arabic,55

## II Documentation

75

<b>10</b>	<b>sTeX-Basics</b>	<b>76</b>
10.1	Macros and Environments . . . . .	76
10.1.1	HTML Annotations . . . . .	76
10.1.2	Babel Languages . . . . .	77
10.1.3	Auxiliary Methods . . . . .	77
<b>11</b>	<b>sTeX-MathHub</b>	<b>78</b>
11.1	Macros and Environments . . . . .	78
11.1.1	Files, Paths, URIs . . . . .	78
11.1.2	MathHub Archives . . . . .	79
11.1.3	Using Content in Archives . . . . .	80
<b>12</b>	<b>sTeX-References</b>	<b>81</b>
12.1	Macros and Environments . . . . .	81
12.1.1	Setting Reference Targets . . . . .	81
12.1.2	Using References . . . . .	82
<b>13</b>	<b>sTeX-Modules</b>	<b>83</b>
13.1	Macros and Environments . . . . .	83
13.1.1	The <code>smodule</code> environment . . . . .	85
<b>14</b>	<b>sTeX-Module Inheritance</b>	<b>87</b>
14.1	Macros and Environments . . . . .	87
14.1.1	SMS Mode . . . . .	87
14.1.2	Imports and Inheritance . . . . .	88
<b>15</b>	<b>sTeX-Symbols</b>	<b>90</b>
15.1	Macros and Environments . . . . .	90
<b>16</b>	<b>sTeX-Terms</b>	<b>92</b>
16.1	Macros and Environments . . . . .	92
<b>17</b>	<b>sTeX-Structural Features</b>	<b>94</b>
17.1	Macros and Environments . . . . .	94
17.1.1	Structures . . . . .	94
<b>18</b>	<b>sTeX-Statements</b>	<b>95</b>
18.1	Macros and Environments . . . . .	95
<b>19</b>	<b>sTeX-Proofs: Structural Markup for Proofs</b>	<b>96</b>
<b>20</b>	<b>sTeX-Metatheory</b>	<b>97</b>
20.1	Symbols . . . . .	97
<b>III</b>	<b>Extensions</b>	<b>98</b>
<b>21</b>	<b>Tikzinput: Treating TIKZ code as images</b>	<b>99</b>
21.1	Macros and Environments . . . . .	99
<b>22</b>	<b>document-structure: Semantic Markup for Open Mathematical Documents in L<sup>A</sup>T<sub>E</sub>X</b>	<b>100</b>

<b>23</b>	<b>NotesSlides – Slides and Course Notes</b>	<b>101</b>
<b>24</b>	<b>problem.sty: An Infrastructure for formatting Problems</b>	<b>102</b>
<b>25</b>	<b>hwexam.sty/cls: An Infrastructure for formatting Assignments and Exams</b>	<b>103</b>
<b>IV</b>	<b>Implementation</b>	<b>104</b>
<b>26</b>	<b>STEX-Basics Implementation</b>	<b>105</b>
26.1	The STEXDocument Class . . . . .	105
26.2	Preliminaries . . . . .	106
26.3	Messages and logging . . . . .	106
26.4	HTML Annotations . . . . .	107
26.5	Babel Languages . . . . .	109
26.6	Persistence . . . . .	111
26.7	Auxiliary Methods . . . . .	111
<b>27</b>	<b>STEX-MathHub Implementation</b>	<b>115</b>
27.1	Generic Path Handling . . . . .	115
27.2	PWD and kpsewhich . . . . .	117
27.3	File Hooks and Tracking . . . . .	118
27.4	MathHub Repositories . . . . .	119
27.5	Using Content in Archives . . . . .	124
<b>28</b>	<b>STEX-References Implementation</b>	<b>129</b>
28.1	Document URIs and URLs . . . . .	129
28.2	Setting Reference Targets . . . . .	131
28.3	Using References . . . . .	134
<b>29</b>	<b>STEX-Modules Implementation</b>	<b>141</b>
29.1	The smodule environment . . . . .	145
29.2	Invoking modules . . . . .	151
<b>30</b>	<b>STEX-Module Inheritance Implementation</b>	<b>154</b>
30.1	SMS Mode . . . . .	154
30.2	Inheritance . . . . .	159
<b>31</b>	<b>STEX-Symbols Implementation</b>	<b>164</b>
31.1	Symbol Declarations . . . . .	164
31.2	Notations . . . . .	172
31.3	Variables . . . . .	182
<b>32</b>	<b>STEX-Terms Implementation</b>	<b>191</b>
32.1	Symbol Invocations . . . . .	191
32.2	Terms . . . . .	199
32.3	Notation Components . . . . .	204
32.4	Variables . . . . .	206
32.5	Sequences . . . . .	209

<b>33</b>	<b>STeX-Structural Features Implementation</b>	<b>210</b>
33.1	Imports with modification . . . . .	211
33.2	The feature environment . . . . .	219
33.3	Structure . . . . .	219
<b>34</b>	<b>STeX-Statements Implementation</b>	<b>231</b>
34.1	Definitions . . . . .	231
34.2	Assertions . . . . .	237
34.3	Examples . . . . .	240
34.4	Logical Paragraphs . . . . .	243
<b>35</b>	<b>The Implementation</b>	<b>248</b>
35.1	Proofs . . . . .	248
<b>36</b>	<b>STeX-Others Implementation</b>	<b>257</b>
<b>37</b>	<b>STeX-Metatheory Implementation</b>	<b>259</b>
<b>38</b>	<b>Tikzinput Implementation</b>	<b>262</b>
<b>STeX Document structure and implementation</b>	<b>265</b>	
STeX_PackageOptions	underchapter ,arabic,3939,STeX_annotate:nnn countersect	265,arabic,11
STeX_DocumentStructure	underchapter ,arabic,3939,STeX_annotate:nnn countersect	266,arabic,22
STeX_AnnotatedBackmatter	underchapter ,arabic,3939,STeX_annotate:nnn countersect	270,arabic,33
STeX_GlobalVariables	underchapter ,arabic,3939,STeX_annotate:nnn countersect	272,arabic,44
<b>40</b>	<b>NotesSlides – Implementation</b>	<b>273</b>
40.1	Class and Package Options . . . . .	273
40.2	Notes and Slides . . . . .	275
40.3	Header and Footer Lines . . . . .	279
40.4	Frame Images . . . . .	281
40.5	Sectioning . . . . .	282
40.6	Excursions . . . . .	285
<b>41</b>	<b>The Implementation</b>	<b>287</b>
41.1	Package Options . . . . .	287
41.2	Problems and Solutions . . . . .	288
41.3	Markup for Added Value Services . . . . .	295
41.4	Multiple Choice Blocks . . . . .	295
41.5	Filling in Concrete Solutions . . . . .	296
41.6	Including Problems . . . . .	297
41.7	Reporting Metadata . . . . .	298
41.8	Testing and Spacing . . . . .	299
<b>42</b>	<b>Implementation: The hwexam Package</b>	<b>301</b>
42.1	Package Options . . . . .	301
42.2	Assignments . . . . .	302
42.3	Including Assignments . . . . .	305
42.4	Typesetting Exams . . . . .	306
42.5	Leftovers . . . . .	STeX_annotate:nnn



# Part I

## Manual



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



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



# Chapter 1

## What is sTeX?

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

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

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

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

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

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

# Chapter 2

## Setup

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

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

Both are legitimate and useful. The first requires a significantly smaller tool-chain, so we describe it first. The second requires a much more substantial toolchain of knowledge management systems.

Luckily, the  $\text{sTeX}$ -IDE will take care of much of the setup required for the full toolchain, if you are willing to use it.

### 2.1 Setting up the $\text{sTeX}$ Package

#### 2.1.1 Minimal Setup for the $\text{sTeX}$ Package

In the best of all worlds, there is no setup, as you already have a new version of  $\text{\TeX}$ Live on your system as a  $\text{\LaTeX}$  enthusiast. If not now is the time to install it; see [TL]. You can usually update  $\text{\TeX}$ Live via a package manager or the  $\text{\TeX}$ Live manager **tlmgr**.  $\text{sTeX}$  requires a  $\text{\TeX}$  kernel newer than February 2022.

Alternatively, you can install  $\text{sTeX}$  from CTAN, the Comprehensive  $\text{\TeX}$  Archive Network; see [ST] for details. We assume you have the  $\text{sTeX}$  package in at least version 3.2 (September 2022).

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

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

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

and keep it updated by pulling updates via `git pull` in the cloned  $\text{sTeX}$  directory. Make sure to either clone the  $\text{sTeX}$  repository into a local `texmf-tree` or to update your `TEXINPUTS` environment variable, e.g. by placing the following line in your `.bashrc`:

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

### 2.1.3 Setting your MathHub Directory

One of  $\text{\LaTeX}$ 's features is a proper *module system* of interconnected document snippets for mathematical content. Analogously to *object-oriented programming*, it allows for “object-oriented mathematics” via individual combinable and, importantly, *reusable* modules, developed collaboratively.

To make use of such modules, the  $\text{\LaTeX}$  system needs to be told where to find them. There are several ways to do so (see [subsection  \$\text{\LaTeX}\$  \\_annotate:nnn counterchapter ,arabic,55](#), [subsection  \$\text{\LaTeX}\$  \\_annotate:nnn countersection ,arabic,22](#), [subsection  \$\text{\LaTeX}\$  \\_annotate:nnn countersubsection ,arabic,11](#)), but the most convenient way to do so is via a system variable.

To do so, create a directory **MathHub** somewhere on your local file system and set the environment variable **MATHHUB** to the file path to that directory.

In linux, you can do so by writing

```
export MATHHUB="/path/to/your/MathHub"
```

in your `~/.profile` (for all shells) or `~/.bashrc` (for the bash terminal only) file.

## 2.2 Setting up the $\text{\LaTeX}$ IDE

The  $\text{\LaTeX}$  IDE consists of two components using the *Language Server Protocol (LSP)*: A *client* in the form of a VSCode extension, and a *server* included in the MMT system. Installing the extension will open up a setup routine that will guide you through the rest.

### 2.2.1 The $\text{\LaTeX}$ VSCode Extension

If you have not already, you should first install the VSCode editor available at <https://code.visualstudio.com/>.

Next, open VSCode and install the  $\text{\LaTeX}$  extension by clicking on the *extensions* menu on the very left of the VSCode window and searching for “ $\text{\LaTeX}$ ” in the “*Search Extensions in Marketplace*” field, as in [Figure 1](#), and clicking the *Install*-button of the  $\text{\LaTeX}$  extension by KWARC.

### 2.2.2 Setting up Mmt

Next, open any directory (**File**  $\rightarrow$  **Open Folder...**) that contains a `.tex`-file, and a setup window as in [Figure 2](#) will pop up. Click on the highlighted link ‘*here*’ and download the latest version of the `MMT.jar` file (at least version 23.0.0) anywhere you like. Then click the “*Browse...*”-button and select your freshly downloaded `MMT.jar`.

If you have already set a system variable for your MathHub-directory, you are now done and can click “*Finish*”. If you have not, you can now also enter a directory path in the lower text field, and the VSCode extension will attempt to globally set one up for you, depending on your operating system.

Once you click “*Finish*”, the client will connect to <https://stexmmt.mathhub.info/:stex>, query for available archives, download the core libraries required for all (or most) semantic services (`MMT/urtheories` and `stex/meta-inf`) and set up  $\text{\LaTeX}$  for you automatically.

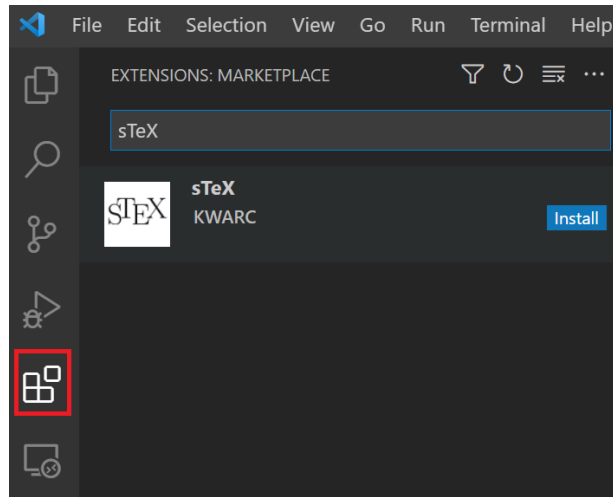


Figure 1: Installing the  $\text{sTeX}$  extension for VSCode

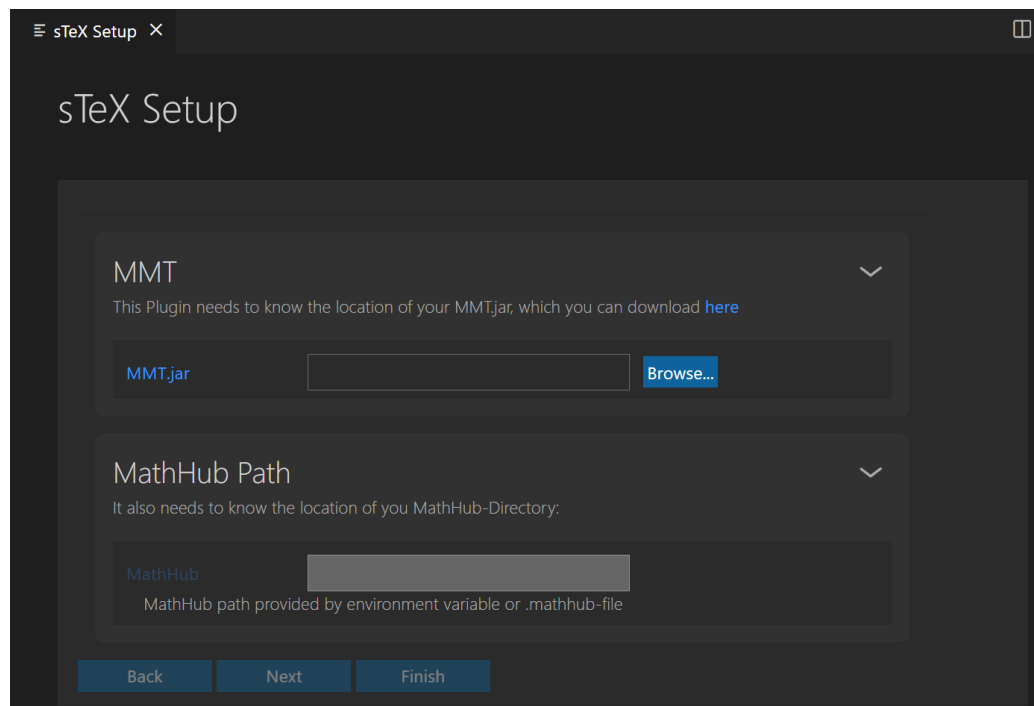


Figure 2:  $\text{sTeX}$  Setup Routine

## 2.3 Manual Setup

In lieu of using the  $\text{\S}\text{\TeX}$  IDE, we can do the following:

### 2.3.1 $\text{\S}\text{\TeX}$ Archives (Manual Setup)

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

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

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

Note that  $\text{\S}\text{\TeX}$  archives often depend on other archives, thus you should be prepared to clone these as well – e.g. if `pdflatex` reports missing files. To make sure that  $\text{\S}\text{\TeX}$  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  \$\text{\S}\text{\TeX}\$  \\_annotate:nnn counterchapter ,arabic,55.](#) [\$\text{\S}\text{\TeX}\$  \\_annotate:nnn countersection ,arabic,22\).](#)

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

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

Foregoing on the  $\text{\S}\text{\TeX}$  IDE, we will need several additional (on top of the minimal setup above) pieces of software; namely:

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

Following the setup routine (Step 3) will entail designating a **MathHub**-directory on your local file system, where the MMT system will look for  $\text{\S}\text{\TeX}$ /MMT content archives.

- **$\text{\S}\text{\TeX}$  Archives** If we only care about  $\text{\LaTeX}$  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)  $\text{\S}\text{\TeX}$  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 smgglom` will download all `smgglom` archives.

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

## Chapter 3

# The $\text{\TeX}$ IDE

## Chapter 4

# A First sTeX Document

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

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

Compiling this document with pdflatex should yield the output

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

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

**Theorem 0.2.** The [geometric series converges](#) towards 1.

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

**Remark 4.0.1:**

Note that all of the highlighting, tooltips, coloring and the environment headers come from `stexthm` – by default, the amount of additional packages loaded is kept to a minimum and all the presentations can be customized, see [section `\TeX \_annotate:nnn counterchapter ,arabic,77.\TeX \_annotate:nnn countersection ,arabic,33`](#).

Let’s investigate this document in detail to understand the respective parts of the `\TeX` markup infrastructure:

```
smodule (env.) \begin{smodule}{GeometricSeries}
               ...
               \end{smodule}
```

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

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

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

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

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

```
\usemodule
```

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



---

---

`\symdef`

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

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

---

---

`\comp`

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

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

What follows are two *TeX-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 *LaTeX* templates predefine environments like `definition` or `theorem` with different syntax, we use `sdefinition`, `sassertion`, `sexample` etc. instead. You can customize these environments to e.g. simply wrap around some predefined `theorem`-environment. That way, we can still use `sassertion` to provide semantic information, while being fully compatible with (and using the document presentation of) predefined environments.

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

---

---

`\symname`

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

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

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

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

---

---

`\symref`

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

---

<code>\definame</code>	The <code>\definame{geometricSeries}</code> ...
<code>\definiendum</code>	The <code>sdefinition</code> -environment provides two additional macros, <code>\definame</code> and <code>\definiendum</code> which behave similarly 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.

---

```

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

```

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

---

<code>\svar</code>	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> .
--------------------	--

---



---

<code>\definiens</code>	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.
-------------------------	--

---

## 4.1 OMDoc/xhtml Conversion

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

**TODO VSCode Plugin**

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

```

<mrow resource="" property="stex:definiens">
  <mrow resource="...?series?infinitesum" property="stex:OMBIND">
    <munderover displaystyle="true">
      <mo resource="...?series?infinitesum" property="stex:comp"> $\Sigma$ </mo>
      <mrow>
        <mrow resource="1" property="stex:arg">
          <mi resource="var://n" property="stex:OMV">n</mi>
        </mrow>
        <mo resource="...?series?infinitesum" property="stex:comp">=</mo>
        <mi resource="2" property="stex:arg">1</mi>
      </mrow>
    </munderover>
  </mrow>
</mrow>

```

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

```

</mrow>
<mi resource="...?series?infinitesum" property="stex:comp">∞</mi>
</munderover>
<mrow resource="3" property="stex:arg">
<mfrac resource="...?realarith?division#frac#" property="stex:OMA">
<mi resource="1" property="stex:arg">1</mi>
<mrow resource="2" property="stex:arg">
<msup resource="...realarith?exponentiation" property="stex:OMA">
<mi resource="1" property="stex:arg">2</mi>
<mrow resource="2" property="stex:arg">
<mi resource="var://n" property="stex:OMV">n</mi>
</mrow>
</msup>
</mrow>
</mfrac>
</mrow>
</mrow>
</mrow>

```

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

```

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

```

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

#### Remark 4.1.1:

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.

## 4.2 Mmt/OMDoc Conversion

Another way to convert our document to *actual* MMT/OMDOC is to put it in an  $\text{\LaTeX}$  **archive** (see [section  \$\text{\LaTeX}\$  \\_\\_annotate:nnn counterchapter ,arabic,55. \$\text{\LaTeX}\$  \\_\\_annotate:nnn countersection ,arabic,22](#)) and have MMT take care of everything.

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

## Chapter 5

# Creating sTeX Content

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

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

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

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

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

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

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

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

## 5.1 How Knowledge is Organized in sTeX

sTeX content is organized on multiple levels:

1. sTeX **archives** (see [section sTeX \\_annotate:nnn counterchapter ,arabic,55.sTeX \\_annotate:nnn countersection ,arabic,22](#)) contain individual `.tex`-files.
2. These may contain sTeX **modules**, introduced via `\begin{smodule}{ModuleName}`.

3. Modules contain  $\text{\S}\text{\TeX}$  **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.
4.  $\text{\S}\text{\TeX}$  **expressions** finally are built up from usages of semantic macros.

- $\text{\S}\text{\TeX}$  archives are simultaneously MMT archives, and the same directory structure is consequently used.
  - $\text{\S}\text{\TeX}$  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 [RK13].
  - Symbol declarations induce OMDoc/MMT *constants*, with optional (formal) *type* and *definiens* components.
  - Finally,  $\text{\S}\text{\TeX}$  expressions are converted to OMDoc/MMT terms, which use the abstract syntax (and XML encoding) of OPENMATH [Bus+04].

$\hookrightarrow M \rightarrow$   
 $\hookrightarrow M \rightarrow$   
 $\hookrightarrow T \rightarrow$

## 5.2 $\text{\S}\text{\TeX}$ Archives

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

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

1. If the  $\text{\S}\text{\TeX}$  package is loaded with the option `mathhub=/path/to/mathhub`, then  $\text{\S}\text{\TeX}$  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  $\text{\S}\text{\TeX}$ -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,  $\text{\S}\text{\TeX}$  will attempt to retrieve the system variable `MATHHUB`, assuming it will point to the local MathHub-directory. Since this variant needs setting up only *once* and is machine-specific (rather than defined in tex code), it is compatible with collaborating and sharing tex content, and hence recommended.
4. Finally, if all else fails,  $\text{\S}\text{\TeX}$  will look for a file `~/.stex/mathhub.path`. If this file exists,  $\text{\S}\text{\TeX}$  will assume that it contains the path to the local MathHub-directory. This method is recommended on systems where it is difficult to set environment variables.

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

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

Each such archive needs two subdirectories:

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

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

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

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

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

### 5.2.3 MANIFEST.MF-Files

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

```
id: smglom/calculus
source-base: http://mathhub.info/smgglom/calculus
narration-base: http://mathhub.info/smgglom/calculus
dependencies: smglom/arithmetic,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`.

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

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

---

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

---

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

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

---

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

---

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

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



---

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

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

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

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

---

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

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

**Remark 5.2.1:**

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

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

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

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

## 5.3 Module, Symbol and Notation Declarations

### 5.3.1 The `smodule`-Environment

`smodule` (*env.*) A new module is declared using the basic syntax

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

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

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

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

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

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

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

#### Example 1

Input:

```

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

```

Output:

Hello World

---

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

For example:

#### Example 2

Input:

```

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

```

Output:

```

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

```

### 5.3.2 Declaring New Symbols and Notations

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

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

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

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

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

#### Example 3

Input:

```

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

```

Output:

```

Given a foo, we can...

```

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

#### Example 4

Input:

```

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

```

Output:

```

this is a symbol taking two arguments.

```

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

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

---

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

#### Example 5

Input:

```

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

```

Output:

```

First: a; Second: b

```

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

---

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

We can introduce a new notation **highlight** for  $\binarysymbol$  that fixes this flaw, which we can subsequently use with  $\binarysymbol[\text{highlight}]$ :

### Example 6

Input:

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

Output:

First:  $a$ ; Second:  $b$



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



Note that it is required that

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

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

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

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

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

---

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

### Example 7

Input:

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

Output:

```
1.: a; 2.: b
```

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

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

---

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

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

---

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

## Operator Notations

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

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

### Example 8

Input:

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

Output:

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

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

### 5.3.3 Argument Modes

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

#### Mode-b Arguments

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

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

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

### Example 9

Input:

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

Output:

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

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

### Mode-a Arguments

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

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

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

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

### Example 10

Input:

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

Output:

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



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

#### Example 11

Input:

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

Output:

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

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

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

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

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

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

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

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

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

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

#### Mode-B Arguments

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

#### Example 12

Input:

```

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

```

Output:

$\forall x,y,z.P$

### 5.3.4 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  $x \mapsto x+1$

### 5.3.5 Precedences and Automated Bracketing

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

#### Example 15

Input:

```

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

```

Output:

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

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

#### Example 16

Input:

```

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

```

Output:

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

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

### Example 17

Input:

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

Output:

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

but we can also do better by supplying *precedences* and have  $\TeX$  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, e.g. for bracket-like notations such as intervals – 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.

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

When  $\text{\S}\text{\TeX}$  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.  $\text{\S}\text{\TeX}$  starts out with  $p_d = \text{\code{\infprec}}$ .
2.  $\text{\S}\text{\TeX}$  encounters `\addition` with  $p_{op} = 100$ . Since  $100 \not> \text{\code{\infprec}}$ , it inserts no parentheses.
3. Next,  $\text{\S}\text{\TeX}$  encounters the two arguments for `\addition`. Both have no specifically provided argument precedence, so  $\text{\S}\text{\TeX}$  uses  $p_d = p_{op} = 100$  for both and recurses.
4. Next,  $\text{\S}\text{\TeX}$  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  $\text{\S}\text{\TeX}$  again inserts no parentheses.
6. Since the notation of `\multiplication` has no explicitly set argument precedences,  $\text{\S}\text{\TeX}$  uses the operator precedence for all arguments of `\multiplication`, hence sets  $p_d = p_{op} = 50$  and recurses.
7. Next,  $\text{\S}\text{\TeX}$  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  $\text{\S}\text{\TeX}$  to insert parentheses, and we proceed as before.

### 5.3.6 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  $\text{\TeX}$  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;\neginfprec
7 ]{\comp{f}#1}
8 \vardef{varn}[name=n,type=\Nat]{\comp{n}}
9 \vardef{varx}[name=x,type=\Nat]{\comp{x}}
10
11 Given a function $\varf!:\funtype{\Nat}{\Nat}$,
12 by $\addition{\varf!,\varn}$ we mean the function\rustexBREAK
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

### 5.3.7 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 T<sub>E</sub>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. Note that both the starting as well as the ending index may be variables.

This is best shown by example:

### Example 20

Input:

```

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

$a_1^1, \dots, a_n^m$  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:

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

## 5.4 Module Inheritance and Structures

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

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

### 5.4.1 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:  
 $\hookrightarrow$  `\begin{smodule}[lang=<lang>]{Foo}` generates a theory `some/namespace?Foo` that only contains the “formal” part of the module – i.e. exactly the content that is exported when using `\importmodule`.  
 $\rightsquigarrow$  Additionally, MMT generates a *language theory* `some/namespace/Foo?<lang>` that includes `some/namespace?Foo` and contains all the other document content – variable declarations, includes for each `\usemodule`, etc.

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



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

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

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

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

## 5.4.2 Simple Inheritance and Namespaces

---

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

---

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

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



Ideally,  $\text{\LaTeX}$  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  $\text{\LaTeX}$  can resolve them accordingly. Largely, users need not concern themselves with namespaces at all, but for completeness sake, we describe how they are constructed:

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

In other words: outside of archives, the namespace corresponds to the file URI



with the filename dropped iff it is equal to the module name, and ignoring the (optional) language suffix.

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



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

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

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

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

---

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



For persistency reasons, everything in an `\STEXexport` is digested by T<sub>E</sub>X in the L<sup>A</sup>T<sub>E</sub>X3-category code scheme. This means that the characters `_` and `:` are considered *letters* and valid parts of control sequence names, and space characters are



ignored entirely. For spaces, use the character `~` instead, and keep in mind, that if you want to use subscripts, you should use `\c_math_subscript_token` instead of `_`!

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

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

### 5.4.3 The `mathstructure` Environment

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

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

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

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

How this works is again best demonstrated by example:

#### Example 24

Input:

```

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

Output:

A *monoid* is...

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

### Example 25

Input:

```

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

Output:

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

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

### Example 26

Input:

```

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

Output:

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

---

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

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

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

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

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

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

---

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

This allows us to do things like:

### Example 27

Input:

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

Output:

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

.

and

### Example 28

Input:

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

Output:

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

.

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

**usestructure** (*env.*) The `usestructure{<struct>}` environment is used in multilingual settings as a parallel to the `mathstructure`. It opens a group and then issues a `\usemodule{.../<struct>-structure}` that gives the body access to all the semantic macros in the referenced structure.

### 5.4.4 The copymodule Environment

TODO: explain

Given modules:

### Example 29

Input:

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

Output:

.

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

### Example 30

Input:

```

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

```

Output:

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

TODO: explain donotclone

## 5.4.5 The interpretmodule Environment

TODO: explain

### Example 31

Input:

```

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

```

Output:

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

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

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

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

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



## Chapter 6

# Using $\text{\TeX}$ Symbols

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

### 6.1 `\symref` and its variants

---

<code>\symref</code> <code>\symname</code>	We have already seen <code>\symname</code> and <code>\symref</code> , the latter being the more general. <code>\symref{&lt;symbolname&gt;}{&lt;code&gt;}</code> marks-up <code>&lt;code&gt;</code> as referencing <code>&lt;symbolname&gt;</code> . Since quite often, the <code>&lt;code&gt;</code> should be (a variant of) the name of the symbol anyway, we also have <code>\symname{&lt;symbolname&gt;}</code> .
---	--

---

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

#### Example 32

Input:

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

Output:

A natural number is...

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

---

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

### Example 33

Input:

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

Output:

Natural numbers are...



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

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

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

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

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

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

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

### Example 34

Input:

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

Output:

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

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

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



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

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

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

### Example 35

Input:

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

Output:

```
Addition is...
```

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

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

### Example 36

Input:

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

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

Output:

adding  $k$  yields...

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

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

### Example 37

Input:

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

Output:

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

If we take features like `\inputref` and `\mhinput` (and the `sfragment`-environment, see `subsection STEX _annotate:nnn counterchapter ,arabic,99.STEX _annotate:nnn countersection ,arabic,22.STEX _annotate:nnn countersubsection ,arabic,11`) seriously, and build large documents modularly from individually compiling documents for sections, chapters and so on, cross-referencing becomes an interesting problem.

Say, we have a document `main.tex`, which `\inputrefs` a section `section1.tex`, which references a definition with label `some_definition` in `section2.tex` (subsequently also inputted in `main.tex`). Then the numbering of the definition will depend on the *document context* in which the document fragment `section2.tex` occurs - in `section2.tex` itself (as a standalone document), it might be *Definition 1*, in `main.tex` it might be *Definition 3.1*, and in `section1.tex`, the definition *does not even occur*, so it needs to be referenced by some other text.

What we would want in that instance is an equivalent of `\autoref`, that takes the document context into account to yield something like *Definition 1*, *Definition 3.1* or *Definition 1 in the section on Foo* respectively.

The `\sref` command attempts to do precisely that. Unlike plain `\ref`, `\autoref` etc., `\sref` refers to not just a *label*, but instead a pair consisting of a *label* and the *document* in whose context we want to refer to it. Conversely, every *document* (i.e. standalone compilable `.tex`-file) keeps track of the “names” (*Definition 3.1* etc.) for every label as determined in the context of the document, and stores them in a dedicated file `\jobname.sref`. Additionally, every document has a “reference name” (e.g. “the section on Foo”). This allows us to refer to “label  $x$  in document  $D$ ” to yield “*Definition*

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

1 in the section on Foo”. And of course,  $\text{\TeX}$  can decide based on the current document to either refer to the label by its “full name” or directly as e.g. *Definition 3.1* depending on whether the label occurs in the current document anyway (and link to it accordingly).

For that to work, we need to supply (up to) three pieces of information:

- The *label* of the reference target (e.g. `some_definition`),
- (optionally) the *file*/document containing the reference target (e.g. `section2`). This is not strictly necessary, but allows for additional disambiguation between possibly duplicate labels across files, and
- (optionally) the document context, in which we want to refer to the reference target (e.g. `main`).

Additionally, the document in which we want to reference a label needs a title for external references.

---

```
\sref [archive=<archive1>,file=<file>]
      {\<label>}[archive=<archive2>,in=<document-context>,title=<title>]
```

---

This command references  $\langle label \rangle$  (declared in  $\langle file \rangle$  in  $\langle archive1 \rangle$ ). If the object (section, figure, etc.) with that label occurs ultimately in the same document, `\sref` will ignore the second set of optional arguments and simply defer to `\autoref` if that command exists, or `\ref` if the `hyperref` package is not included.

If the referenced object does *not* occur in the current document however, `\sref` will refer to it by the object’s name as it occurs in the file  $\langle document-context \rangle$  in  $\langle archive2 \rangle$ .

For example, the reference to the `sfragment`-environment above will appear as “subsection 7.2.1 (Introduction) in the  $\text{\TeX}$ 3 manual” if you are reading this in the package documentation for `stex-references` directly, but as a linked “subsection 7.2.1” in the full documentation or manual. This is achieved using

```
\sref [file=stex-document-structure]{sec:ds:intro}[in=../stex-manual,title={the \stex}]
```

For a further example, the following:

### Part III

will say “Part III” (and link accordingly) in the full documentation, and “Part III (Extensions) in the full  $\text{\TeX}$ 3 documentation” everywhere else. This is achieved using

```
\sref [file=../stex-doc]{part:extends}[in=../stex-doc,title={the full \stex}3 document]
```

---

```
\extref [archive=<archive1>,file=<file>]
        {\<label>}[archive=<archive2>,in=<document-context>,title=<title>]}
```

---

The `\extref`-command behaves exactly like `\sref`, but takes *required* the document context argument and will always use it for generating the document text, regardless of whether the label occurs in the current document.

## Chapter 7

# sTeX Statements

### 7.1 Definitions, Theorems, Examples, Paragraphs

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

The corresponding environments for that are:

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

The *presentation* of these environments can be customized to use e.g. predefined theorem-environments, see [section sTeX \\_annotate:nnn counterchapter ,arabic,77.sTeX \\_annotate:nnn countersection ,arabic,33](#) for details.

All of these environments take optional arguments in the form of `key=value`-pairs. Common to all of them are the keys `id=` (for cross-referencing, see [chapter sTeX \\_annotate:nnn counterchapter ,arabic,88](#)), `type=` for customization (see [section sTeX \\_annotate:nnn counterchapter ,arabic,77.sTeX \\_annotate:nnn countersection ,arabic,33](#)) and additional information (e.g. definition principles, “difficulty” etc), as well as `title=` (for giving the paragraph a title), and finally `for=`.

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

#### Example 38

Input:

```

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

```

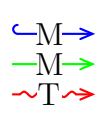
Output:

**Example 7.1.1** (An Example).  $2+3$  is 5,  $2\cdot 3$  is 6.

---

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

---


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

---

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

---

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

### Example 39

Input:

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

```

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

```

Output:

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

**Axiom 7.1.2** (Associativity).  $\circ$  is associative

**Axiom 7.1.3** (Unit).  $x \circ e = x$  for all  $x \in U$

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

EdN:4

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

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

```

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

```

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



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

## 7.2 Proofs

The `stex-proof` package supplies macros and environment that allow to annotate the structure of mathematical proofs in  $\text{\LaTeX}$  document. This structure can be used by MKM systems for added-value services, either directly from the  $\text{\LaTeX}$  sources, or after translation.

Its central component is the `sproof`-environment, whose body consists of:

- *subproofs* via the `subproof`-environment,
- *proof steps* via the `\spfstep`, `\eqstep` `\assumption`, and `\conclude` macros, and
- *comments*, via normal text without special markup.

`sproof`, `subproof` and the various proof step macros take the following optional arguments:

`id` ( $\langle string \rangle$ ) for referencing,

`method` ( $\langle string \rangle$ ) the proof method (e.g. contradiction, induction,...)

`term` ( $\langle token list \rangle$ ) the (ideally semantically-marked up) proposition that is derived/proven by this proof/subproof/proof step.

Additionally, they take one mandatory argument for the document text to be annotated, or (in the case of the environments) as an introductory description of the proof itself. Since the latter often contains the `term` to be derived as text, alternatively to providing it as an optional argument, the mandatory argument can use the `\yield`-macro to mark it up in the text.

The `sproof` and `subproof` environments additionally take two optional arguments:

`for` the symbol identifier/name corresponding to the `sassertion` to be proven. This too subsumes `\yield` and the `term`-argument.

`hide` In the pdf, this only shows the mandatory argument text and hides the body of the environment. In the HTML (as served by MMT), the bodies of all `proof` and `subproof` environments are *collapsible*, and `hide` collapses the body by default.

```

1 \begin{sassertion}[type=theorem,name=sqrt2irr]
2   \conclusion{\irrational{\arg{\realroot{2}}$ is \comp{irrational}}}.
3 \end{sassertion}
4
5 \begin{sproof}[for=sqrt2irr,method=contradiction]{By contradiction}
6   \assumption{Assume \yield{\rational{\arg{\realroot{2}}$ is
7     \comp{rational}}}}
8   \begin{subproof}[method=straightforward]{Then
9     \yield{\$eq{\ratfrac{\intpow{\vara}{2}}{\intpow{\varb}{2}}{2}$
10       for some $\inset{\vara,\varb}\PosInt$ with
11         \coprime{\arg{\vara},\arg{\varb}$ \comp{coprime}}}}
```

<sup>2</sup>Of course,  $\text{\LaTeX}$  can not check that the assertions are the “correct” ones – but if the assertions (both in monoid as well as those for addition and zero) are properly marked up, MMT can. **TODO: should**

```

12 \assumption{By assumption, \yield{there are
13 $\inset{\vara,\varb}\PosInt$ with
14 $\realroot{2}=\ratfrac{\vara}{\varb}$}}
15 \spfstep{wlog, we can assume \coprime{$\arg{\vara},\arg{\varb}$}
16 to be \comp{coprime}}
17 % a comment:
18 If not, reduce the fraction until numerator and denominator
19 are coprime, and let the resulting components be
20 $\vara$ and $\varb$
21 \spfstep{Then \yield{$\eq{\intpow{\ratfrac{\vara}{\varb}}{2}{2}$}}
22 \eqstep{\ratfrac{\intpow{\vara}{2}}{\intpow{\varb}{2}}}}
23 \end{subproof}
24 \begin{subproof}[term=\divides{2}{\vara},method=straightforward]{
25 Then $\vara$ is even}
26 \spfstep{Multiplying the equation by $\intpow{\varb}{2}$ yields
27 $\yield{\eq{\intpow{\vara}{2}}{\inttimes{2}{\intpow{\varb}{2}}}}$}
28 \spfstep[term=\divides{2}{\intpow{\vara}{2}}]{Hence
29 $\intpow{\vara}{2}$ is even}
30 \conclude[term=\divides{2}{\vara}]{Hence $\vara$ is even as well}
31 % another comment:
32 Hint: Think about the prime factorizations of $\vara$ and
33 $\intpow{\vara}{2}$
34 \end{subproof}
35 \begin{subproof}[term=\divides{2}{\varb},method=straightforward,]{
36 Then $\varb$ is also even}
37 \spfstep{Since $\vara$ is even, we have \yield{some $\varc$ $
38 such that $\eq{\inttimes{2}{\varc}}{\vara}$}}
39 \spfstep{Plugging into the above, we get
40 \yield{$\eq{\intpow{\inttimes{2}{\vara}}{2}
41 {\inttimes{2}{\intpow{\varb}{2}}}$}}
42 \eqstep{\inttimes{4}{\intpow{\vara}{2}}}
43 \spfstep{Dividing both sides by $2$ yields
44 \yield{$\eq{\intpow{\varb}{2}}{\inttimes{2}{\intpow{\vara}{2}}}$}}
45 \spfstep[term=\divides{2}{\intpow{\varb}{2}}]{Hence
46 $\intpow{\varb}{2}$ is even}
47 \conclude[term=\divides{2}{\varb}]{Hence $\varb$ is even}
48 % one more comment:
49 By the same argument as above
50 \end{subproof}
51 \conclude[term=\contradiction]{Contradiction to $\vara,\varb$ being
52 \symname{coprime}.}
53 \end{spproof}

```

which will produce:

**Theorem 7.2.1.**  $\sqrt{2}$  is *irrational*.

**Proof:** By contradiction

1. Assume  $\sqrt{2}$  is *rational*
2. Then  $(\frac{a}{b})^2=2$  for some  $a,b \in \mathbb{Z}^+$  with  $a,b$  *coprime*
  - 2.1. By assumption, there are  $a,b \in \mathbb{Z}^+$  with  $\sqrt{2} = \frac{a}{b}$
  - 2.2. wlog, we can assume  $a,b$  to be *coprime*

*If not, reduce the fraction until numerator and denominator are coprime, and let the re-*

sulting components be  $a$  and  $b$

2.3. Then  $(\frac{a}{b})^2=2$

$$= \frac{a^2}{b^2}$$

3. Then  $a$  is even

3.1. Multiplying the equation by  $b^2$  yields  $a^2=2b^2$

3.2. Hence  $a^2$  is even

$\Rightarrow$  Hence  $a$  is even as well

*Hint: Think about the prime factorizations of  $a$  and  $a^2$*

4. Then  $b$  is also even

4.1. Since  $a$  is even, we have some  $c$  such that  $2c=a$

4.2. Plugging into the above, we get  $(2a)^2=2b^2$

$$= 4a^2$$

4.3. Dividing both sides by 2 yields  $b^2=2a^2$

4.4. Hence  $b^2$  is even

$\Rightarrow$  Hence  $b$  is even

*By the same argument as above*

$\Rightarrow$  Contradiction to  $a, b$  being coprime.

□

If we mark all subproofs with `hide`, we will obtain the following instead:

**Theorem 7.2.2.**  $\sqrt{2}$  is irrational.

**Proof:** By contradiction

1. Assume  $\sqrt{2}$  is rational

2. Then  $(\frac{a}{b})^2=2$  for some  $a, b \in \mathbb{Z}^+$  with  $a, b$  coprime

3. Then  $a$  is even

4. Then  $b$  is also even

$\Rightarrow$  Contradiction to  $a, b$  being coprime.

□

However, the hidden subproofs will still be shown in the HTML, only in an expandable section which is collapsed by default.

The above style of writing proofs is usually called *structured proofs*. They have a huge advantage over the traditional purely prosaic style, in that (as the name suggests) the actual *structure* of the proof is made explicit, which almost always makes it considerably more comprehensible. We, among many others, encourage the general use of structured proofs.

Alas, most proofs are not written in this style, and we would do users a disservice by insisting on this style. For that reason, the `spfblock` environment turns all subproofs and proof step macros into presentationally neutral *inline* annotations, as in the induction step of the following example:

```
1 \begin{sproof}[id=simple-proof,method=induction]
2   {We prove that  $\sum_{i=1}^n 2i-1=n^2$  by induction over  $n$ }
```

```

3 For the induction we have to consider three cases: % <- a comment
4 \begin{subproof}{\$n=1\$}
5   \spfstep*{then we compute  $1=1^2$ }
6 \end{subproof}
7 \begin{subproof}{\$n=2\$}
8   This case is not really necessary, but we do it for the
9   fun of it (and to get more intuition).
10  \spfstep*{We compute  $1+3=2^2=4$ .}
11 \end{subproof}
12 \begin{subproof}{\$n>1\$}\begin{spfblock}
13   \assumption[id=ind-hyp]{
14     Now, we assume that the assertion is true for a certain  $k \geq 1$ ,
15     i.e.  $\mathsf{yield}\{\sum_{i=1}^k (2i-1) = k^2\}$ .
16   }
17
18   We have to show that we can derive the assertion for  $n=k+1$  from
19   this assumption, i.e.  $\sum_{i=1}^{k+1} (2i-1) = (k+1)^2$ .
20
21   \spfstep{
22     We obtain  $\mathsf{yield}\{\sum_{i=1}^{k+1} (2i-1) =$ 
23      $\sum_{i=1}^k (2i-1) + 2(k+1) - 1\}$ 
24     \spfjust{by \splitsum{\comp{splitting the sum}}
25     \arg*{\mathsf{yield}\{\sum_{i=1}^{k+1} (2i-1) = (k+1)^2\}}}.
26   }
27   \spfstep{
28     Thus we have  $\mathsf{yield}\{\sum_{i=1}^{k+1} (2i-1) = k^2 + 2k + 1\}$ 
29     \spfjust{by \symname{induction-hypothesis}}.
30   }
31   \conclude{
32     We can \spfjust{\simplification{\comp{simplify} the right-hand side
33     \arg*{ $k^2 + 2k + 1$ }} to
34      $(k+1)^2$ , which proves the assertion.
35   }
36 \end{subproof}\end{spfblock}
37 \conclude{
38   We have considered all the cases, so we have proven the assertion.
39 }
40 \end{spproof}

```

This yields the following result:

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

*For the induction we have to consider three cases:*

1.  $n = 1$

then we compute  $1 = 1^2$

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  $1 + 3 = 2^2 = 4$ .

3.  $n > 1$

Now, we assume that the assertion is true for a certain  $k \geq 1$ , i.e.  $\sum_{i=1}^k (2i - 1) = k^2$ .

We have to show that we can derive the assertion for  $n = k + 1$  from this assumption,

i.e.  $\sum_{i=1}^{k+1} (2i - 1) = (k + 1)^2$ .  
 We obtain  $\sum_{i=1}^{k+1} 2i - 1 = \sum_{i=1}^k 2i - 1 + 2(k + 1) - 1$  by [splitting the sum](#). Thus  
 we have  $\sum_{i=1}^{k+1} (2i - 1) = k^2 + 2k + 1$  by [induction hypothesis](#). We can [simplify](#) the  
 right-hand side to  $k + 1^2$ , which proves the assertion.  
 $\Rightarrow$  We have considered all the cases, so we have proven the assertion. □

**sproof** (*env.*) The **sproof** 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 **spfstep**, **spfcomment**, and **spfcases** environments that are used to markup the proof steps.

---

**\spfilea** The **\spfilea** 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 same optional argument as **sproof** and another one: a natural language text that sketches the proof.

---

**\spfstep** Regular proof steps are marked up with the **\spfstep** macro, 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.

---

**\yield** See above

---

**\spfjust** This evidence is marked up with the **\spfjust** macro in the **stex-proofs** package. This environment totally invisible to the formatted result; it wraps the text in the proof step that corresponds to the evidence (ideally, a semantically marked-up term).

---

**\assumption** The **\assumption** macro allows to mark up a (justified) assumption.

---

**\justarg**

**subproof** (*env.*) The **subproof** environment is used to mark up a subproof. This environment takes an optional **KeyVal** argument for semantic annotations and a second argument that allows to specify an introductory comment (just like in the **proof** environment). The **method** key can be used to give the name of the proof method executed to make this subproof.

---

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

The `stex-proofs` package provides the `\sproofend` macro for this.

---

`\sProofEndSymbol` If a different symbol for the proof end is to be used (e.g. *q.e.d*), then this can be obtained by specifying it using the `\sProofEndSymbol` configuration macro (e.g. by specifying `\sProofEndSymbol{q.e.d}`).

Some of the proof structuring macros above will insert proof end symbols for sub-proofs, in most cases, this is desirable to make the proof structure explicit, but sometimes this wastes space (especially, if a proof ends in a case analysis which will supply its own proof end marker). To suppress it locally, just set `proofend={}` in them or use `\sProofEndSymbol{}`.

### 7.3 Highlighting and Presentation Customizations

The environments starting with `s` (i.e. `smodule`, `sassertion`, `sexample`, `sdefinition`, `sparagraph` and `sproof`) by default produce no additional output whatsoever (except for the environment content of course). Instead, the document that uses them (whether directly or e.g. via `\inputref`) can decide how these environments are supposed to look like.

The `stexthm` package defines some default customizations that can be used, but of course many existing  $\text{\LaTeX}$  templates come with their own `definition`, `theorem` and similar environments that authors are supposed (or even required) to use. Their concrete syntax however is usually not compatible with all the additional arguments that  $\text{\LaTeX}$  allows for semantic information.

Therefore we introduced the separate environments `sdefinition` etc. instead of using `definition` directly. We allow authors to specify how these environments should be styled via the commands `stexpatch*`.

---

`\stexpatchmodule`  
`\stexpatchdefinition`  
`\stexpatchassertion`  
`\stexpatchexample`  
`\stexpatchparagraph`  
`\stexpatchproof`

All of these commands take one optional and two proper arguments, i.e. `\stexpatch* [<type>] {<begin-code>} {<end-code>}`.

After  $\text{\LaTeX}$  reads and processes the optional arguments for these environments, (some of) their values are stored in the macros `\s*<field>` (i.e. `sexampleid`, `\sassertionname`, etc.). It then checks for all the values `<type>` in the `type=`-list, whether an `\stexpatch* [<type>]` for the current environment has been called. If it finds one, it uses the patches `<begin-code>` and `<end-code>` to mark up the current environment. If no patch for (any of) the type(s) is found, it checks whether and `\stexpatch*` was called without optional argument.

For example, if we want to use a predefined `theorem` environment for `sassertions` with `type=theorem`, we can do

```
1 \stexpatchassertion[theorem]{\begin{theorem}}{\end{theorem}}
```

...or, rather, since e.g. `theorem`-like environments defined using `amsthm` take an optional title as argument, we can do:

```

1 \stexpatchassertion[theorem]
2   {\ifx\sassertiontitle\@empty
3     \begin{theorem}
4   \else
5     \begin{theorem}[\sassertiontitle]
6   \fi}
7 {\end{theorem}}

```

Or, if we want *all kinds of sdefinitions* to use a predefined definition-environment irrespective of their type=, then we can issue the following customization patch:

```

1 \stexpatchdefinition
2   {\ifx\sdefinitiontitle\@empty
3     \begin{definition}
4   \else
5     \begin{definition}[\sdefinitiontitle]
6   \fi}
7 {\end{definition}}

```

<hr/>	
<code>\compemph</code>	Apart from the environments, we can control how $\text{\TeX}$ highlights variables, notation
<code>\varemp</code>	components, <code>\symrefs</code> and <code>\definiendums</code> , respectively.
<code>\symrefemph</code>	To do so, we simply redefine these four macros. For example, to highlight nota-
<code>\defemph</code>	tion components (i.e. everything in a <code>\comp</code> ) in blue, as in this document, we can do
	<code>\def\compemph#1{\textcolor{blue}{#1}}</code> . By default, <code>\compemph</code> et al do nothing.

<hr/>	
<code>\compemph@uri</code>	For each of the four macros, there exists an additional macro that takes the full URI of
<code>\varemp@uri</code>	the relevant symbol currently being highlighted as a second argument. That allows us to
<code>\symrefemph@uri</code>	e.g. use pdf tooltips and links. For example, this document uses <sup>5</sup>
<code>\defemph@uri</code>	
	<pre> 1 \protected\def\symrefemph@uri#1#2{ 2   \pdftooltip{ 3     \symrefemph{#1} 4   }{ 5     URI:~\detokenize{#2} 6   } 7 } </pre>

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

## Chapter 8

# Cross References

If we take features like `\inputref` and `\mhinput` (and the `sfragment`-environment, see subsection `\TeX _annotate:nnn counterchapter ,arabic,99.\TeX _annotate:nnn countersection ,arabic,22.\TeX _annotate:nnn countersubsection ,arabic,11`) seriously, and build large documents modularly from individually compiling documents for sections, chapters and so on, cross-referencing becomes an interesting problem.

Say, we have a document `main.tex`, which `\inputrefs` a section `section1.tex`, which references a definition with label `some_definition` in `section2.tex` (subsequently also inputted in `main.tex`). Then the numbering of the definition will depend on the *document context* in which the document fragment `section2.tex` occurs - in `section2.tex` itself (as a standalone document), it might be *Definition 1*, in `main.tex` it might be *Definition 3.1*, and in `section1.tex`, the definition *does not even occur*, so it needs to be referenced by some other text.

What we would want in that instance is an equivalent of `\autoref`, that takes the document context into account to yield something like *Definition 1*, *Definition 3.1* or *Definition 1 in the section on Foo* respectively.

The `\sref` command attempts to do precisely that. Unlike plain `\ref`, `\autoref` etc., `\sref` refers to not just a *label*, but instead a pair consisting of a *label* and the *document* in whose context we want to refer to it. Conversely, every *document* (i.e. standalone compilable `.tex`-file) keeps track of the “names” (*Definition 3.1* etc.) for every label as determined in the context of the document, and stores them in a dedicated file `\jobname.sref`. Additionally, every document has a “reference name” (e.g. “*the section on Foo*”). This allows us to refer to “label *x* in document *D*” to yield “*Definition 1 in the section on Foo*”. And of course, `\TeX` can decide based on the current document to either refer to the label by its “full name” or directly as e.g. *Definition 3.1* depending on whether the label occurs in the current document anyway (and link to it accordingly).

For that to work, we need to supply (up to) three pieces of information:

- The *label* of the reference target (e.g. `some_definition`),
- (optionally) the *file*/document containing the reference target (e.g. `section2`). This is not strictly necessary, but allows for additional disambiguation between possibly duplicate labels across files, and
- (optionally) the document context, in which we want to refer to the reference target (e.g. `main`).



Additionally, the document in which we want to reference a label needs a title for external references.

---

**\sref** `\sref[archive=<archive1>,file=<file>]  
{<label>}[archive=<archive2>,in=<document-context>,title=<title>]`

---

This command references *<label>* (declared in *<file>* in *<archive1>*). If the object (section, figure, etc.) with that label occurs ultimately in the same document, `\sref` will ignore the second set of optional arguments and simply defer to `\autoref` if that command exists, or `\ref` if the `hyperref` package is not included.

If the referenced object does *not* occur in the current document however, `\sref` will refer to it by the object's name as it occurs in the file *<document-context>* in *<archive2>*.

For example, the reference to the `sfragment`-environment above will appear as “subsection 7.2.1 (Introduction) in the `gTeX3` manual” if you are reading this in the package documentation for `stex-references` directly, but as a linked “subsection 7.2.1” in the full documentation or manual. This is achieved using

```
\sref[file=stex-document-structure]{sec:ds:intro}[in=./stex-manual,title={the \sTeX}]
```

For a further example, the following:

### Part III

will say “Part III” (and link accordingly) in the full documentation, and “Part III (Extensions) in the full `gTeX3` documentation” everywhere else. This is achieved using

```
\sref[file=./stex-doc]{part:extends}[in=./stex-doc,title={the full \sTeX}3 document]
```

---

**\extref** `\sref[archive=<archive1>,file=<file>]  
{<label>}[archive=<archive2>,in=<document-context>,title=<title>]}`

---

The `\extref`-command behaves exactly like `\sref`, but takes *required* the document context argument and will always use it for generating the document text, regardless of whether the label occurs in the current document.

## Chapter 9

# Additional Packages

### 9.1 Tikzinput: Treating TIKZ code as images

---

**image** The behavior of the `ikzinput` package is determined by whether the `image` option is given. If it is not, then the `tikz` package is loaded, all other options are passed on to it and `\tikzinput{<file>}` inputs the TIKZ file `<file>.tex`; if not, only the `graphicx` package is loaded and `\tikzinput{<file>}` loads an image file `<file>.<ext>` generated from `<file>.tex`.

The selective input functionality of the `tikzinput` package assumes that the TIKZ pictures are externalized into a standalone picture file, such as the following one

```
1 \documentclass{standalone}
2 \usepackage{tikz}
3 \usetikzpackage{...}
4 \begin{document}
5   \begin{tikzpicture}
6     ...
7   \end{tikzpicture}
8 \end{document}
```

The `standalone` class is a minimal  $\text{\LaTeX}$  class that when loaded in a document that uses the `standalone` package: the preamble and the `document` environment are disregarded during loading, so they do not pose any problems. In effect, an `\input` of the file above only sees the `tikzpicture` environment, but the file itself is standalone in the sense that we can run  $\text{\LaTeX}$  over it separately, e.g. for generating an image file from it.

---

**\tikzinput** This is exactly where the `tikzinput` package comes in: it supplies the `\tikzinput` macro, which – depending on the `image` option – either directly inputs the TIKZ picture (source) or tries to load an image file generated from it.

Concretely, if the `image` option is not set for the `tikzinput` package, then `\tikzinput[<opt>]{<file>}` disregards the optional argument `<opt>` and inputs `<file>.tex` via `\input` and resizes it to as specified in the `width` and `height` keys. If it is, `\tikzinput[<opt>]{<file>}` expands to `\includegraphics[<opt>]{<file>}`.

`\ctikzinput` is a version of `\tikzinput` that is centered.

---

<code>\mhtikzinput</code>	<code>\mhtizkinput</code> is a variant of <code>\tikzinput</code> that treats its file path argument as a relative path in a math archive in analogy to <code>\inputref</code> . To give the archive path, we use the <code>mhrepos=</code> key. Again, <code>\cmhtizkinput</code> is a version of <code>\mhtikzinput</code> that is centered.
<code>\cmhtikzinput</code>	

---



---

<code>\libusetikzlibrary</code>	Sometimes, we want to supply archive-specific TIKZ libraries in the <code>lib</code> folder of the archive or the <code>meta-inf/lib</code> of the archive group. Then we need an analogon to <code>\libinput</code> for <code>\usetikzlibrary</code> . The <code>stex-tikzinput</code> package provides the <code>libusetikzlibrary</code> for this purpose.
---------------------------------	---

---

## 9.2 Modular Document Structuring

### 9.2.1 Introduction

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

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

DAG models of documents allow to replace the “Copy and Paste” in the source document with a label-and-reference model where document are shared in the document source and the formatter does the copying during document formatting/presentation.

### 9.2.2 Package Options

The `document-structure` package accepts 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>

### 9.2.3 Document Fragments

`sfragment` (*env.*) The structure of the document is given by nested `sfragment` environments. In the  $\text{\LaTeX}$  route, the `sfragment` environment is flexibly mapped to sectioning commands, inducing the proper sectioning level from the nesting of `sfragment` environments. Correspondingly, the `sfragment` environment takes an optional key/value argument for metadata followed by a regular argument for the (section) title of the sfragment. The optional metadata argument has the keys `id` for an identifier, `creators` and `contributors` for the Dublin Core metadata [DCM03]. The option `short` allows to give a short title for the generated section. If the title contains semantic macros, we need to give the `loadmodules` key (it needs no value). For instance we would have

```

1 \begin{smodule}{foo}
2   \symdef{bar}{Ba_r}
3   ...
4   \begin{sfragment}[id=sec.bardriv,loadmodules]
5     {Introducing $\protect\bar$ Derivations}

```

TeX automatically computes the sectioning level, from the nesting of `sfragment` environments.

But sometimes, we want to skip levels (e.g. to use a `\subsection*` as an introduction for a chapter).

`blindfragment` (*env.*) Therefore the `document-structure` package provides a variant `blindfragment` that does not produce markup, but increments the sectioning level and logically groups document parts that belong together, but where traditional document markup relies on convention rather than explicit markup. The `blindfragment` environment is useful e.g. for creating frontmatter at the correct level. The example below shows a typical setup for the outer document structure of a book with parts and chapters.

```

1 \begin{document}
2 \begin{blindfragment}
3 \begin{blindfragment}
4 \begin{frontmatter}
5 \maketitle\newpage
6 \begin{sfragment}{Preface}
7 ... <<preface>> ...
8 \end{sfragment}
9 \clearpage\setcounter{tocdepth}{4}\tableofcontents\clearpage
10 \end{frontmatter}
11 \end{blindfragment}
12 ... <<introductory remarks>> ...
13 \end{blindfragment}
14 \begin{sfragment}{Introduction}
15 ... <<intro>> ...
16 \end{sfragment}
17 ... <<more chapters>> ...
18 \bibliographystyle{alpha}\bibliography{kwarc}
19 \end{document}

```

Here we use two levels of `blindfragment`:

- The outer one groups the introductory parts of the book (which we assume to have a sectioning hierarchy topping at the part level). This `blindfragment` makes sure that the introductory remarks become a “chapter” instead of a “part”.
- The inner one groups the frontmatter<sup>3</sup> and makes the preface of the book a section-level construct. The `frontmatter` environment also suppresses numbering as is traditional for prefaces.

---

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

---

<sup>3</sup>We shied away from redefining the `frontmatter` to induce a `blindfragment`, but this may be the “right” way to go in the future.

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

## 9.2.4 Ending Documents Prematurely

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

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

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

## 9.2.5 Global Document Variables

To make document fragments more reusable, we sometimes want to make the content depend on the context. We use **document variables** for that.

<hr/> <code>\setSGvar</code> <hr/>	<code>\setSGvar{&lt;vname&gt;}{&lt;text&gt;}</code> to set the global variable <code>&lt;vname&gt;</code> to <code>&lt;text&gt;</code> and <code>\useSGvar{&lt;vname&gt;}</code> to reference it.
<code>\useSGvar</code>	

<hr/> <code>\ifSGvar</code> <hr/>	With <code>\ifSGvar</code> we can test for the contents of a global variable: the macro call <code>\ifSGvar{&lt;vname&gt;}{&lt;val&gt;}{&lt;ctext&gt;}</code> tests the content of the global variable <code>&lt;vname&gt;</code> , only if (after expansion) it is equal to <code>&lt;val&gt;</code> , the conditional text <code>&lt;ctext&gt;</code> is formatted.
-----------------------------------	---

## 9.3 Slides and Course Notes

### 9.3.1 Introduction

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

The `notesslides` class takes the notion of a slide frame from Till Tantau’s excellent `beamer` class and adapts its notion of frames for use in the  $\TeX$  and OMDoc. To

support semantic course notes, it extends the notion of mixing frames and explanatory text, but rather than treating the frames as images (or integrating their contents into the flowing text), the `notesslides` package displays the slides as such in the course notes to give students a visual anchor into the slide presentation in the course (and to distinguish the different writing styles in slides and course notes).

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

### 9.3.2 Package Options

The `notesslides` class takes a variety of class options:

<u>slides</u> <u>notes</u>	The options <code>slides</code> and <code>notes</code> switch between slides mode and notes mode (see <a href="#">subsection <code>\TeX _annotate:nnn counterchapter ,arabic,99,\TeX _annotate:nnn countersection ,arabic,33,\TeX _annotate:nnn countersubsection ,arabic,33</code></a> ).
<u>sectocframes</u>	If the option <code>sectocframes</code> is given, then for the <code>sfragments</code> , special frames with the <code>sfragment</code> title (and number) are generated.
<u>frameimages</u> <u>fiboxed</u>	If the option <code>frameimages</code> is set, then slide mode also shows the <code>\frameimage</code> -generated frames (see ??). If also the <code>fiboxed</code> option is given, the slides are surrounded by a box.

### 9.3.3 Notes and Slides

- frame** (*env.*) Slides are represented with the `frame` environment just like in the `beamer` class, see [\[Tanb\]](#) for details.
- note** (*env.*) The `notesslides` class adds the `note` environment for encapsulating the course note fragments.



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  $\LaTeX$  becomes confused and throws error messages that are difficult to decipher.

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

```

1 \ifnotes\maketitle\else
2 \frame[noframenumbering]\maketitle\fi
3
4 \begin{note}
5   We start this course with ...
6 \end{note}
7
8 \begin{frame}
9   \frametitle{The first slide}

```

```

10 ...
11 \end{frame}
12 \begin{note}
13 ... and more explanatory text
14 \end{note}
15
16 \begin{frame}
17 \frametitle{The second slide}
18 ...
19 \end{frame}
20 ...

```

---

**\ifnotes** Note the use of the `\ifnotes` conditional, which allows different treatment between `notes` and `slides` mode – manually setting `\notesttrue` or `\notestfalse` 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).

---

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

`nparagraph` (*env.*) There are some environments that tend to occur at the top-level of `note` environments.

`nparagraph` (*env.*) We make convenience versions of these: e.g. the `nparagraph` environment is just an

`ndefinition` (*env.*) `sparagraph` inside a `note` environment (but looks nicer in the source, since it avoids one

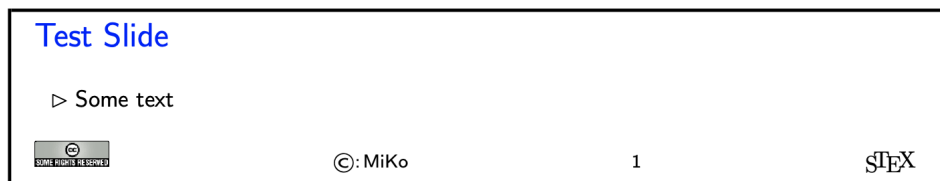
`nexample` (*env.*) level of source indenting). Similarly, we have the `nfragment`, `ndefinition`, `nexample`,

`nsproof` (*env.*) `nsproof`, and `nassertion` environments.

`nassertion` (*env.*)

### 9.3.4 Customizing Header and Footer Lines

The `notesslides` package and class comes with a simple default theme named `sTeX` that provided by the `beamerthemesTeX`. It is assumed as the default theme for `sTeX`-based notes and slides. The result in `notes` mode (which is like the `slides` version except that the slide height is variable) is



The footer line can be customized. In particular the logos.

---

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

---



---

`\setsource` The default footer line of the `notesslides` package mentions copyright and licensing. In `notesslides \source` stores the author's name as the copyright holder. By default it is the author's name as defined in the `\author` macro in the preamble. `\setsource{<name>}` can change the writer's name.

---



---

`\setlicensing` For licensing, we use the Creative Commons Attribution-ShareAlike license by default to strengthen the public domain. If package `hyperref` is loaded, then we can attach a hyperlink to the license logo. `\setlicensing[<url>]{<logo name>}` is used for customization, where `<url>` is optional.

---

### 9.3.5 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  $\text{\LaTeX}$  notes.

---

`\frameimage` In this case we can use `\frameimage[<opt>]{<path>}`, where `<opt>` are the options of `\includegraphics` from the `graphicx` package [CR99] and `<path>` is the file path (extension can be left off like in `\includegraphics`). We have added the `label` key that allows to give a frame label that can be referenced like a regular `beamer` frame.

---

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

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

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


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

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

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

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

---

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

---



### 9.3.6 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:

```
1 \excursion{founif}{../fragments/founif.en}
2 {We will cover first-order unification in}
3 ...
4 \begin{appendix}\printexcursions\end{appendix}
```

It generates a paragraph that references the excursion whose source is in the file `../fragments/founif.en.tex` and automatically books the file for the `\printexcursions` command that is used here to put it into the appendix. We will look at the mechanics now.

---

`\excursion` The `\excursion{<ref>}{<path>}{<text>}` is syntactic sugar for

```
1 \begin{nparagraph}[title=Excursion]
2   \activateexcursion{founif}{../ex/founif}
3   We will cover first-order unification in \sref{founif}.
4 \end{nparagraph}
```

---

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

Sometimes, we want to reference – in an excursion – part of another. We can use `\excursionref{<label>}` for that.

---

`\excursiongroup` Finally, we usually want to put the excursions into an `sfragment` environment and add an introduction, therefore we provide the a variant of the `\printexcursions` macro: `\excursiongroup[id=<id>,intro=<path>]` is equivalent to

```
1 \begin{note}
2 \begin{sfragment}[id=<id>]{Excursions}
3   \inputref{<path>}
4   \printexcursions
5 \end{sfragment}
6 \end{note}
```



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

## 9.4 Representing Problems and Solutions

### 9.4.1 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>4</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.

### 9.4.2 Problems and Solutions

---

<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.

`problem (env.)` The main environment provided by the `problempackage` is (surprise surprise) the `problem` environment. It is used to mark up problems and exercises. The environment takes an optional `KeyVal` argument with the keys `id` as an identifier that can be reference later, `pts` for the points to be gained from this exercise in homework or quiz situations, `min` for the estimated minutes needed to solve the problem, and finally `title` for an informative title of the problem.

#### Example 40

Input:

---

<sup>4</sup>for the moment multiple choice problems are not supported, but may well be in a future version

```

1 \documentclass{article}
2 \usepackage[solutions,hints,pts,min]{problem}
3 \begin{document}
4   \begin{sproblem}[id=elephants,pts=10,min=2,title=Fitting Elephants]
5     How many Elephants can you fit into a Volkswagen beetle?
6     \begin{hint}
7       Think positively, this is simple!
8     \end{hint}
9     \begin{exnote}
10      Justify your answer
11    \end{exnote}
12  \begin{solution}[for=elephants]
13    Four, two in the front seats, and two in the back.
14    \begin{gnote}
15      if they do not give the justification deduct 5 pts
16    \end{gnote}
17  \end{solution}
18 \end{sproblem}
19 \end{document}

```

Output:

**Problem 9.4.1 (Fitting Elephants)**  
 How many Elephants can you fit into a Volkswagen beetle?  


---

**Hint:** Think positively, this is simple!  


---

**Note:** Justify your answer  


---



---

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


---

**Grading:** if they do not give the justification deduct 5 pts  


---



---

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

`hint (env.)` The `hint` and `exnote` environments can be used in a `problem` environment to give hints  
`exnote (env.)` and to make notes that elaborate certain aspects of the problem. The `gnote` (grading  
`gnote (env.)` notes) environment can be used to document situations that may arise in grading.

---

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

---

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

### 9.4.3 Markup for Added-Value Services

The `problem` package is all about specifying the meaning of the various moving parts of practice/exam problems. The motivation for the additional markup is that we can base added-value services from these, for instance auto-grading and immediate feedback.

The simplest example of this are multiple-choice problems, where the `problem` package allows to annotate answer options with the intended values and possibly feedback that can be delivered to the users in an interactive setting. In this section we will give some infrastructure for these, we expect that this will grow over time.

#### Multiple Choice Blocks

`mcb` (*env.*) Multiple choice blocks can be formatted using the `mcb` environment, in which single choices are marked up with `\mcc` macro.

---

`\mcc` `\mcc[⟨keyvals⟩]{⟨text⟩}` takes an optional key/value argument `⟨keyvals⟩` for choice meta-data and a required argument `⟨text⟩` for the proposed answer text. The following keys are supported

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

What we see when this is formatted to PDF depends on the context. In solutions mode (we start the solutions in the code fragment below) we get

#### Example 41

Input:

```
1 \startsolutions
2 \begin{sproblem}[title=Functions,name=functions1]
3   What is the keyword to introduce a function definition in python?
4   \begin{mcb}
5     \mcc[T]{def}
6     \mcc[F,feedback=that is for C and C++){function}
7     \mcc[F,feedback=that is for Standard ML]{fun}
8     \mcc[F,Ftext=Noooooooooo,feedback=that is for Java]{public static void}
9   \end{mcb}
10 \end{sproblem}
```

Output:

#### Problem 9.4.2 (Functions)

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

☐ def

**Correct!**

☐ function

**Wrong!** *that is for C and C++*

☐ fun

**Wrong!** *that is for Standard ML*

☐ public static void

**Wrong!** *that is for Java*

In “exam mode” where disable solutions (here via \stopsolutions)

#### Example 42

Input:

```
1 \stopsolutions
2 \begin{sproblem}[title=Functions,name=functions1]
3   What is the keyword to introduce a function definition in python?
4   \begin{mcb}
5     \mcc[T]{def}
6     \mcc[F,feedback=that is for C and C++){function}
7     \mcc[F,feedback=that is for Standard ML]{fun}
8     \mcc[F,Ftext=Nooooooooo,feedback=that is for Java]{public static void}
9   \end{mcb}
10 \end{sproblem}
```

Output:

#### Problem 9.4.3 (Functions)

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

☐ def

☐ function

☐ fun

☐ public static void

we get the questions without solutions (that is what the students see during the exam/quiz).

#### Filling-In Concrete Solutions

The next simplest situation, where we can implement auto-grading is the case where we have fill-in-the-blanks

---

`\fillinsol` The `\fillinsol` macro takes<sup>6</sup> an a single argument, which contains a concrete solution (i.e. a number, a string, ...), which generates a fill-in-box in test mode:

#### Example 43

Input:

```
1 \stopsolutions
2 \begin{problem}[id=elephants.fillin,title=Fitting Elephants]
3   How many Elephants can you fit into a Volkswagen beetle? \fillinsol{4}
4 \end{problem}
```

Output:

#### Problem 9.4.4 (Fitting Elephants)

How many Elephants can you fit into a Volkswagen beetle?

and the actual solution in solutions mode:

#### Example 44

Input:

```
1 \begin{problem}[id=elephants.fillin,title=Fitting Elephants]
2   How many Elephants can you fit into a Volkswagen beetle? \fillinsol{4}
3 \end{problem}
```

Output:

#### Problem 9.4.5 (Fitting Elephants)

How many Elephants can you fit into a Volkswagen beetle? !

If we do not want to leak information about the solution by the size of the blank we can also give `\fillinsol` an optional argument with a size: `\fillinsol[3cm]{12}` makes a box three cm wide.

Obviously, the required argument of `\fillinsol` can be used for auto-grading. For concrete data like numbers, this is immediate, for more complex data like strings “soft comparisons” might be in order.<sup>7</sup>

## 9.4.4 Including Problems

---

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

The sum of the points and estimated minutes (that we specified in the `pts` and `min` keys to the `problem` environment or the `\includeproblem` macro) to the log file and the

---

<sup>7</sup>EDNOTE: For the moment we only assume a single concrete value as correct. In the future we will almost certainly want to extend the functionality to multiple answer classes that allow different feedback like in MCQ. This still needs a bit of design. Also we want to make the formatting of the answer in solutions/test mode configurable.

screen after each run. This is useful in preparing exams, where we want to make sure that the students can indeed solve the problems in an allotted time period.

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

### 9.4.5 Testing and Spacing

The `problem` package is often used by the `hwexam` package, which is used to create homework assignments and exams. Both of these have a “test mode” (invoked by the package option `test`), where certain information –master solutions or feedback – is not shown in the presentation.

`\testspace` `\testspace` takes an argument that expands to a dimension, and leaves vertical space accordingly. Specific instances exist: `\testsmallspace`, `\testsmallspace`, `\testsmallspace` `\testsmallspace` give small (1cm), medium (2cm), and big (3cm) vertical space.  
`\testsmallspace` `\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.  
`\testnewpage`  
`\testemptypage`

## 9.5 Homeworks, Quizzes and Exams

### 9.5.1 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. It is designed to be compatible with `problems.sty`, and inherits some of the functionality.

### 9.5.2 Package Options

---

`solutions` 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).  
`notes`  
`hints`  
`gnotes`  
`pts`  
`min`

---

`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  $\text{\LaTeX}$  source.

### 9.5.3 Assignments

**assignment** (*env.*) 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 **title** — the ordinal of the **assignment** environment), **title** (for the assignment title; this is **type** referenced in the title of the assignment sheet), **type** (for the assignment type; e.g. “quiz”, **given** or “homework”), **given** (for the date the assignment was given), and **due** (for the date the assignment is due).

### 9.5.4 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.

### 9.5.5 Typesetting Exams

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

```
reqpts 1 \title{320101 General Computer Science (Fall 2010)}
        2 \begin{testheading}[duration=one hour,min=60,reqpts=27]
        3   Good luck to all students!
        4 \end{testheading}
```

Will result in



Name:

Matriculation Number:

## 320101 General Computer Science (Fall 2010)

2022-09-27

**You have one hour (sharp) for the test;**

Write the solutions to the sheet.

The estimated time for solving this exam is 60 minutes, leaving you 0 minutes for revising your exam.

You can reach 40 points if you solve all problems. You will only need 27 points for a perfect score, i.e. 13 points are bonus points.

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

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

	To be used for grading, do not write here													
prob.	9.4.1	9.4.2	9.4.3	9.4.4	9.4.5	1.1	2.1	2.2	2.3	3.1	3.2	3.3	Sum	grade
total	10					4	4	6	6	4	4	2	40	
reached														

good luck

EdN:8

8

---

<sup>8</sup>EDNOTE: MK: The first three “problems” come from the stex examples above, how do we get rid of this?

**Part II**  
**Documentation**

# Chapter 10

## STEX-Basics

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

### 10.1 Macros and Environments

---

<code>\sTeX</code>	Both print this ST <sub>E</sub> X 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>*.

#### 10.1.1 HTML Annotations

---

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

---

---

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

---

---

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

---

---

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

---

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

---

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

---

Annotates the HTML generated by `⟨content⟩` with

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

### 10.1.2 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`.

### 10.1.3 Auxiliary Methods

---

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

---

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

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

---

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

---

# Chapter 11

## STEX-MathHub

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

### 11.1 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.
--------------------------------	---

---

#### 11.1.1 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>	The inverse; turns a path into a string and stores it in the second argument variable, or
<code>\stex_path_to_string:N</code>	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> *	
<code>\stex_path_if_absolute:N<math>\underline{T}</math></code> *	

---

Checks whether the path provided is *absolute*, i.e. starts with an empty segment

---

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

---

### 11.1.2 MathHub Archives

---

<code>\mathhub</code>	We determine the path to the local MathHub folder via one of four means, in order of
<code>\c_stex_mathhub_seq</code>	precedence:
<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, or
4. a path specified in `~/.stex/mathhub.path`.

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

---

<code>\l_stex_current_repository_prop</code>
--

---

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

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

---

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

---

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

---

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

---



---

<code>\stex_in_repository:nn</code>	<code>\stex_in_repository:nn{<i>repository-name</i>}{<i>code</i>}</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}`.

### 11.1.3 Using Content in Archives

<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/> <code>\inputref</code> <hr/> <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/> <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/> <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/> <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/> <code>\mhgraphics</code> <hr/> <code>\cmhgraphics</code>	<i>If</i> the <code>graphicx</code> package is loaded, these macros are defined at <code>\begin{document}</code> . <code>\mhgraphics</code> takes the same arguments as <code>\includegraphics</code> , with the additional optional key <code>mhrepos</code> . It then resolves the file path in <code>\mhgraphics[mhrepos=Foo/Bar]{foo/bar.png}</code> relative to the <code>source</code> -folder of the <code>Foo/Bar</code> -archive. <code>\cmhgraphics</code> additional wraps the image in a <code>center</code> -environment.
<hr/> <code>\lstinputmhlisting</code> <hr/> <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 12

# STEX-References

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

### 12.1 Macros and Environments

---

`\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`

#### 12.1.1 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>`.



### 12.1.2 Using References

---

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

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

---

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

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

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

---

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

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

# Chapter 13

## sTeX-Modules

This sub package contains code related to Modules

### 13.1 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:

<hr/> <hr/> <code>\c_stex_module_&lt;URI&gt;_prop</code>	A property list with the following fields: <ul style="list-style-type: none"><li><code>name</code> The <i>name</i> of the module,</li><li><code>ns</code> the <i>namespace</i> in field <code>ns</code>,</li><li><code>file</code> the <i>file</i> containing the module, as a sequence of path fragments</li><li><code>lang</code> the module's <i>language</i>,</li><li><code>sig</code> the language of the signature module, if the current file is a translation from some other language,</li><li><code>deprecate</code> if this module is deprecated, the module that replaces it,</li><li><code>meta</code> the metatheory of the module.</li></ul>
--	---

<hr/> <hr/> <code>\c_stex_module_&lt;URI&gt;_code</code>	The code to execute when this module is activated (i.e. imported), e.g. to set all the semantic macros, notations, etc.
--	---

<hr/> <hr/> <code>\c_stex_module_&lt;URI&gt;_constants</code>	The names of all constants declared in the module
---	---

<hr/> <hr/> <code>\c_stex_module_&lt;URI&gt;_constants</code>	The full URIs of all modules imported in this module
---	--

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

---

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

---

Computes the current namespace as follows:

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

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

### 13.1.1 The smodule environment

**module** (*env.*) `\begin{module}[\langle options \rangle]{\langle name \rangle}`

Opens a new module with name `\langle name \rangle`. Options are:

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

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

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

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

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

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

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

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

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

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

---

**\stex\_module\_setup:nn** `\stex_module_setup:nn{\langle params \rangle}{\langle name \rangle}`

---

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

---

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

---

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

---

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

---

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

---

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

---

---

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

---

# Chapter 14

## STEX-Module Inheritance

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

### 14.1 Macros and Environments

#### 14.1.1 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:` ★ Tests whether SMS mode is currently active.  
`\stex_if_smsmode:` *TF* ★

---

---

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

---

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

---

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

---

### 14.1.2 Imports and Inheritance

---

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

---

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

---

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

---

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

---

<code>\stex_import_module_uri:nn</code>	<code>\stex_import_module_uri:nn {&lt;archive-ID&gt;} {&lt;module-path&gt;}</code>
---	--

---

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

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

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

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

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

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

2. Otherwise:

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

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

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

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

---

<code>\l_stex_import_name_str</code>	stores the result in these four variables.
<code>\l_stex_import_archive_str</code>	
<code>\l_stex_import_path_str</code>	
<code>\l_stex_import_ns_str</code>	

---



---

<code>\stex_import_require_module:nnnn</code>	<code>{&lt;ns&gt;} {&lt;archive-ID&gt;} {&lt;path&gt;} {&lt;name&gt;}</code>
---	--

---

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

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



# Chapter 15

## $\text{\texttt{S}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$ -Symbols

Code related to symbol declarations and notations

### 15.1 Macros and Environments

---

$\text{\texttt{\textbackslash symdecl}}$	$\text{\texttt{\textbackslash symdecl}}\{\langle\textit{macroname}\rangle\}[\langle\textit{args}\rangle]$
--	---

---

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

- **name**: An (OMDOC) name. By default equal to  $\langle\textit{macroname}\rangle$ .
- **type**: An (ideally semantic) term, representing a *type*. Not used by  $\text{\texttt{S}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$ , but passed on to MMT for semantic services.
- **def**: An (ideally semantic) term, representing a *definiens*. Not used by  $\text{\texttt{S}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$ , but passed on to MMT for semantic services.
- **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.  $\text{\texttt{\textbackslash symdecl}}\{\texttt{plus}}\}[\texttt{args=ii}]$  allows for  $\text{\texttt{\textbackslash plus}}\{2\}\{2\}$ .
  - a** an *associative* argument; i.e. a sequence of arbitrarily many arguments provided as a comma-separated list, e.g.  $\text{\texttt{\textbackslash symdecl}}\{\texttt{plus}}\}[\texttt{args=a}]$  allows for  $\text{\texttt{\textbackslash plus}}\{2,2,2\}$ .
  - b** a *variable* argument. Is treated by  $\text{\texttt{S}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$  like an **i**-argument, but an application is turned into an  $\text{\texttt{OMBind}}$  in OMDOC, binding the provided variable in the subsequent arguments of the operator; e.g.  $\text{\texttt{\textbackslash symdecl}}\{\texttt{forall}}\}[\texttt{args=bi}]$  allows for  $\text{\texttt{\textbackslash forall}}\{x\}\text{\texttt{\textbackslash in}}\{\texttt{Nat}}\}\{x\geq 0\}$ .

<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>type</code> (token list),</li> <li>• <code>args</code> (string of is, as and bs),</li> <li>• <code>arity</code> (integer string),</li> <li>• <code>assocs</code> (integer string; number of associative arguments),</li> </ul>
<hr/> <hr/> <code>\stex_all_symbols:n</code>	<p>Iterates over all currently available symbols. Requires two <code>\seq_map_break:</code> to break fully.</p>
<hr/> <hr/> <code>\stex_get_symbol:n</code>	<p>Computes the full URI of a symbol from a macro argument, e.g. the macro name, the macro itself, the full URI...</p>
<hr/> <hr/> <code>\notation</code>	<p><code>\notation[&lt;args&gt;]{&lt;symbol&gt;}{&lt;notations<sup>+</sup>&gt;}</code></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 16

## STEX-Terms

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

### 16.1 Macros and Environments

---

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

---

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

---

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

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

---

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

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

---

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

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

<hr/> <code>\STEXInternalTermMathAssocArgiiii</code> <hr/>	<code>\stex_term_arg:nnn⟨int⟩⟨prec⟩⟨notation⟩⟨type⟩⟨body⟩</code>
	Annotates $\langle body \rangle$ as the $\langle int \rangle$ th (associative) <i>sequence</i> argument (as comma-separated list of terms) of the current OMA or OMBIND, with (downwards) argument precedence $\langle prec \rangle$ and associative notation $\langle notation \rangle$ .
<hr/> <code>\infpref</code> <code>\neginfpref</code> <hr/>	Maximal and minimal notation precedences.
<hr/> <code>\dobrackets</code> <hr/>	<code>\dobrackets {⟨body⟩}</code> Puts $\langle body \rangle$ in parentheses; scaled if in display mode unscaled otherwise. Uses the current $\TeX$ brackets (by default ( and )), which can be changed temporarily using <code>\withbrackets</code> .
<hr/> <code>\withbrackets</code> <hr/>	<code>\withbrackets ⟨left⟩ ⟨right⟩ {⟨body⟩}</code> Temporarily (i.e. within $\langle body \rangle$ ) sets the brackets used by $\TeX$ for automated bracketing (by default ( and )) to $\langle left \rangle$ and $\langle right \rangle$ . Note that $\langle left \rangle$ and $\langle right \rangle$ need to be allowed after <code>\left</code> and <code>\right</code> in display-mode.
<hr/> <code>\stex_term_custom:nn</code> <hr/>	<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/> <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> <hr/>	<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/> <code>\STEXinvisible</code> <hr/>	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/> <code>\ellipses</code> <hr/>	TODO

## Chapter 17

# TeX-Structural Features

Code related to structural features

### 17.1 Macros and Environments

#### 17.1.1 Structures

`mathstructure` (*env.*) TODO

## Chapter 18

# TeX-Statements

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

### 18.1 Macros and Environments

`symboldoc (env.) \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 19

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

## Chapter 20

# sTeX-Metatheory

### 20.1 Symbols



**Part III**  
**Extensions**

## Chapter 21

# Tikzinput: Treating TIKZ code as images

### 21.1 Macros and Environments

## Chapter 22

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

## Chapter 23

# NotesSlides – Slides and Course Notes

## Chapter 24

# `problem.sty`: An Infrastructure for formatting Problems

## Chapter 25

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

**Part IV**  
**Implementation**

## Chapter 26

# $\text{\TeX}$ -Basics Implementation

### 26.1 The $\text{\TeX}$ Document Class

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

```
1 <*cls>
2
3 %%%%%%%%% basics.dtx %%%%%%%%%
4
5 \RequirePackage{expl3,l3keys2e}
6 \ProvidesExplClass{stex}{2022/09/14}{3.2.0}{sTeX document class}
7
8 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
9 \ProcessOptions
10
11 \bool_set_true:N \c_stex_document_class_bool
12
13 \RequirePackage{stex}
14
15 \stex_html_backend:TF {
16   \LoadClass{article}
17 }{
18   \LoadClass[border=1px,varwidth,crop=false]{standalone}
19   \setlength\textwidth{15cm}
20 }
21 \RequirePackage{standalone}
22
23
24 \clist_if_empty:NT \c_stex_languages_clist {
25   \seq_get_right:Nn \g_stex_currentfile_seq \l_tmpa_str
26   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
27   \seq_pop_right:Nn \l_tmpa_seq \l_tmpa_str % .tex
28   \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
29     \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
30       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
```



```

31   }
32 }
33 \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
34 \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
35   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
36   \prop_if_in:NoT \c_stex_languages_prop \l_tmpa_str {
37     \stex_debug:nn{language} {Language~\l_tmpa_str~
38       inferred~from~file~name}
39   \exp_args:NNo \stex_set_language:Nn \l_tmpa_str \l_tmpa_str
40 }
41 }
42 }
43 </cls>

```

## 26.2 Preliminaries

```

44 <*package>
45
46 %%%%%%%%% basics.dtx %%%%%%%%%
47
48 \RequirePackage{expl3,l3keys2e,ltxcmds}
49 \ProvidesExplPackage{stex}{2022/09/14}{3.2.0}{sTeX package}
50
51 \bool_if_exist:NF \c_stex_document_class_bool {
52   \bool_set_false:N \c_stex_document_class_bool
53   \RequirePackage{standalone}
54 }
55
56 \message{^^J*~This~is~sTeX~version~3.2.0~*^^J}
57
58 %\RequirePackage{morewrites}
59 %\RequirePackage{amsmath}
60
61 Package options:
62 \keys_define:nn { stex } {
63   debug      .clist_set:N = \c_stex_debug_clist ,
64   lang       .clist_set:N = \c_stex_languages_clist ,
65   mathhub    .tl_set_x:N = \mathhub ,
66   usesms     .bool_set:N = \c_stex_persist_mode_bool ,
67   writesms   .bool_set:N = \c_stex_persist_write_mode_bool ,
68   image      .bool_set:N = \c_tikzinput_image_bool ,
69   unknown    .code:n      = {}
70 }
71 \ProcessKeysOptions { stex }

```

**\stex** The sTeX logo:

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

(End definition for `\stex` and `\sTeX`. These functions are documented on page 76.)

## 26.3 Messages and logging

```

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

```

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

```

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

```

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

Redirecting messages:

```

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

```

## 26.4 HTML Annotations

```

107 <@@=stex_annotate>

```

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

```

108 \tl_new:N \l_stex_html_arg_tl

```

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

`\_stex_html_checkempty:n`

```

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

```

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

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

`\stex_if_do_html:TF`

```

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

```

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

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

```

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

```

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

`\stex_annotate_html:nn`

`\stex_annotate_invisible:n`

`\stex_annotate_invisible:nnn`

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

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

```

132 \ifcsname if@rustex\endcsname\else
133   \expandafter\newif\csname if@rustex\endcsname
134   \@rustexfalse
135 \fi
136 \ifcsname if@latexml\endcsname\else
137   \expandafter\newif\csname if@latexml\endcsname
138   \@latexmlfalse
139 \fi
140 \tl_if_exist:NF\stex@backend{
141   \if@rustex
142     \def\stex@backend{rustex}
143   \else
144     \if@latexml
145       \def\stex@backend{latexml}
146     \else

```

```

147     \cs_if_exist:NTF\HCode{
148         \def\stex@backend{tex4ht}
149     }{
150         \def\stex@backend{pdflatex}
151     }
152     \fi
153     \fi
154 }
155 \input{stex-backend-\stex@backend.cfg}
156
157 \newif\ifstexhtml
158 \stex_html_backend:TF\stexhtmltrue\stexhtmlfalse
159

```

(End definition for `\stex_annotate:nnn`, `\stex_annotate_invisible:n`, and `\stex_annotate_invisible:nnn`. These functions are documented on page 77.)

## 26.5 Babel Languages

```

160 <@@=stex_language>

```

`\c_stex_languages_prop`  
`\c_stex_language_abbrevs_prop`

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

```

161 \exp_args:NNx \prop_const_from_keyval:Nn \c_stex_languages_prop { \tl_to_str:n {
162     en = english ,
163     de = ngerman ,
164     ar = arabic ,
165     bg = bulgarian ,
166     ru = russian ,
167     fi = finnish ,
168     ro = romanian ,
169     tr = turkish ,
170     fr = french
171 }}
172
173 \exp_args:NNx \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop { \tl_to_str:n {
174     english   = en ,
175     ngerman   = de ,
176     arabic    = ar ,
177     bulgarian = bg ,
178     russian   = ru ,
179     finnish   = fi ,
180     romanian  = ro ,
181     turkish   = tr ,
182     french    = fr
183 }}
184 % todo: chinese simplified (zhs)
185 %     chinese traditional (zht)

```

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

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

```

186 \cs_new_protected:Nn \stex_set_language:Nn {
187     \str_set:Nx \l_tmpa_str {#2}
188     \prop_get:NoNT \c_stex_languages_prop \l_tmpa_str #1 {

```

```

189 \ifx\@onlypreamble\@notprerr
190 \ltx@ifpackageloaded{babel}{
191 \exp_args:No \selectlanguage #1
192 }{}
193 \else
194 \exp_args:No \str_if_eq:nnTF #1 {turkish} {
195 \RequirePackage[#1,shorthands=:!]{babel}
196 }{
197 \RequirePackage[#1]{babel}
198 }
199 \fi
200 }
201 }
202
203 \clist_if_empty:NF \c_stex_languages_clist {
204 \bool_set_false:N \l_tmpa_bool
205 \clist_clear:N \l_tmpa_clist
206 \clist_map_inline:Nn \c_stex_languages_clist {
207 \str_set:Nx \l_tmpa_str {#1}
208 \str_if_eq:nnT {#1}{tr}{
209 \bool_set_true:N \l_tmpa_bool
210 }
211 \prop_get:NoNTF \c_stex_languages_prop \l_tmpa_str \l_tmpa_str {
212 \clist_put_right:No \l_tmpa_clist \l_tmpa_str
213 } {
214 \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
215 }
216 }
217 \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
218 \bool_if:NTF \l_tmpa_bool {
219 \RequirePackage[\clist_use:Nn \l_tmpa_clist,,shorthands=:!]{babel}
220 }{
221 \RequirePackage[\clist_use:Nn \l_tmpa_clist,]{babel}
222 }
223 }
224
225 \AtBeginDocument{
226 \stex_html_backend:T {
227 \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
228 \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
229 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
230 \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
231 \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
232 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
233 \stex_debug:nn{basics} {Language~\l_tmpa_str~
234 inferred~from~file~name}
235 \stex_annotate_invisible:nnn{language}{ \l_tmpa_str }{}
236 }
237 }
238 }
239

```

## 26.6 Persistence

```

240 <@@=stex_persist>
241 \bool_if:NTF \c_stex_persist_mode_bool {
242   \def \stex_persist:n #1 {}
243   \def \stex_persist:x #1 {}
244 }{
245   \bool_if:NTF \c_stex_persist_write_mode_bool {
246     \iow_new:N \c__stex_persist_iow
247     \iow_open:Nn \c__stex_persist_iow{\jobname.sms}
248     \AtEndDocument{
249       \iow_close:N \c__stex_persist_iow
250     }
251     \cs_new_protected:Nn \stex_persist:n {
252       \tl_set:Nn \l_tmpa_tl { #1 }
253       \regex_replace_all:nnN { \cP\# } { \cO\# } \l_tmpa_tl
254       \regex_replace_all:nnN { \ } { \~ } \l_tmpa_tl
255       \exp_args:NNo \iow_now:Nn \c__stex_persist_iow \l_tmpa_tl
256     }
257     \cs_generate_variant:Nn \stex_persist:n {x}
258   }{
259     \def \stex_persist:n #1 {}
260     \def \stex_persist:x #1 {}
261   }
262 }

```

## 26.7 Auxiliary Methods

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

```

263 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
264   \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
265   \def#1{
266     \msg_error:nnnn{stex}{error/deactivated-macro}{\detokenize{#1}}{#2}
267   }
268 }

```

(End definition for \stex\_deactivate\_macro:Nn. This function is documented on page 77.)

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

```

269 \cs_new_protected:Nn \stex_reactivate_macro:N {
270   \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname
271 }

```

(End definition for \stex\_reactivate\_macro:N. This function is documented on page 77.)

**\ignorespacesandpars**

```

272 \protected\def\ignorespacesandpars{
273   \begingroup\catcode13=10\relax
274   \@ifnextchar\par{
275     \endgroup\expandafter\ignorespacesandpars\@gobble
276   }{
277     \endgroup
278   }
279 }

```

```

280
281 \cs_new_protected:Nn \stex_copy_control_sequence:NNN {
282   \tl_set:Nx \_tmp_args_tl {\cs_argument_spec:N #2}
283   \exp_args:NNo \tl_remove_all:Nn \_tmp_args_tl \c_hash_str
284   \int_set:Nn \l_tmpa_int {\tl_count:N \_tmp_args_tl}
285
286   \tl_clear:N \_tmp_args_tl
287   \int_step_inline:nn \l_tmpa_int {
288     \tl_put_right:Nx \_tmp_args_tl {{\exp_not:n{####}\exp_not:n{##1}}}
289   }
290
291   \tl_set:Nn #3 {\cs_generate_from_arg_count:NNnn #1 \cs_set:Npn}
292   \tl_put_right:Nx #3 { {\int_use:N \l_tmpa_int}{
293     \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
294     \exp_after:wN\exp_after:wN\exp_after:wN\exp_after:wN {
295       \exp_after:wN #2 \_tmp_args_tl
296     }
297   }}
298 }
299 \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {cNN}
300 \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {NcN}
301 \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {ccN}
302
303 \cs_new_protected:Nn \stex_copy_control_sequence_ii:NNN {
304   \tl_set:Nx \_tmp_args_tl {\cs_argument_spec:N #2}
305   \exp_args:NNo \tl_remove_all:Nn \_tmp_args_tl \c_hash_str
306   \int_set:Nn \l_tmpa_int {\tl_count:N \_tmp_args_tl}
307
308   \tl_clear:N \_tmp_args_tl
309   \int_step_inline:nn \l_tmpa_int {
310     \tl_put_right:Nx \_tmp_args_tl {{\exp_not:n{#####}\exp_not:n{##1}}}
311   }
312
313   \edef \_tmp_args_tl {
314     \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
315     \exp_after:wN\exp_after:wN\exp_after:wN {
316       \exp_after:wN #2 \_tmp_args_tl
317     }
318   }
319
320   \exp_after:wN \def \exp_after:wN \_tmp_args_tl
321   \exp_after:wN ##\exp_after:wN 1 \exp_after:wN ##\exp_after:wN 2
322   \exp_after:wN { \_tmp_args_tl }
323
324   \edef \_tmp_args_tl {
325     \exp_after:wN \exp_not:n \exp_after:wN {
326       \_tmp_args_tl {####1}{####2}
327     }
328   }
329
330   \tl_set:Nn #3 {\cs_generate_from_arg_count:NNnn #1 \cs_set:Npn}
331   \tl_put_right:Nx #3 { {\int_use:N \l_tmpa_int}{
332     \exp_after:wN\exp_not:n\exp_after:wN{\_tmp_args_tl}
333   }}

```

```

334 }
335
336 \cs_generate_variant:Nn \stex_copy_control_sequence_ii:NNN {cNN}
337 \cs_generate_variant:Nn \stex_copy_control_sequence_ii:NNN {NcN}
338 \cs_generate_variant:Nn \stex_copy_control_sequence_ii:NNN {ccN}

```

(End definition for \ignorespacesandpars. This function is documented on page 77.)

\MMTrule

```

339 \NewDocumentCommand \MMTrule {m m}{
340   \seq_set_split:Nnn \l_tmpa_seq , {#2}
341   \int_zero:N \l_tmpa_int
342   \stex_annotate_invisible:nnn{mmtrule}{scala://#1}{
343     \seq_if_empty:NF \l_tmpa_seq {
344       $\seq_map_inline:Nn \l_tmpa_seq {
345         \int_incr:N \l_tmpa_int
346         \stex_annotate:nnn{arg}{i\int_use:N \l_tmpa_int}{##1}
347       }$
348     }
349   }
350 }
351
352 \NewDocumentCommand \MMTinclude {m}{
353   \stex_annotate_invisible:nnn{import}{#1}{-}
354 }
355
356 \tl_new:N \g_stex_document_title
357 \cs_new_protected:Npn \STEXTtitle #1 {
358   \tl_if_empty:NT \g_stex_document_title {
359     \tl_gset:Nn \g_stex_document_title { #1 }
360   }
361 }
362 \cs_new_protected:Nn \stex_document_title:n {
363   \tl_if_empty:NT \g_stex_document_title {
364     \tl_gset:Nn \g_stex_document_title { #1 }
365     \stex_annotate_invisible:n{\noindent
366       \stex_annotate:nnn{doctitle}{-}{ #1 }
367     \par}
368   }
369 }
370 \AtBeginDocument {
371   \let \STEXTtitle \stex_document_title:n
372   \tl_if_empty:NF \g_stex_document_title {
373     \stex_annotate_invisible:n{\noindent
374       \stex_annotate:nnn{doctitle}{-}{ \g_stex_document_title }
375     \par}
376   }
377   \let \stex_maketitle:\maketitle
378   \def \maketitle{
379     \tl_if_empty:NF \@title {
380       \exp_args:No \stex_document_title:n \@title
381     }
382     \stex_maketitle:
383   }

```



```

384 }
385
386 \cs_new_protected:Nn \stex_par: {
387   \mode_if_vertical:F{
388     \if@minipage\else\if@nobreak\else\par\fi\fi
389   }
390 }
391
392 \cs_new_protected:Nn \__stex_persist_patchcounter:n{
393   \cs_set_eq:cc{__stex_persist_tmp_#1}{@#1}
394   \cs_set:cpn {@#1} ##1 {
395     \stex_annotate:nnn{counter}{
396       \expandafter\expandafter\expandafter
397       \expandafter\expandafter\expandafter
398       \expandafter\@gobble
399       \expandafter\expandafter\expandafter\@gobble
400       \expandafter\@gobble\detokenize{##1},
401       #1,\number##1}{\use:c{__stex_persist_tmp_#1}{##1}}
402   }
403 }
404
405 \cs_new_protected:Nn \stex_patch_counters: {
406   \__stex_persist_patchcounter:n{arabic}
407   \__stex_persist_patchcounter:n{roman}
408   \__stex_persist_patchcounter:n{Roman}
409   \__stex_persist_patchcounter:n{alph}
410   \__stex_persist_patchcounter:n{Alph}
411   \__stex_persist_patchcounter:n{fnsymbol}
412   \let\__stex_persist_tmp_refstepcounter\refstepcounter
413   \cs_set:Npn\refstepcounter##1{
414     \__stex_persist_tmp_refstepcounter{##1}
415     \stex_annotate:nnn{stepcounter}{##1}{ }
416   }
417 }
418
419 \cs_new_protected:Nn \stex_unpatch_counters: {
420   \let\@arabic\__stex_persist_tmp_arabic
421   \let\@roman\__stex_persist_tmp_roman
422   \let\@Roman\__stex_persist_tmp_Roman
423   \let\@alph\__stex_persist_tmp_alph
424   \let\@Alph\__stex_persist_tmp_Alph
425   \let\@fnsymbol\__stex_persist_tmp_fnsymbol
426   \let\refstepcounter\__stex_persist_tmp_refstepcounter
427 }
428
429 %\AtBeginDocument{
430 %}
431
432 </package>

```

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

## Chapter 27

# STEX -MathHub Implementation

```
433 <*package>
434
435 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
436
437 <@@=stex_path>
438
439 Warnings and error messages
440 \msg_new:nnn{stex}{error/norepository}{
441   No~archive~#1~found~in~#2
442 }
443 \msg_new:nnn{stex}{error/notinarchive}{
444   Not~currently~in~an~archive,~but~\detokenize{#1}~
445   needs~one!
446 }
447 \msg_new:nnn{stex}{error/nofile}{
448   \detokenize{#1}~could~not~find~file~#2
449 }
450 \msg_new:nnn{stex}{error/twofiles}{
451   \detokenize{#1}~found~two~candidates~for~#2
452 }
```

### 27.1 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`

```
451 \cs_new_protected:Nn \stex_path_from_string:Nn {
452   \stex_debug:nn{files}{#2}
453   \str_set:Nx \l_tmpa_str { #2 }
454   \str_if_empty:NTF \l_tmpa_str {
455     \seq_clear:N #1
456   }{
457     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
458     \sys_if_platform_windows:T{
```

```

459     \seq_clear:N \l_tmpa_tl
460     \seq_map_inline:Nn #1 {
461         \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
462         \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
463     }
464     \seq_set_eq:NN #1 \l_tmpa_tl
465 }
466 \stex_path_canonicalize:N #1
467 }
468 \stex_debug:nn{files}{Yields: \stex_path_to_string:N#1}
469 }
470

```

(End definition for `\stex_path_from_string:Nn`. This function is documented on page 78.)

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

```

471 \cs_new_protected:Nn \stex_path_to_string:NN {
472     \exp_args:NNe \str_set:Nn #2 { \seq_use:Nn #1 / }
473 }
474
475 \cs_new:Nn \stex_path_to_string:N {
476     \seq_use:Nn #1 /
477 }

```

(End definition for `\stex_path_to_string:NN` and `\stex_path_to_string:N`. These functions are documented on page 78.)

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

```

478 \str_const:Nn \c__stex_path_dot_str {.}
479 \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.

```

480 \cs_new_protected:Nn \stex_path_canonicalize:N {
481     \stex_debug:nn{paths}{canonicalizing~\seq_use:Nn #1 /}
482     \bool_set_false:N \l__stex_path_in_path_bool
483     \seq_if_empty:NF #1 {
484         \seq_clear:N \l_tmpa_seq
485         \seq_get_left:NN #1 \l_tmpa_tl
486         \str_if_empty:NT \l_tmpa_tl {
487             \seq_put_right:Nn \l_tmpa_seq {}
488         }
489         \seq_map_inline:Nn #1 {
490             \str_set:Nn \l_tmpa_tl { ##1 }
491             \str_if_eq:NnF \l_tmpa_tl \c__stex_path_dot_str {
492                 \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
493                     \bool_set_true:N \l__stex_path_in_path_bool
494                     \seq_if_empty:NNTF \l_tmpa_seq {
495                         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
496                             \c__stex_path_up_str
497                         }
498                     }{
499                         \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl

```

```

500         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
501             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
502                 \c__stex_path_up_str
503             }
504         }{
505             \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
506         }
507     }
508 }{
509     \str_if_empty:NNTF \l_tmpa_tl {
510         \bool_if:NT \l__stex_path_in_path_bool {
511             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
512         }
513     } {
514         \bool_set_true:N \l__stex_path_in_path_bool
515         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
516     }
517 }
518 }
519 }
520 \seq_gset_eq:NN #1 \l_tmpa_seq
521 \stex_debug:nn{paths}{...returns~\seq_use:Nn #1 /}
522 }
523 }

```

(End definition for `\stex_path_canonicalize:N`. This function is documented on page 78.)

`\stex_path_if_absolute_p:N`  
`\stex_path_if_absolute:N`**TF**

```

524 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
525     \seq_if_empty:NNTF #1 {
526         \prg_return_false:
527     }{
528         \seq_get_left:NN #1 \l_tmpa_tl
529         \sys_if_platform_windows:TF{
530             \str_if_in:NnTF \l_tmpa_tl {:}{
531                 \prg_return_true:
532             }{
533                 \prg_return_false:
534             }
535         }{
536             \str_if_empty:NNTF \l_tmpa_tl {
537                 \prg_return_true:
538             }{
539                 \prg_return_false:
540             }
541         }
542     }
543 }

```

(End definition for `\stex_path_if_absolute:NNTF`. This function is documented on page 78.)

## 27.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

544 \str_new:N\l_stex_kpsewhich_return_str
545 \cs_new_protected:Nn \stex_kpsewhich:n {\begingroup
546   \catcode'\ =12
547   \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
548   \tl_gset_eq:NN \l_tmpa_tl \l_tmpa_tl
549   \endgroup
550   \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
551   \tl_trim_spaces:N \l_stex_kpsewhich_return_str
552 }

```

(End definition for `\stex_kpsewhich:n`. This function is documented on page 78.)

We determine the PWD

`\c_stex_pwd_seq`  
`\c_stex_pwd_str`

```

553 \sys_if_platform_windows:TF{
554   \begingroup\escapechar=-1\catcode'\ =12
555   \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
556   \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
557   \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_
558   }}{
559     \stex_kpsewhich:n{-var-value~PWD}
560   }
561
562 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
563 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
564 \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 78.)

## 27.3 File Hooks and Tracking

```

565 <@@=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

```

566 \seq_gclear_new:N\g__stex_files_stack

```

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`  
`\c_stex_mainfile_str`

```

567 \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
568 \stex_path_from_string:Nn \c_stex_mainfile_seq
569   \c_stex_mainfile_str

```

(End definition for `\c_stex_mainfile_seq` and `\c_stex_mainfile_str`. These variables are documented on page 78.)

`\g_stex_currentfile_seq`

```

570 \seq_gclear_new:N\g_stex_currentfile_seq

```

(End definition for `\g_stex_currentfile_seq`. This variable is documented on page 79.)

`\stex_filestack_push:n`

```

571 \cs_new_protected:Nn \stex_filestack_push:n {
572   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
573   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
574     \stex_path_from_string:Nn\g_stex_currentfile_seq{
575       \c_stex_pwd_str/#1
576     }
577   }
578   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
579   \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
580   \stex_get_document_uri:
581 }

```

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

`\stex_filestack_pop:`

```

582 \cs_new_protected:Nn \stex_filestack_pop: {
583   \seq_if_empty:NF\g__stex_files_stack{
584     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
585   }
586   \seq_if_empty:NTF\g__stex_files_stack{
587     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
588   }{
589     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
590     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
591   }
592   \stex_get_document_uri:
593 }

```

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

Hooks for the current file:

```

594 \AddToHook{file/before}{
595   \tl_if_empty:NTF\CurrentFilePath{
596     \stex_filestack_push:n{\CurrentFile}
597   }{
598     \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
599   }
600 }
601 \AddToHook{file/after}{
602   \stex_filestack_pop:
603 }

```

## 27.4 MathHub Repositories

604 `<@=stex_mathhub>`

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

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

```

605 \str_if_empty:NTF\mathhub{
606   \sys_if_platform_windows:TF{
607     \begingroup\escapechar=-1\catcode'\=12

```

```

608 \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str MATHHUB\c_percent_str}
609 \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
610 \exp_args:NNx\str_if_eq:ont\l_stex_kpsewhich_return_str{\c_percent_str MATHHUB\c_percent
611 \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_st
612 }{
613 \stex_kpsewhich:n{-var-value-MATHHUB}
614 }
615 \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
616
617 \str_if_empty:NT \c_stex_mathhub_str {
618 \sys_if_platform_windows:TF{
619 \begingroup\escapechar=-1\catcode'\=12
620 \exp_args:Nx\stex_kpsewhich:n{-var-value-HOME}
621 \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
622 \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_s
623 }{
624 \stex_kpsewhich:n{-var-value-HOME}
625 }
626 \ior_open:NnT \g_tmpa_ior{\l_stex_kpsewhich_return_str / .stex / mathhub.path}{
627 \begingroup\escapechar=-1\catcode'\=12
628 \ior_str_get:NN \g_tmpa_ior \l_tmpa_str
629 \sys_if_platform_windows:T{
630 \exp_args:NNx\str_replace_all:Nnn\l_tmpa_str{\c_backslash_str}/
631 }
632 \str_gset_eq:NN \c_stex_mathhub_str\l_tmpa_str
633 \endgroup
634 \ior_close:N \g_tmpa_ior
635 }
636 }
637 \str_if_empty:NTF\c_stex_mathhub_str{
638 \msg_warning:nn{stex}{warning/nomathhub}
639 }{
640 \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
641 \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
642 }
643 }{
644 \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
645 \stex_path_if_absolute:NF \c_stex_mathhub_seq {
646 \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
647 \c_stex_pwd_str/\mathhub
648 }
649 }
650 \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
651 \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
652 }

```

(End definition for `\mathhub`, `\c_stex_mathhub_seq`, and `\c_stex_mathhub_str`. These variables are documented on page 79.)

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

```

653 \cs_new_protected:Nn \_stex_mathhub_do_manifest:n {
654 \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
655 \str_set:Nx \l_tmpa_str { #1 }

```

```

656 \prop_new:c { c_stex_mathhub_#1_manifest_prop }
657 \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
658 \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
659 \__stex_mathhub_find_manifest:N \l_tmpa_seq
660 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
661   \msg_error:nnxx{stex}{error/norepository}{#1}{
662     \stex_path_to_string:N \c_stex_mathhub_str
663   }
664   \input{Fatal-Error!}
665 } {
666   \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
667 }
668 }
669 }

```

(End definition for \\_\_stex\_mathhub\_do\_manifest:n.)

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

```

670 \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:

```

671 \cs_new_protected:Nn \__stex_mathhub_find_manifest:N {
672   \seq_set_eq:NN\l_tmpa_seq #1
673   \bool_set_true:N\l_tmpa_bool
674   \bool_while_do:Nn \l_tmpa_bool {
675     \seq_if_empty:NTF \l_tmpa_seq {
676       \bool_set_false:N\l_tmpa_bool
677     }{
678       \file_if_exist:nTF{
679         \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
680       }{
681         \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
682         \bool_set_false:N\l_tmpa_bool
683       }{
684         \file_if_exist:nTF{
685           \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
686         }{
687           \seq_put_right:Nn\l_tmpa_seq{META-INF}
688           \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
689           \bool_set_false:N\l_tmpa_bool
690         }{
691           \file_if_exist:nTF{
692             \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
693           }{
694             \seq_put_right:Nn\l_tmpa_seq{meta-inf}
695             \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
696             \bool_set_false:N\l_tmpa_bool
697           }{
698             \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
699           }
700         }

```



```

701     }
702   }
703 }
704 \seq_set_eq:NN \l__stex_mathhub_manifest_file_seq \l_tmpa_seq
705 }

```

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

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

```

706 \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:

```

707 \cs_new_protected:Nn \__stex_mathhub_parse_manifest:n {
708   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
709   \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
710   \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
711     \str_set:Nn \l_tmpa_str {##1}
712     \exp_args:NNoo \seq_set_split:Nnn
713       \l_tmpb_seq \c_colon_str \l_tmpa_str
714     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
715       \exp_args:NNe \str_set:Nn \l_tmpb_tl {
716         \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
717       }
718       \exp_args:No \str_case:nnTF \l_tmpa_tl {
719         {id} {
720           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
721             { id } \l_tmpb_tl
722         }
723         {narration-base} {
724           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
725             { narr } \l_tmpb_tl
726         }
727         {url-base} {
728           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
729             { docurl } \l_tmpb_tl
730         }
731         {source-base} {
732           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
733             { ns } \l_tmpb_tl
734         }
735         {ns} {
736           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
737             { ns } \l_tmpb_tl
738         }
739         {dependencies} {
740           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
741             { deps } \l_tmpb_tl
742         }
743       }{}{}
744     }{}
745   }
746   \ior_close:N \c__stex_mathhub_manifest_ior

```

```

747 \stex_persist:x {
748   \prop_set_from_keyval:cn{ c_stex_mathhub_#1_manifest_prop }{
749     \exp_after:wN \prop_to_keyval:N \csname c_stex_mathhub_#1_manifest_prop\endcsname
750   }
751 }
752 }

```

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

`\stex_set_current_repository:n`

```

753 \cs_new_protected:Nn \stex_set_current_repository:n {
754   \stex_require_repository:n { #1 }
755   \prop_set_eq:Nc \l_stex_current_repository_prop {
756     c_stex_mathhub_#1_manifest_prop
757   }
758 }

```

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

`\stex_require_repository:n`

```

759 \cs_new_protected:Nn \stex_require_repository:n {
760   \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
761     \stex_debug:nn{mathhub}{Opening~archive:~#1}
762     \__stex_mathhub_do_manifest:n { #1 }
763   }
764 }

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

765 %\prop_new:N \l_stex_current_repository_prop
766 \bool_if:NF \c_stex_persist_mode_bool {
767   \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
768   \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
769     \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
770   } {
771     \__stex_mathhub_parse_manifest:n { main }
772     \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
773     \l_tmpa_str
774     \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str_manifest_prop }
775     \c_stex_mathhub_main_manifest_prop
776     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
777     \stex_debug:nn{mathhub}{Current~repository:~
778       \prop_item:Nn \l_stex_current_repository_prop {id}
779     }
780   }
781 }

```

(End definition for `\l_stex_current_repository_prop`. This variable is documented on page 79.)

`\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.

```

782 \cs_new_protected:Nn \stex_in_repository:nn {
783   \str_set:Nx \l_tmpa_str { #1 }
784   \cs_set:Npn \l_tmpa_cs ##1 { #2 }

```

```

785 \str_if_empty:NTF \l_tmpa_str {
786   \prop_if_exist:NTF \l_stex_current_repository_prop {
787     \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
788     \exp_args:Ne \l_tmpa_cs{
789       \prop_item:Nn \l_stex_current_repository_prop { id }
790     }
791   }{
792     \l_tmpa_cs{}
793   }
794 }{
795   \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}
796   \stex_require_repository:n \l_tmpa_str
797   \str_set:Nx \l_tmpa_str { #1 }
798   \exp_args:Nne \use:nn {
799     \stex_set_current_repository:n \l_tmpa_str
800     \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
801   }{
802     \stex_debug:nn{mathhub}{switching~back~to:~
803     \prop_if_exist:NTF \l_stex_current_repository_prop {
804       \prop_item:Nn \l_stex_current_repository_prop { id }::~
805     \meaning\l_stex_current_repository_prop
806     }{
807       no~repository
808     }
809   }
810   \prop_if_exist:NTF \l_stex_current_repository_prop {
811     \stex_set_current_repository:n {
812       \prop_item:Nn \l_stex_current_repository_prop { id }
813     }
814   }{
815     \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
816   }
817 }
818 }
819 }

```

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

## 27.5 Using Content in Archives

`\mhpath`

```

820 \def \mhpath #1 #2 {
821   \exp_args:Ne \tl_if_empty:NTF{#1}{
822     \c_stex_mathhub_str /
823     \prop_item:Nn \l_stex_current_repository_prop { id }
824     / source / #2
825   }{
826     \c_stex_mathhub_str / #1 / source / #2
827   }
828 }

```

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

**\inputref**  
**\mhinput**

```

829 \newif \ifinputref \inputreffalse
830
831 \cs_new_protected:Nn \__stex_mathhub_mhinput:nn {
832   \stex_in_repository:nn {#1} {
833     \ifinputref
834       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
835     \else
836       \inputreftrue
837       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
838     \inputreffalse
839   \fi
840 }
841 }
842 \NewDocumentCommand \mhinput { 0{} m}{
843   \__stex_mathhub_mhinput:nn{ #1 }{ #2 }
844 }
845
846 \cs_new_protected:Nn \__stex_mathhub_inputref:nn {
847   \stex_in_repository:nn {#1} {
848     \stex_html_backend:TF {
849       \str_clear:N \l_tmpa_str
850       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
851         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
852       }
853
854       \tl_if_empty:nTF{ ##1 }{
855         \IfFileExists{#2}{
856           \stex_annotate_invisible:nnn{inputref}{
857             \l_tmpa_str / #2
858           }{}
859         }{
860           \input{#2}
861         }
862       }{
863         \IfFileExists{ \c_stex_mathhub_str / ##1 / source / #2 }{
864           \stex_annotate_invisible:nnn{inputref}{
865             \l_tmpa_str / #2
866           }{}
867         }{
868           \input{ \c_stex_mathhub_str / ##1 / source / #2 }
869         }
870       }
871
872     }{
873       \begingroup
874       \inputreftrue
875       \tl_if_empty:nTF{ ##1 }{
876         \input{#2}
877       }{
878         \input{ \c_stex_mathhub_str / ##1 / source / #2 }
879       }
880       \endgroup
881     }

```

```

882 }
883 }
884 \NewDocumentCommand \inputref { 0{ } m }{
885   \__stex_mathhub_inputref:nn{ #1 }{ #2 }
886 }

```

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

### `\addmhbibresource`

```

887 \cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {
888   \stex_in_repository:nn {#1} {
889     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
890   }
891 }
892 \newcommand\addmhbibresource[2][]{
893   \__stex_mathhub_mhbibresource:nn{ #1 }{ #2 }
894 }

```

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

### `\libinput`

```

895 \cs_new_protected:Npn \libinput #1 {
896   \prop_if_exist:NF \l_stex_current_repository_prop {
897     \msg_error:nnn{stex}{error/notinarchive}\libinput
898   }
899   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
900     \msg_error:nnn{stex}{error/notinarchive}\libinput
901   }
902   \seq_clear:N \l__stex_mathhub_libinput_files_seq
903   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
904   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
905
906   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
907     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}
908     \IfFileExists{ \l_tmpa_str }{
909       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
910     }{}
911     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
912     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
913   }
914
915   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
916   \IfFileExists{ \l_tmpa_str }{
917     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
918   }{}
919
920   \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
921     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
922   }{
923     \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
924       \input{ ##1 }
925     }
926   }
927 }

```

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

## `\libusepackage`

```

928 \NewDocumentCommand \libusepackage {0{} m} {
929   \prop_if_exist:NF \l_stex_current_repository_prop {
930     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
931   }
932   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
933     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
934   }
935   \seq_clear:N \l__stex_mathhub_libinput_files_seq
936   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
937   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
938
939   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
940     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2}
941     \IfFileExists{ \l_tmpa_str.sty }{
942       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
943     }{}
944     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
945     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
946   }
947
948   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2}
949   \IfFileExists{ \l_tmpa_str.sty }{
950     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
951   }{}
952
953   \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
954     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
955   }{
956     \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
957       \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
958         \usepackage[#1]{ ##1 }
959       }
960     }{
961       \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
962     }
963   }
964 }

```

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

## `\mhgraphics`

## `\cmhgraphics`

```

965
966 \AddToHook{begindocument}{
967   \ltx@ifpackageloaded{graphicx}{
968     \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
969     \providecommand\mhgraphics[2][]{%
970       \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
971       \includegraphics[#1]{\mhp@h\Gin@mhrepos{#2}}
972     }
973   }

```

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

`\lstinputmhlisting`  
`\clstinputmhlisting`

```
974 \ltx@ifpackageloaded{listings}{  
975   \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}  
976   \newcommand\lstinputmhlisting[2][]{%  
977     \def\lst@mhrepos{}\setkeys{lst}{#1}%  
978     \lstinputlisting[#1]{\mhpath\lst@mhrepos{#2}}}  
979   \newcommand\clstinputmhlisting[2][]{\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}  
980 }{}  
981 }  
982  
983 \end{package}
```

*(End definition for `\lstinputmhlisting` and `\clstinputmhlisting`. These functions are documented on page 80.)*

## Chapter 28

# STEX -References Implementation

```
984 <*package>
985
986 %%%%%%%%% stex-references.dtx %%%%%%%%%
987
988 <@@=stex_refs>
989
990 Warnings and error messages
991 \msg_new:nnn{stex}{error/extrefmissing}{
992   Missing~in~or~cite~value~for~\detokenize{\extref}!
993 }
994 \msg_new:nnn{stex}{warning/smsmissing}{
995   .sref~file~#1~doesn't~exist!
996 }
997 \msg_new:nnn{stex}{warning/smslabelmissing}{
998   No~label~#2~in~.sref~file~#1!
999 }
```

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

```
998 \iow_new:N \c__stex_refs_refs_iow
999 \AtBeginDocument{
1000   \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
1001 }
1002 \AtEndDocument{
1003   \iow_close:N \c__stex_refs_refs_iow
1004 }
```

### 28.1 Document URIs and URLs

`\l_stex_current_docns_str`

```
1005 \str_new:N \l_stex_current_docns_str
```

*(End definition for `\l_stex_current_docns_str`. This variable is documented on page 81.)*

`\stex_get_document_uri:`

```
1006 \cs_new_protected:Nn \stex_get_document_uri: {
```



```

1007 \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1008 \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1009 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1010 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1011 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1012
1013 \str_clear:N \l_tmpa_str
1014 \prop_if_exist:NT \l_stex_current_repository_prop {
1015   \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
1016     \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
1017   }
1018 }
1019
1020 \str_if_empty:NTF \l_tmpa_str {
1021   \str_set:Nx \l_stex_current_docns_str {
1022     file:/\stex_path_to_string:N \l_tmpa_seq
1023   }
1024 }{
1025   \bool_set_true:N \l_tmpa_bool
1026   \bool_while_do:Nn \l_tmpa_bool {
1027     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1028     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1029       {source} { \bool_set_false:N \l_tmpa_bool }
1030     }{}{
1031       \seq_if_empty:NT \l_tmpa_seq {
1032         \bool_set_false:N \l_tmpa_bool
1033       }
1034     }
1035   }
1036
1037   \seq_if_empty:NTF \l_tmpa_seq {
1038     \str_gset_eq:NN \l_stex_current_docns_str \l_tmpa_str
1039   }{
1040     \str_gset:Nx \l_stex_current_docns_str {
1041       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
1042     }
1043   }
1044 }
1045 %\stex_get_document_url:
1046 }

```

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

`\l_stex_current_docurl_str`

```

1047 \str_new:N \l_stex_current_docurl_str

```

(End definition for `\l_stex_current_docurl_str`. This variable is documented on page 81.)

`\stex_get_document_url:`

```

1048 \cs_new_protected:Nn \stex_get_document_url: {
1049   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1050   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1051   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1052   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1053   \seq_put_right:No \l_tmpa_seq \l_tmpb_str

```

```

1054 \str_clear:N \l_tmpa_str
1055 \prop_if_exist:NT \l_stex_current_repository_prop {
1056   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
1057     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
1058       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
1059     }
1060   }
1061 }
1062 }
1063
1064 \str_if_empty:NTF \l_tmpa_str {
1065   \str_set:Nx \l_stex_current_docurl_str {
1066     file:/\stex_path_to_string:N \l_tmpa_seq
1067   }
1068 }{
1069   \bool_set_true:N \l_tmpa_bool
1070   \bool_while_do:Nn \l_tmpa_bool {
1071     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1072     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1073       {source} { \bool_set_false:N \l_tmpa_bool }
1074     }{}{
1075       \seq_if_empty:NT \l_tmpa_seq {
1076         \bool_set_false:N \l_tmpa_bool
1077       }
1078     }
1079   }
1080
1081   \seq_if_empty:NTF \l_tmpa_seq {
1082     \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
1083   }{
1084     \str_set:Nx \l_stex_current_docurl_str {
1085       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
1086     }
1087   }
1088 }
1089 }

```

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

## 28.2 Setting Reference Targets

```

1090 \str_const:Nn \c__stex_refs_url_str{URL}
1091 \str_const:Nn \c__stex_refs_ref_str{REF}
1092 \str_new:N \l__stex_refs_curr_label_str
1093 % @currentlabel -> number
1094 % @currentlabelname -> title
1095 % @currentHref -> name.number <- id of some kind
1096 % @currentcounter <- name/id
1097 % \#autorefname <- "Section"
1098 % \theH# -> \arabic{section}
1099 % \the# -> number
1100 % \hyper@makecurrent{#}
1101 \int_new:N \l__stex_refs_unnamed_counter_int

```

Restoring references from .sref-files

\STEXInternalSrefRestoreTarget

1102 \cs\_new\_protected:Npn \STEXInternalSrefRestoreTarget #1#2#3#4#5 {}

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

\stex\_ref\_new\_doc\_target:n

```

1103 \seq_new:N \g_stex_ref_files_seq
1104
1105 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
1106   %\stex_get_document_uri:
1107   \str_clear:N \l__stex_refs_curr_label_str
1108   \str_set:Nx \l_tmpa_str { #1 }
1109   \str_if_empty:NT \l_tmpa_str {
1110     \int_gincr:N \l__stex_refs_unnamed_counter_int
1111     \str_set:Nx \l_tmpa_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
1112   }
1113   \str_set:Nx \l__stex_refs_curr_label_str {
1114     \l_stex_current_docns_str?\l_tmpa_str
1115   }
1116
1117   \exp_args:Noo \STEXInternalAuxAddDocRef\l_stex_current_docns_str\l_tmpa_str
1118
1119   %\seq_if_exist:cF{g__stex_refs_labels_\l_tmpa_str_seq}{
1120   %   \seq_new:c {g__stex_refs_labels_\l_tmpa_str_seq}
1121   %}
1122   %\seq_if_in:coF{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str {
1123   %   \seq_gput_right:co{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str
1124   %}
1125
1126
1127   \stex_if_smsmode:TF {
1128     %\stex_get_document_url:
1129     %\str_gset_eq:cN {sref_url_\l__stex_refs_curr_label_str_str}\l_stex_current_docurl_str
1130     %\str_gset_eq:cN {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_url_str
1131   }{
1132     \iow_now:Nx \c__stex_refs_refs_iow {
1133       \STEXInternalSrefRestoreTarget
1134       {\l_stex_current_docns_str}
1135       {\l_tmpa_str}
1136       {\@currentcounter}
1137       {\@currentlabel}
1138       {\tl_if_exist:NT\@currentlabelname{\exp_args:No\unexpanded\@currentlabelname}}
1139     }
1140     %\iow_now:Nx \c__stex_refs_refs_iow {
1141     %   {\l_stex_current_docns_str?\l_tmpa_str}~=={\use:c{\@currentcounter autorefname}~\@currentlabel}
1142     %   \stex_debug:nn{sref}{New~label~\l__stex_refs_curr_label_str~at~\use:c{\use:c{\@currentcounter}~\@currentlabel}}
1143     %   \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
1144     %   \immediate\write\auxout{\STEXInternalAuxAddDocRef{\l_stex_current_docns_str}{\l_tmpa_str}{\l__stex_refs_curr_label_str}}
1145     %   \str_gset:cx {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_ref_str
1146     %}
1147   }
1148   \NewDocumentCommand \slabel {m} {\stex_ref_new_doc_target:n {#1}}

```

(End definition for `\stex_ref_new_doc_target:n`. This function is documented on page 81.)

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

```

1149 \cs_new_protected:Npn \STEXInternalAuxAddDocRef #1 #2 {
1150   \exp_args:NNx \seq_if_in:NnTF \g_stex_ref_files_seq {\detokenize{#1}} {
1151     \exp_args:Nnx \seq_if_in:cnF{g_stex_ref_ #1 _seq}{\detokenize{#2}}{
1152       \exp_args:Nnx \seq_gput_left:cn{g_stex_ref_ #1 _seq}{\detokenize{#2}}
1153     }
1154   }{
1155     \exp_args:NNx \seq_gput_right:Nn \g_stex_ref_files_seq {\detokenize{#1}}
1156     %\seq_if_exist:cF{g_stex_ref_ #1 _seq}{
1157       \seq_new:c{g_stex_ref_ #1 _seq} % <- seq_new throws errors??
1158     %}
1159     \exp_args:Nnx \seq_gput_left:cn{g_stex_ref_ #1 _seq}{\detokenize{#2}}
1160   }
1161
1162   %\str_set:Nn \l_tmpa_str {#1?#2}
1163   %\str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
1164   %\seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
1165     % \seq_new:c {g__stex_refs_labels_#2_seq}
1166     %}
1167   %\seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
1168     % \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
1169     %}
1170 }

```

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

```

1171 \AtEndDocument{
1172   \def\STEXInternalAuxAddDocRef#1 #2 {}{}
1173 }

```

`\stex_ref_new_sym_target:n`

```

1174 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
1175
1176   % \stex_if_smsmode:TF {
1177   %   \str_if_exist:cF{sref_sym_#1_type}{
1178   %     \stex_get_document_url:
1179   %     \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str
1180   %     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
1181   %   }
1182   % }{
1183   %   \str_if_empty:NF \l__stex_refs_curr_label_str {
1184   %     \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
1185   %     \immediate\write\@auxout{
1186   %       \exp_not:N\expandafter\def\exp_not:N\csname \exp_not:N\detokenize{sref_sym_#1_label
1187   %         \l__stex_refs_curr_label_str
1188   %       }
1189   %     }
1190   %   }
1191   % }
1192 }

```

(End definition for `\stex_ref_new_sym_target:n`. This function is documented on page 81.)

## 28.3 Using References

**\sref** Optional arguments:

```

1193
1194 \keys_define:nn { stex / sref / 1 } {
1195   archive .str_set_x:N = \l__stex_refs_repo_str,
1196   file     .str_set_x:N = \l__stex_refs_file_str,
1197   % TODO get rid of this
1198   fallback .code:n = {},
1199   pre      .code:n = {},
1200   post     .code:n = {}
1201 }
1202 \cs_new_protected:Nn \__stex_refs_args_i:n {
1203   \str_clear:N \l__stex_refs_repo_str
1204   \str_clear:N \l__stex_refs_file_str
1205   \keys_set:nn { stex / sref / 1 } { #1 }
1206 }
1207 \keys_define:nn { stex / sref / 2 } {
1208   in       .str_set_x:N = \l__stex_refs_in_str,
1209   archive  .str_set_x:N = \l__stex_refs_repob_str,
1210   title    .tl_set:N = \l__stex_refs_title_tl
1211 }
1212 \cs_new_protected:Nn \__stex_refs_args_ii:n {
1213   \str_clear:N \l__stex_refs_in_str
1214   \tl_clear:N \l__stex_refs_title_tl
1215   \str_clear:N \l__stex_refs_repob_str
1216   \keys_set:nn { stex / sref / 2 } { #1 }
1217 }

```

The actual macro:

```

1218 \NewDocumentCommand \sref { 0{} m 0{} }{
1219   \__stex_refs_args_i:n{#1}
1220   \__stex_refs_args_ii:n{#3}
1221   \str_clear:N \l__stex_refs_uri_str
1222   \__stex_refs_find_uri:n{#2}
1223   \__stex_refs_do_sref:n{#2}
1224 }
1225 \NewDocumentCommand \extref { 0{} m m }{
1226   \__stex_refs_args_i:n{#1}
1227   \__stex_refs_args_ii:n{#3}
1228   \str_if_empty:NT \l__stex_refs_in_str {
1229     \msg_error:nn{stex}{error/extrefmissing}
1230   }
1231   \str_clear:N \l__stex_refs_uri_str
1232   \__stex_refs_find_uri:n{#2}
1233   \__stex_refs_do_sref_in:n{#2}
1234 }
1235
1236 \cs_new_protected:Nn \__stex_refs_find_uri:n {
1237   \stex_debug:nn{sref}{File:~\l__stex_refs_file_str^^JRepo:\l__stex_refs_repo_str}
1238   \str_if_empty:NTF \l__stex_refs_file_str {
1239     \stex_debug:nn{sref}{Empty.~Checking~current~file~for~#1}
1240     \seq_if_exist:cT{g_stex_ref_\l_stex_current_docns_str_seq}{
1241       \seq_map_inline:cn{g_stex_ref_\l_stex_current_docns_str_seq}{

```

```

1242     \str_if_eq:nnT{#1}{##1}{
1243       \str_set_eq:NN \l__stex_refs_uri_str \l_stex_current_docns_str
1244       \stex_debug:nn{sref}{Found.}
1245       \seq_map_break:
1246     }
1247   }
1248 }
1249 \str_if_empty:NT \l__stex_refs_uri_str {
1250   \stex_debug:nn{sref}{Checking~other~files}
1251   \seq_map_inline:Nn \g_stex_ref_files_seq {
1252     \stex_debug:nn{sref}{##1...}
1253     \seq_map_inline:cn{g_stex_ref_##1_seq}{
1254       \str_if_eq:nnT{#1}{####1}{
1255         \stex_debug:nn{sref}{Found~##1}
1256         \str_set:Nn \l__stex_refs_uri_str {##1}
1257         \seq_map_break:n{\seq_map_break:}
1258       }
1259     }
1260   }
1261 }
1262 }{
1263   \str_if_empty:NTF \l__stex_refs_repo_str {
1264     \prop_if_exist:NTF \l_stex_current_repository_prop {
1265       \stex_debug:nn{sref}{in~archive~\prop_item:Nn \l_stex_current_repository_prop { id }
1266       \prop_get:NnN \l_stex_current_repository_prop { ns } \l__stex_refs_uri_str
1267       \stex_debug:nn{sref}{namespace:~\l__stex_refs_uri_str}
1268       \str_set:Nx \l__stex_refs_uri_str {\l__stex_refs_uri_str / \l__stex_refs_file_str}
1269       \stex_path_from_string:Nn \l_tmpb_seq \l__stex_refs_uri_str
1270       \str_set:Nx \l__stex_refs_uri_str {\stex_path_to_string:N \l_tmpb_seq}
1271       \stex_debug:nn{sref}{Return:~\l__stex_refs_uri_str}
1272     }{
1273       \stex_debug:nn{sref}{Not~in~archive}
1274       \stex_path_from_string:Nn \l_tmpb_seq {
1275         \stex_path_to_string:N \g_stex_currentfile_seq/ .. / \l__stex_refs_file_str
1276       }
1277       \str_set:Nx \l__stex_refs_uri_str {file:~\stex_path_to_string:N \l_tmpb_seq}
1278     }
1279   }{
1280     \stex_require_repository:n \l__stex_refs_repo_str
1281     \prop_get:cnN { c_stex_mathhub_\l__stex_refs_repo_str _manifest_prop } { ns } \l__stex
1282     \str_set:Nx \l__stex_refs_uri_str {\l__stex_refs_uri_str / \l__stex_refs_file_str}
1283     \stex_path_from_string:Nn \l_tmpb_seq \l__stex_refs_uri_str
1284     \str_set:Nx \l__stex_refs_uri_str {\stex_path_to_string:N \l_tmpb_seq}
1285   }
1286 }
1287 }
1288
1289 \cs_new_protected:Nn \__stex_refs_do_autoref:n{
1290   \cs_if_exist:cTF{autoref}{
1291     \exp_args:Nx\autoref{sref_#1}
1292   }{
1293     \exp_args:Nx\ref{sref_#1}
1294   }
1295 }

```

```

1296
1297 \cs_new_protected:Nn \__stex_refs_do_sref:n {
1298   \str_if_empty:NTF \l__stex_refs_uri_str {
1299     \str_if_empty:NTF \l__stex_refs_in_str {
1300       \stex_debug:nn{sref}{autoref~on~#1}
1301       \__stex_refs_do_autoref:n{#1}
1302     }{
1303       \stex_debug:nn{sref}{srefin~on~#1}
1304       \__stex_refs_do_sref_in:n{#1}
1305     }
1306   }{
1307     \exp_args:NNo \seq_if_in:NnTF \g_stex_ref_files_seq \l__stex_refs_uri_str {
1308       \exp_args:Nnx \seq_if_in:cnTF{g_stex_ref\_l__stex_refs_uri_str_seq}{\detokenize{#1}}{
1309         \stex_debug:nn{sref}{Reference~found~in~ref~files;~autoref~on~\l__stex_refs_uri_str?#1}
1310         \__stex_refs_do_autoref:n{\l__stex_refs_uri_str?#1}
1311       }{
1312         \str_if_empty:NTF \l__stex_refs_in_str {
1313           \stex_debug:nn{sref}{in~empty;~autoref~on~\l__stex_refs_uri_str?#1}
1314           \__stex_refs_do_autoref:n{\l__stex_refs_uri_str?#1}
1315         }{
1316           \stex_debug:nn{sref}{in~non~empty;~srefin~on~\l__stex_refs_uri_str?#1}
1317           \__stex_refs_do_sref_in:n{#1}
1318         }
1319       }
1320     }{
1321       \str_if_empty:NTF \l__stex_refs_in_str {
1322         \stex_debug:nn{sref}{in~empty;~autoref~on~\l__stex_refs_uri_str?#1}
1323         \__stex_refs_do_autoref:n{\l__stex_refs_uri_str?#1}
1324       }{
1325         \stex_debug:nn{sref}{in~non~empty;~srefin~on~\l__stex_refs_uri_str?#1}
1326         \__stex_refs_do_sref_in:n{#1}
1327       }
1328     }
1329   }
1330 }
1331
1332 \cs_new_protected:Nn \__stex_refs_restore_target:nnnnn {
1333   \str_if_empty:NTF \l__stex_refs_uri_str {
1334     \exp_args:No \str_if_eq:nnT \l__stex_refs_id_str {#2}{
1335       \tl_set:Nn \l__stex_refs_return_tl {
1336         \use:c{#3autorefname}~#4\tl_if_empty:nF{#5}{~(5)}~in~
1337         \tl_if_empty:NTF\l__stex_refs_title_tl{
1338           ???
1339         }\l__stex_refs_title_tl
1340       }
1341     }
1342   }{
1343     \stex_debug:nn{sref}{\l__stex_refs_uri_str{~ == ~ #1 ~ ?}
1344     \exp_args:No \str_if_eq:nnT \l__stex_refs_uri_str {#1}{
1345       \stex_debug:nn{sref}{\l__stex_refs_id_str~ == ~ #2 ~ ?}
1346       \exp_args:No \str_if_eq:nnT \l__stex_refs_id_str {#2}{
1347         \stex_debug:nn{sref}{success!}
1348         \tl_set:Nn \l__stex_refs_return_tl {
1349           \use:c{#3autorefname}~#4\tl_if_empty:nF{#5}{~(5)}~in~

```

```

1350         \tl_if_empty:nTF\l__stex_refs_title_tl{
1351             ???
1352         }\l__stex_refs_title_tl
1353     }
1354     \endinput
1355 }
1356 }
1357 }
1358 }
1359
1360 \cs_new_protected:Nn \__stex_refs_do_sref_in:n {
1361     \stex_debug:nn{sref}{In: \l__stex_refs_in_str^^JRepo:\l__stex_refs_repo_str}
1362     \stex_debug:nn{sref}{URI: \l__stex_refs_uri_str?#1}
1363     %\msg_warning:nnn{stex}{warning/smsmissing}{<filename>}
1364     \begingroup\catcode13=9\relax\catcode10=9\relax
1365     \str_if_empty:NTF \l__stex_refs_repob_str {
1366         \prop_if_exist:NTF \l_stex_current_repository_prop {
1367             \str_set:Nx \l_tmpa_str {
1368                 \c_stex_mathhub_str /
1369                 \prop_item:Nn \l_stex_current_repository_prop { id }
1370                 / source / \l__stex_refs_in_str .sref
1371             }
1372         }{
1373             \str_set:Nx \l_tmpa_str {
1374                 \stex_path_to_string:N \g_stex_currentfile_seq/ .. / \l__stex_refs_in_str . sref
1375             }
1376         }
1377     }{
1378         \str_set:Nx \l_tmpa_str {
1379             \c_stex_mathhub_str / \l__stex_refs_repob_str
1380             / source / \l__stex_refs_in_str . sref
1381         }
1382     }
1383     \stex_path_from_string:Nn \l_tmpb_seq \l_tmpa_str
1384     \stex_path_to_string:NN \l_tmpb_seq \l_tmpa_str
1385     \stex_debug:nn{sref}{File: \l_tmpa_str}
1386     \exp_args:No \IfFileExists \l_tmpa_str {
1387         \tl_clear:N \l__stex_refs_return_tl
1388         \str_set:Nn \l__stex_refs_id_str {#1}
1389         \let\STEXInternalSrefRestoreTarget\__stex_refs_restore_target:nnnnn
1390         \use:c{@ @ input}{\l_tmpa_str}
1391         \exp_args:No \tl_if_empty:nTF \l__stex_refs_return_tl {
1392             \exp_args:Nnno \msg_warning:nnnn{stex}{warning/smslabelmissing}\l_tmpa_str{#1}
1393             \__stex_refs_do_autoref:n{
1394                 \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
1395             }
1396         }{
1397             \l__stex_refs_return_tl
1398         }
1399     }{
1400         \exp_args:Nnno \msg_warning:nnn{stex}{warning/smsmissing}\l_tmpa_str
1401         \__stex_refs_do_autoref:n{
1402             \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
1403         }

```



```

1404     }
1405 \endgroup
1406 }
1407
1408 % \__stex_refs_args:n { #1 }
1409 % \str_if_empty:NTF \l__stex_refs_indocument_str {
1410 %   \str_set:Nx \l_tmpa_str { #2 }
1411 %   \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
1412 %   \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
1413 %     \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str_seq}{
1414 %       \seq_get_left:cNF {g__stex_refs_labels_\l_tmpa_str_seq} \l_tmpa_str {
1415 %         \str_clear:N \l_tmpa_str
1416 %       }
1417 %     }{
1418 %       \str_clear:N \l_tmpa_str
1419 %     }
1420 %   }{
1421 %     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1422 %     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1423 %     \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
1424 %     \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str_seq}{
1425 %       \str_set_eq:NN \l_tmpc_str \l_tmpa_str
1426 %       \str_clear:N \l_tmpa_str
1427 %       \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str_seq} {
1428 %         \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
1429 %           \str_range:nnn { ##1 }{-\l_tmpa_int}{ -1 }
1430 %         }{
1431 %           \seq_map_break:n {
1432 %             \str_set:Nn \l_tmpa_str { ##1 }
1433 %           }
1434 %         }
1435 %       }
1436 %     }{
1437 %       \str_clear:N \l_tmpa_str
1438 %     }
1439 %   }
1440 % \str_if_empty:NTF \l_tmpa_str {
1441 %   \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_li
1442 % }{
1443 %   \str_if_eq:cNTF {sref_\l_tmpa_str_type} \c__stex_refs_ref_str {
1444 %     \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1445 %       \cs_if_exist:cTF{autoref}{
1446 %         \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1447 %       }{
1448 %         \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1449 %       }
1450 %     }{
1451 %       \ltx@ifpackageloaded{hyperref}{
1452 %         \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
1453 %       }{
1454 %         \l__stex_refs_linktext_tl
1455 %       }
1456 %     }
1457 %   }{

```

```

1458 % \ltx@ifpackageloaded{hyperref}{
1459 % \href{\use:c{sref_url_1\l_tmpa_str _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_t1
1460 % }{
1461 % \tl_if_empty:NTF \l__stex_refs_linktext_t1 \l__stex_refs_fallback_t1 \l__stex_refs_fallback_t1
1462 % }
1463 % }
1464 % }
1465 % }{
1466 % TODO
1467 % }
1468 %}

```

(End definition for `\sref`. This function is documented on page 82.)

### `\srefsym`

```

1469 \NewDocumentCommand \srefsym { 0{} m}{
1470 \stex_get_symbol:n { #2 }
1471 \__stex_refs_sym_aux:nn{#1}{\l_stex_get_symbol_uri_str}
1472 }
1473
1474 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1475
1476 % \str_if_exist:cTF {sref_sym_#2 _label_str }{
1477 % \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1478 % }{
1479 % \__stex_refs_args:n { #1 }
1480 % \str_if_empty:NTF \l__stex_refs_indocument_str {
1481 % \tl_if_exist:cTF{sref_sym_#2 _type}{
1482 % % doc uri in \l_tmpb_str
1483 % \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1484 % \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1485 % % reference
1486 % \tl_if_empty:NTF \l__stex_refs_linktext_t1 {
1487 % \cs_if_exist:cTF{autoref}{
1488 % \l__stex_refs_pre_t1\autoref{sref_sym_#2}\l__stex_refs_post_t1
1489 % }{
1490 % \l__stex_refs_pre_t1\ref{sref_sym_#2}\l__stex_refs_post_t1
1491 % }
1492 % }{
1493 % \ltx@ifpackageloaded{hyperref}{
1494 % \hyperref[sref_sym_#2]\l__stex_refs_linktext_t1
1495 % }{
1496 % \l__stex_refs_linktext_t1
1497 % }
1498 % }
1499 % }{
1500 % % URL
1501 % \ltx@ifpackageloaded{hyperref}{
1502 % \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_t1
1503 % }{
1504 % \tl_if_empty:NTF \l__stex_refs_linktext_t1 \l__stex_refs_fallback_t1 \l__stex_refs_fallback_t1
1505 % }
1506 % }
1507 % }{

```

```

1508 %          \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_
1509 %          }
1510 %      }{
1511 %          % TODO
1512 %      }
1513 %  }
1514 }

```

*(End definition for \srefsym. This function is documented on page 82.)*

**\srefsymuri**

```

1515 \cs_new_protected:Npn \srefsymuri #1 #2 { % TODO
1516   #2%\__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1517 }

```

*(End definition for \srefsymuri. This function is documented on page 82.)*

```

1518 </package>

```

## Chapter 29

# STEX -Modules Implementation

```
1519 <*package>
1520
1521 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1522
1523 <@@=stex_modules>
1524
1525 Warnings and error messages
1526 \msg_new:nnn{stex}{error/unknownmodule}{
1527   No~module~#1~found
1528 }
1529 \msg_new:nnn{stex}{error/syntax}{
1530   Syntax~error:~#1
1531 }
1532 \msg_new:nnn{stex}{error/siglanguage}{
1533   Module~#1~declares~signature~#2,~but~does~not~
1534   declare~its~language
1535 }
1536 \msg_new:nnn{stex}{warning/deprecated}{
1537   #1~is~deprecated;~please~use~#2~instead!
1538 }
1539 \msg_new:nnn{stex}{error/conflictingmodules}{
1540   Conflicting~imports~for~module~#1
1541 }
```

`\l_stex_current_module_str` The current module:

```
1541 \str_new:N \l_stex_current_module_str
```

*(End definition for \l\_stex\_current\_module\_str. This variable is documented on page 84.)*

`\l_stex_all_modules_seq` Stores all available modules

```
1542 \seq_new:N \l_stex_all_modules_seq
```

*(End definition for \l\_stex\_all\_modules\_seq. This variable is documented on page 84.)*

```

\stex_if_in_module_p:
\stex_if_in_module:TF
1543 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1544   \str_if_empty:NTF \l_stex_current_module_str
1545   \prg_return_false: \prg_return_true:
1546 }

```

(End definition for `\stex_if_in_module:TF`. This function is documented on page 84.)

```

\stex_if_module_exists_p:n
\stex_if_module_exists:nTF
1547 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1548   \prop_if_exist:cTF { c_stex_module_#1_prop }
1549   \prg_return_true: \prg_return_false:
1550 }

```

(End definition for `\stex_if_module_exists:nTF`. This function is documented on page 84.)

`\stex_add_to_current_module:n` Only allowed within modules:

```

\STEXexport
1551 \cs_new_protected:Nn \stex_execute_in_module:n { \stex_if_in_module:T {
1552   \stex_add_to_current_module:n { #1 }
1553   \stex_do_up_to_module:n { #1 }
1554 }}
1555 \cs_generate_variant:Nn \stex_execute_in_module:n {x}
1556
1557 \cs_new_protected:Nn \stex_add_to_current_module:n {
1558   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str_code} { #1 }
1559 }
1560 \cs_generate_variant:Nn \stex_add_to_current_module:n {x}
1561 \cs_new_protected:Npn \STEXexport {
1562   \ExplSyntaxOn
1563   \__stex_modules_export:n
1564 }
1565 \cs_new_protected:Nn \__stex_modules_export:n {
1566   \ignorespacesandpars#1\ExplSyntaxOff
1567   \stex_add_to_current_module:n { \ignorespacesandpars#1}
1568   \stex_smsmode_do:
1569 }
1570 \let \stex_module_export_helper:n \use:n
1571 \stex_deactivate_macro:Nn \STEXexport {module~environments}

```

(End definition for `\stex_add_to_current_module:n` and `\STEXexport`. These functions are documented on page 84.)

```

\stex_add_constant_to_current_module:n
1572 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1573   \str_set:Nx \l_tmpa_str { #1 }
1574   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str_constants} { \l_tmpa_str }
1575 }

```

(End definition for `\stex_add_constant_to_current_module:n`. This function is documented on page 84.)

```

\stex_add_import_to_current_module:n
1576 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1577   \str_set:Nx \l_tmpa_str { #1 }
1578   \exp_args:Nno

```

```

1579 \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1580 \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1581 }
1582 }

```

(End definition for `\stex_add_import_to_current_module:n`. This function is documented on page 84.)

`\stex_collect_imports:n`

```

1583 \cs_new_protected:Nn \stex_collect_imports:n {
1584 \seq_clear:N \l_stex_collect_imports_seq
1585 \__stex_modules_collect_imports:n {#1}
1586 }
1587 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1588 \seq_map_inline:cn {c_stex_module_#1_imports} {
1589 \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1590 \__stex_modules_collect_imports:n { ##1 }
1591 }
1592 }
1593 \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1594 \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1595 }
1596 }

```

(End definition for `\stex_collect_imports:n`. This function is documented on page 84.)

`\stex_do_up_to_module:n`

```

1597 \int_new:N \l__stex_modules_group_depth_int
1598 \cs_new_protected:Nn \stex_do_up_to_module:n {
1599 \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1600 #1
1601 }{
1602 #1
1603 \expandafter \tl_gset:Nn
1604 \csname l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1605 \expandafter\expandafter\expandafter\endcsname
1606 \expandafter\expandafter\expandafter { \csname
1607 l__stex_modules_aftergroup_\l_stex_current_module_str _tl\endcsname #1 }
1608 \aftergroup\__stex_modules_aftergroup_do:
1609 }
1610 }
1611 \cs_generate_variant:Nn \stex_do_up_to_module:n {x}
1612 \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1613 \stex_debug:nn{aftergroup}{\cs_meaning:c{
1614 l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1615 }}}
1616 \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1617 \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1618 \tl_gclear:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1619 }{
1620 \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1621 \aftergroup\__stex_modules_aftergroup_do:
1622 }
1623 }
1624 \cs_new_protected:Nn \stex_reset_up_to_module:n {
1625 \expandafter\let\csname l__stex_modules_aftergroup_#1_tl\endcsname\undefined

```

1626 }

(End definition for `\stex_do_up_to_module:n`. This function is documented on page 84.)

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

1627

(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.

```
1628 \str_new:N \l_stex_module_ns_str
1629 \str_new:N \l_stex_module_subpath_str
1630 \cs_new_protected:Nn \__stex_modules_compute_namespace:nN {
1631   \seq_set_eq:NN \l_tmpa_seq #2
1632   % split off file extension
1633   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str % <- filename
1634   \exp_args:Nnno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1635   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str % <- filename without suffixes
1636   \seq_put_right:No \l_tmpa_seq \l_tmpb_str % <- file path including name without suffixes
1637
1638   \bool_set_true:N \l_tmpa_bool
1639   \bool_while_do:Nn \l_tmpa_bool {
1640     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1641     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1642       {source} { \bool_set_false:N \l_tmpa_bool }
1643     }{}{
1644       \seq_if_empty:NT \l_tmpa_seq {
1645         \bool_set_false:N \l_tmpa_bool
1646       }
1647     }
1648   }
1649
1650   \stex_path_to_string:NN \l_tmpa_seq \l_stex_module_subpath_str
1651   % \l_tmpa_seq <- sub-path relative to archive
1652   \str_if_empty:NTF \l_stex_module_subpath_str {
1653     \str_set:Nx \l_stex_module_ns_str {#1}
1654   }{
1655     \str_set:Nx \l_stex_module_ns_str {
1656       #1/\l_stex_module_subpath_str
1657     }
1658   }
1659 }
1660
1661 \cs_new_protected:Nn \stex_modules_current_namespace: {
1662   \str_clear:N \l_stex_module_subpath_str
1663   \prop_if_exist:NTF \l_stex_current_repository_prop {
1664     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1665     \__stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1666   }{
1667     % split off file extension
1668     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1669     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
```

```

1670 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1671 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1672 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1673 \str_set:Nx \l_stex_module_ns_str {
1674   file:/\stex_path_to_string:N \l_tmpa_seq
1675 }
1676 }
1677 }

```

(End definition for `\stex_modules_current_namespace:.` This function is documented on page 85.)

## 29.1 The smodule environment

smodule arguments:

```

1678 \keys_define:nn { stex / module } {
1679   title      .tl_set:N      = \smodulename ,
1680   type       .str_set_x:N   = \smodulename ,
1681   id         .str_set_x:N   = \smoduleid ,
1682   deprecate  .str_set_x:N   = \l_stex_module_deprecate_str ,
1683   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1684   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1685   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1686   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1687   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1688   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1689   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1690 }
1691
1692 \cs_new_protected:Nn \__stex_modules_args:n {
1693   \str_clear:N \smodulename
1694   \str_clear:N \smodulename
1695   \str_clear:N \smoduleid
1696   \str_clear:N \l_stex_module_ns_str
1697   \str_clear:N \l_stex_module_deprecate_str
1698   \str_clear:N \l_stex_module_lang_str
1699   \str_clear:N \l_stex_module_sig_str
1700   \str_clear:N \l_stex_module_creators_str
1701   \str_clear:N \l_stex_module_contributors_str
1702   \str_clear:N \l_stex_module_meta_str
1703   \str_clear:N \l_stex_module_srccite_str
1704   \keys_set:nn { stex / module } { #1 }
1705 }
1706
1707 % module parameters here? In the body?
1708

```

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

```

1709 \cs_new_protected:Nn \stex_module_setup:nn {
1710   \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1711   \str_set:Nx \l_stex_module_name_str { #2 }
1712   \__stex_modules_args:n { #1 }

```



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

Are we in a nested module?

```

1713 \stex_if_in_module:TF {
1714   % Nested module
1715   \prop_get:cnN {c_stex_module\_l_stex_current_module_str _prop}
1716   { ns } \l_stex_module_ns_str
1717   \str_set:Nx \l_stex_module_name_str {
1718     \prop_item:cn {c_stex_module\_l_stex_current_module_str _prop}
1719     { name } / \l_stex_module_name_str
1720   }
1721   \str_if_empty:NT \l_stex_module_lang_str {
1722     \str_set:Nx \l_stex_module_lang_str {
1723       \prop_item:cn {c_stex_module\_l_stex_current_module_str _prop}
1724       { lang }
1725     }
1726   }
1727 }{
1728   % not nested:
1729   \str_if_empty:NT \l_stex_module_ns_str {
1730     \stex_modules_current_namespace:
1731     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1732     / {\l_stex_module_ns_str}
1733     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1734     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1735       \str_set:Nx \l_stex_module_ns_str {
1736         \stex_path_to_string:N \l_tmpa_seq
1737       }
1738     }
1739   }
1740 }

```

Next, we determine the language of the module:

```

1741 \str_if_empty:NT \l_stex_module_lang_str {
1742   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1743   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1744   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1745   \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
1746     \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
1747       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
1748     }
1749   }
1750   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1751   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
1752     \seq_pop_right:NN \l_tmpa_seq \l_stex_module_lang_str
1753     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1754       inferred~from~file~name}
1755   }
1756 }
1757
1758 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
1759   \exp_args:NNo \stex_set_language:Nn \l_tmpa_str \l_stex_module_lang_str
1760 }}

```

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

```

1761 \str_if_empty:NTF \l_stex_module_sig_str {
1762   \exp_args:Nnx \prop_gset_from_keyval:cn {
1763     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1764   } {
1765     name      = \l_stex_module_name_str ,
1766     ns        = \l_stex_module_ns_str ,
1767     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1768     lang      = \l_stex_module_lang_str ,
1769     sig       = \l_stex_module_sig_str ,
1770     deprecate = \l_stex_module_deprecate_str ,
1771     meta      = \l_stex_module_meta_str
1772   }
1773   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1774   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1775   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _copymodules}
1776   \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1777   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1778 \str_if_empty:NT \l_stex_module_meta_str {
1779   \str_set_eq:NN \l_stex_module_meta_str \l_stex_metatheory_str
1780 }
1781 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1782   \bool_set_true:N \l_stex_in_meta_bool
1783   \exp_args:Nx \stex_add_to_current_module:n {
1784     \bool_set_true:N \l_stex_in_meta_bool
1785     \stex_activate_module:n {\l_stex_module_meta_str}
1786     \bool_set_false:N \l_stex_in_meta_bool
1787   }
1788   \stex_activate_module:n {\l_stex_module_meta_str}
1789   \bool_set_false:N \l_stex_in_meta_bool
1790 }
1791 }{
1792   \str_if_empty:NT \l_stex_module_lang_str {
1793     \msg_error:nnxx{stex}{error/siglanguage}{
1794       \l_stex_module_ns_str?\l_stex_module_name_str
1795     }{\l_stex_module_sig_str}
1796   }
1797   \stex_debug:nn{modules}{Signature~\l_stex_module_sig_str~for~\l_stex_module_ns_str?\l_st
1798   \stex_if_module_exists:nTF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1799     \stex_debug:nn{modules}{(already exists)}
1800   }{
1801     \stex_debug:nn{modules}{(needs loading)}
1802     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1803     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1804     \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1805     \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1806     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1807     \str_set:Nx \l_tmpa_str {
1808       \stex_path_to_string:N \l_tmpa_seq /
1809       \l_tmpa_str . \l_stex_module_sig_str .tex
1810     }

```

```

1811 \IfFileExists \l_tmpa_str {
1812   \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1813     \str_clear:N \l_stex_current_module_str
1814     \seq_clear:N \l_stex_all_modules_seq
1815     \stex_debug:nn{modules}{Loading~signature}
1816   }
1817   }{
1818     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1819   }
1820 }
1821 \stex_if_smsmode:F {
1822   \stex_activate_module:n {
1823     \l_stex_module_ns_str ? \l_stex_module_name_str
1824   }
1825 }
1826 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1827 }
1828 \str_if_empty:NF \l_stex_module_deprecate_str {
1829   \msg_warning:nnxx{stex}{warning/deprecated}{
1830     Module~\l_stex_current_module_str
1831   }{
1832     \l_stex_module_deprecate_str
1833   }
1834 }
1835 \seq_put_right:Nx \l_stex_all_modules_seq {
1836   \l_stex_module_ns_str ? \l_stex_module_name_str
1837 }
1838 \tl_clear:c{l_stex_modules_aftergroup \l_stex_module_ns_str ? \l_stex_module_name_str _tl
1839 }

```

(End definition for `\stex_module_setup:nn`. This function is documented on page 85.)

`smodule (env.)` The module environment.

```

\__stex_modules_begin_module: implements \begin{smodule}
1840 \cs_new_protected:Nn \__stex_modules_begin_module: {
1841   \stex_reactivate_macro:N \STEXexport
1842   \stex_reactivate_macro:N \importmodule
1843   \stex_reactivate_macro:N \symdecl
1844   \stex_reactivate_macro:N \notation
1845   \stex_reactivate_macro:N \symdef
1846
1847   \stex_debug:nn{modules}{
1848     New~module:\\
1849     Namespace:~\l_stex_module_ns_str\\
1850     Name:~\l_stex_module_name_str\\
1851     Language:~\l_stex_module_lang_str\\
1852     Signature:~\l_stex_module_sig_str\\
1853     Metatheory:~\l_stex_module_meta_str\\
1854     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1855   }
1856
1857   \stex_if_do_html:T{
1858     \begin{stex_annotate_env} {theory} {

```

```

1859     \l_stex_module_ns_str ? \l_stex_module_name_str
1860   }
1861
1862   \stex_annotate_invisible:nnn{header}{} {
1863     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1864     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1865     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1866       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1867     }
1868     \str_if_empty:NF \smoduletype {
1869       \stex_annotate:nnn{type}{\smoduletype}{}
1870     }
1871   }
1872 }
1873 % TODO: Inherit metatheory for nested modules?
1874 }
1875 \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}

```

1876 \cs_new_protected:Nn \_stex_modules_end_module: {
1877   \stex_debug:nn{modules}{Closing~module~\prop_item:cn {c_stex_module_\l_stex_current_module_str}
1878   \_stex_reset_up_to_module:n \l_stex_current_module_str
1879   \stex_if_smsmode:T {
1880     \stex_persist:x {
1881       \prop_set_from_keyval:cn{c_stex_module_\l_stex_current_module_str _prop}{
1882         \exp_after:wN \prop_to_keyval:N \csname c_stex_module_\l_stex_current_module_str _pr
1883       }
1884       \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _constants}{
1885         \seq_use:cn{c_stex_module_\l_stex_current_module_str _constants},
1886       }
1887       \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _imports}{
1888         \seq_use:cn{c_stex_module_\l_stex_current_module_str _imports},
1889       }
1890       \tl_set:cn {c_stex_module_\l_stex_current_module_str _code}
1891     }
1892     \exp_after:wN \let \exp_after:wN \l_tmpa_tl \csname c_stex_module_\l_stex_current_module_str
1893     \exp_after:wN \stex_persist:n \exp_after:wN { \exp_after:wN { \l_tmpa_tl } }
1894   }
1895 }

```

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

The core environment

```

1896 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1897 \NewDocumentEnvironment { smodule } { 0{} m } {
1898   \stex_module_setup:nn{#1}{#2}
1899   %\par
1900   \stex_if_smsmode:F{
1901     \tl_if_empty:NF \smoduletitle {
1902       \exp_args:No \stex_document_title:n \smoduletitle
1903     }
1904     \tl_clear:N \l_tmpa_tl
1905     \clist_map_inline:Nn \smoduletype {

```

```

1906     \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1907         \tl_set:Nn \l_tmpa_tl {
1908             \stex_patch_counters:
1909             \use:c{__stex_modules_smodule_##1_start:}
1910             \stex_unpatch_counters:
1911         }
1912     }
1913 }
1914 \tl_if_empty:NTF \l_tmpa_tl {
1915     \__stex_modules_smodule_start:
1916 }{
1917     \l_tmpa_tl
1918 }
1919 }
1920 \__stex_modules_begin_module:
1921 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1922     \exp_args:Nx \stex_add_to_current_module:n{
1923         \stex_activate_module:n {\l_stex_module_meta_str}
1924     }
1925 }
1926 \str_if_empty:NF \smoduleid {
1927     \stex_ref_new_doc_target:n \smoduleid
1928 }
1929 \stex_smsmode_do:
1930 } {
1931     \__stex_modules_end_module:
1932     \stex_if_smsmode:F {
1933         \end{stex_annotate_env}
1934         \clist_set:Nn \l_tmpa_clist \smoduletype
1935         \tl_clear:N \l_tmpa_tl
1936         \clist_map_inline:Nn \l_tmpa_clist {
1937             \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1938                 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1939             }
1940         }
1941         \tl_if_empty:NTF \l_tmpa_tl {
1942             \__stex_modules_smodule_end:
1943         }{
1944             \l_tmpa_tl
1945         }
1946     }
1947 }

```

### **\stexpatchmodule**

```

1948 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1949 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1950
1951 \newcommand\stexpatchmodule[3] [] {
1952     \str_set:Nx \l_tmpa_str{ #1 }
1953     \str_if_empty:NTF \l_tmpa_str {
1954         \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1955         \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1956     }{
1957         \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }

```

```

1958     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1959   }
1960 }

```

(End definition for `\stexpatchmodule`. This function is documented on page 85.)

## 29.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n
1961 \NewDocumentCommand \STEXModule { m } {
1962   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1963   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1964   \tl_set:Nn \l_tmpa_tl {
1965     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1966   }
1967   \seq_map_inline:Nn \l_stex_all_modules_seq {
1968     \str_set:Nn \l_tmpb_str { ##1 }
1969     \str_if_eq:eeT { \l_tmpa_str } {
1970       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1971     } {
1972       \seq_map_break:n {
1973         \tl_set:Nn \l_tmpa_tl {
1974           \stex_invoke_module:n { ##1 }
1975         }
1976       }
1977     }
1978   }
1979   \l_tmpa_tl
1980 }
1981
1982 \cs_new_protected:Nn \stex_invoke_module:n {
1983   \stex_debug:nn{modules}{Invoking~module~#1}
1984   \peek_charcode_remove:NTF ! {
1985     \__stex_modules_invoke_uri:nN { #1 }
1986   } {
1987     \peek_charcode_remove:NTF ? {
1988       \__stex_modules_invoke_symbol:nn { #1 }
1989     } {
1990       \msg_error:nnx{stex}{error/syntax}{
1991         ?~or~!~expected~after~
1992         \c_backslash_str STEXModule{#1}
1993       }
1994     }
1995   }
1996 }
1997
1998 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1999   \str_set:Nn #2 { #1 }
2000 }
2001
2002 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
2003   \stex_invoke_symbol:n{#1?#2}
2004 }

```

(End definition for `\STEXModule` and `\stex_invoke_module:n`. These functions are documented on page 85.)

`\stex_activate_module:n`

```

2005 \bool_new:N \l_stex_in_meta_bool
2006 \bool_set_false:N \l_stex_in_meta_bool
2007 \cs_new_protected:Nn \stex_activate_module:n {
2008   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
2009     \stex_debug:nn{modules}{Activating~module~#1}
2010     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
2011     \use:c{ c_stex_module_#1_code }
2012   }
2013 }

```

(End definition for `\stex_activate_module:n`. This function is documented on page 86.)

`mmtinterface (env.)`

```

2014 \NewDocumentEnvironment { mmtinterface } { 0{} m m } {
2015   \stex_module_setup:nn{#1}{#3}
2016   %\par
2017   \stex_if_smsmode:F{
2018     \tl_if_empty:NF \smoduletitle {
2019       \exp_args:No \stex_document_title:n \smoduletitle
2020     }
2021     \tl_clear:N \l_tmpa_tl
2022     \clist_map_inline:Nn \smoduletype {
2023       \tl_if_exist:cT {__stex_modules_smodule_#1_start:}{
2024         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_#1_start:}}
2025       }
2026     }
2027     \tl_if_empty:NTF \l_tmpa_tl {
2028       \__stex_modules_smodule_start:
2029     }{
2030       \l_tmpa_tl
2031     }
2032   }
2033   \__stex_modules_begin_module:
2034   \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
2035     \exp_args:Nx \stex_add_to_current_module:n{
2036       \stex_activate_module:n {\l_stex_module_meta_str}
2037     }
2038   }
2039   \str_if_empty:NF \smoduleid {
2040     \stex_ref_new_doc_target:n \smoduleid
2041   }
2042   \str_set:Nx \l_stex_module_mmtfor_str {#2}
2043   \MMTinclude{#2}
2044   \stex_reactivate_macro:N \mmtdecl
2045   \stex_reactivate_macro:N \mmtdef
2046   \stex_smsmode_do:
2047 }{
2048   \__stex_modules_end_module:
2049   \stex_if_smsmode:F {
2050     \end{stex_annotate_env}

```

```

2051 \clist_set:No \l_tmpa_clist \smodulotype
2052 \tl_clear:N \l_tmpa_tl
2053 \clist_map_inline:Nn \l_tmpa_clist {
2054   \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
2055     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
2056   }
2057 }
2058 \tl_if_empty:NTF \l_tmpa_tl {
2059   __stex_modules_smodule_end:
2060 }{
2061   \l_tmpa_tl
2062 }
2063 }
2064 }
2065 </package>

```



## Chapter 30

# STEX -Module Inheritance Implementation

```
2066 <*package>
2067
2068 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
2069
```

### 30.1 SMS Mode

```
2070 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq
2071 \tl_new:N \g_stex_smsmode_allowedmacros_tl
2072 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
2073 \seq_new:N \g_stex_smsmode_allowedenvs_seq
2074
2075 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
2076   \makeatletter
2077   \makeatother
2078   \ExplSyntaxOn
2079   \ExplSyntaxOff
2080   \rustexBREAK
2081 }
2082
2083 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
2084   \symdef
2085   \importmodule
2086   \notation
2087   \symdecl
2088   \STEXexport
2089   \inlineass
2090   \inlinedef
2091   \inlineex
2092   \endinput
2093   \setnotation
```

```

2094 \copynotation
2095 \assign
2096 \renamedekl
2097 \donotcopy
2098 \instantiate
2099 \textsymdecl
2100 \mmtdef
2101 \setmetatheory
2102 }
2103
2104 \exp_args:Nx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
2105   \tl_to_str:n {
2106     smodule,
2107     copymodule,
2108     interpretmodule,
2109     realization,
2110     sdefinition,
2111     sexample,
2112     sassertion,
2113     sparagraph,
2114     mmtinterface,
2115     mathstructure,
2116     extstructure,
2117     extstructure*
2118   }
2119 }

```

(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 87.)

`\stex_if_smsmode_p:`  
`\stex_if_smsmode:TF`

```

2120 \bool_new:N \g__stex_smsmode_bool
2121 \bool_set_false:N \g__stex_smsmode_bool
2122 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
2123   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
2124 }

```

(End definition for `\stex_if_smsmode:TF`. This function is documented on page 87.)

`\__stex_smsmode_in_smsmode:nn`

```

2125 \cs_new_protected:Nn \__stex_smsmode_in_smsmode:nn { \stex_suppress_html:n {
2126   \vbox_set:Nn \l_tmpa_box {
2127     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
2128     \bool_gset_true:N \g__stex_smsmode_bool
2129     #2
2130     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
2131   }
2132   \box_clear:N \l_tmpa_box
2133 } }

```

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

`\stex_file_in_smsmode:nn`

```

2134 \quark_new:N \q__stex_smsmode_break
2135

```

```

2136 \NewDocumentCommand \__stex_smsmode_importmodule: { 0{} m} {
2137   \seq_gput_right:Nn \l__stex_smsmode_importmodules_seq {{#1}{#2}}
2138   \stex_smsmode_do:
2139 }
2140
2141 \cs_new_protected:Nn \__stex_smsmode_module:nn {
2142   \__stex_modules_args:n{#1}
2143   \stex_if_in_module:F {
2144     \str_if_empty:NF \l_stex_module_sig_str {
2145       \stex_modules_current_namespace:
2146       \str_set:Nx \l_stex_module_name_str { #2 }
2147       \stex_if_module_exists:nF{\l_stex_module_ns_str?\l_stex_module_name_str}{
2148         \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
2149         \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
2150         \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
2151         \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
2152         \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
2153         \str_set:Nx \l_tmpa_str {
2154           \stex_path_to_string:N \l_tmpa_seq /
2155           \l_tmpa_str . \l_stex_module_sig_str .tex
2156         }
2157         \IfFileExists \l_tmpa_str {
2158           \exp_args:NNx \seq_gput_right:Nn \l__stex_smsmode_sigmodules_seq \l_tmpa_str
2159         }{
2160           \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
2161         }
2162       }
2163     }
2164   }
2165 }
2166
2167 \prg_new_conditional:Nnn \__stex_smsmode_check_import_pair:nn {T,F,TF} {
2168   %\stex_debug:nn{import-pair}{\detokenize{{#1}~{#2}}}}
2169   \tl_if_empty:nTF{#1}{
2170     \prop_if_exist:NTF \l_stex_current_repository_prop
2171     {
2172       %\stex_debug:nn{import-pair}{in repository \prop_item:Nn \l_stex_current_repository_
2173       \prg_return_true:
2174     } {
2175       \seq_set_split:Nnn \l_tmpa_seq ? {#2}
2176       \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
2177       \tl_if_empty:NT \l_tmpa_tl {
2178         \seq_pop_left:NN \l_tmpa_seq \l_tmpa_tl
2179       }
2180       %\stex_debug:nn{import-pair}{\seq_use:Nn \l_tmpa_seq,~of~length~\seq_count:N \l_tmpa_
2181       \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1
2182       \prg_return_true: \prg_return_false:
2183     }
2184   }\prg_return_true:
2185 }
2186
2187 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
2188   \stex_filestack_push:n{#1}
2189   \seq_gclear:N \l__stex_smsmode_importmodules_seq

```

```

2190 \seq_gclear:N \l__stex_smsmode_sigmodules_seq
2191 % ----- new -----
2192 \__stex_smsmode_in_smsmode:nn{#1}{
2193   \let\importmodule\__stex_smsmode_importmodule:
2194   \let\stex_module_setup:nn\__stex_smsmode_module:nn
2195   \let\__stex_modules_begin_module:\relax
2196   \let\__stex_modules_end_module:\relax
2197   \seq_clear:N \g_stex_smsmode_allowedenvs_seq
2198   \exp_args:NNx \seq_put_right:Nn \g_stex_smsmode_allowedenvs_seq {\tl_to_str:n{smodule}}
2199   \tl_clear:N \g_stex_smsmode_allowedmacros_tl
2200   \tl_clear:N \g_stex_smsmode_allowedmacros_escape_tl
2201   \tl_put_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\importmodule}
2202   \everyeof{\q__stex_smsmode_break\noexpand}
2203   \expandafter\expandafter\expandafter
2204   \stex_smsmode_do:
2205   \csname @ @ input\endcsname "#1"\relax
2206
2207   \seq_map_inline:Nn \l__stex_smsmode_sigmodules_seq {
2208     \stex_filestack_push:n{##1}
2209     \expandafter\expandafter\expandafter
2210     \stex_smsmode_do:
2211     \csname @ @ input\endcsname "##1"\relax
2212     \stex_filestack_pop:
2213   }
2214 }
2215 % ----- new -----
2216 \__stex_smsmode_in_smsmode:nn{#1} {
2217   #2
2218   % ----- new -----
2219   \begingroup
2220   %\stex_debug:nn{smsmode}{Here:~\seq_use:Nn\l__stex_smsmode_importmodules_seq, }
2221   \seq_map_inline:Nn \l__stex_smsmode_importmodules_seq {
2222     \__stex_smsmode_check_import_pair:nnT ##1 { \begingroup
2223       \stex_import_module_uri:nn ##1
2224       \stex_import_require_module:nnnn
2225       \l_stex_import_ns_str
2226       \l_stex_import_archive_str
2227       \l_stex_import_path_str
2228       \l_stex_import_name_str \endgroup
2229     }
2230   }
2231   \endgroup
2232   \stex_debug:nn{smsmode}{Actually~loading~file~#1}
2233   % ----- new -----
2234   \everyeof{\q__stex_smsmode_break\noexpand}
2235   \expandafter\expandafter\expandafter
2236   \stex_smsmode_do:
2237   \csname @ @ input\endcsname "#1"\relax
2238 }
2239 \stex_filestack_pop:
2240 }

```

(End definition for `\stex_file_in_smsmode:nn`. This function is documented on page 88.)

`\stex_smsmode_do:` is executed on encountering `\` in smsmode. It checks whether the corresponding command

is allowed and executes or ignores it accordingly:

```

2241 \cs_new_protected:Npn \stex_smsmode_do: {
2242   \stex_if_smsmode:T {
2243     \__stex_smsmode_do:w
2244   }
2245 }
2246 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
2247   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
2248     \expandafter\if\expandafter\relax\noexpand#1
2249     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
2250   } \else\expandafter\__stex_smsmode_do:w\fi
2251 }{
2252   \__stex_smsmode_do:w % #1
2253 }
2254 }
2255 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
2256   \cs_if_eq:NNF #1 \q__stex_smsmode_break {
2257     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
2258       #1\__stex_smsmode_do:w
2259     }{
2260       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
2261         #1
2262       }{
2263         \cs_if_eq:NNTF \begin #1 {
2264           \__stex_smsmode_check_begin:n
2265         }{
2266           \cs_if_eq:NNTF \end #1 {
2267             \__stex_smsmode_check_end:n
2268           }{
2269             \__stex_smsmode_do:w
2270           }
2271         }
2272       }
2273     }
2274   }
2275 }
2276
2277 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
2278   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
2279     \begin{#1}
2280   }{
2281     \__stex_smsmode_do:w
2282   }
2283 }
2284 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
2285   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
2286     \end{#1}\__stex_smsmode_do:w
2287   }{
2288     \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
2289   }
2290 }

```

(End definition for `\stex_smsmode_do:`. This function is documented on page 88.)

## 30.2 Inheritance

2291 <@@=stex\_importmodule>

\stex\_import\_module\_uri:nn

```

2292 \cs_new_protected:Nn \stex_import_module_uri:nn {
2293   \str_set:Nx \l_stex_import_archive_str { #1 }
2294   \str_set:Nn \l_stex_import_path_str { #2 }
2295
2296   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
2297   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
2298   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
2299
2300   \stex_modules_current_namespace:
2301   \bool_lazy_all:nTF {
2302     {\str_if_empty_p:N \l_stex_import_archive_str}
2303     {\str_if_empty_p:N \l_stex_import_path_str}
2304     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
2305   }{
2306     \str_set_eq:NN \l_stex_import_path_str \l_stex_module_subpath_str
2307     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
2308   }{
2309     \str_if_empty:NT \l_stex_import_archive_str {
2310       \prop_if_exist:NT \l_stex_current_repository_prop {
2311         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
2312       }
2313     }
2314     \str_if_empty:NTF \l_stex_import_archive_str {
2315       \str_if_empty:NF \l_stex_import_path_str {
2316         \stex_path_from_string:Nn \l_tmpb_seq {
2317           \l_stex_module_ns_str / .. / \l_stex_import_path_str
2318         }
2319         \str_set:Nx \l_stex_import_ns_str {\stex_path_to_string:N \l_tmpb_seq}
2320         \str_replace_once:Nnn \l_stex_import_ns_str {file://} {file://}
2321       }
2322     }{
2323       \stex_require_repository:n \l_stex_import_archive_str
2324       \prop_get:cnN { c_stex_mathhub_\l_stex_import_archive_str _manifest_prop } { ns }
2325       \l_stex_import_ns_str
2326       \str_if_empty:NF \l_stex_import_path_str {
2327         \str_set:Nx \l_stex_import_ns_str {
2328           \l_stex_import_ns_str / \l_stex_import_path_str
2329         }
2330       }
2331     }
2332   }
2333 }
```

(End definition for \stex\_import\_module\_uri:nn. This function is documented on page 89.)

\l\_stex\_import\_name\_str  
\l\_stex\_import\_archive\_str  
\l\_stex\_import\_path\_str  
\l\_stex\_import\_ns\_str

Store the return values of \stex\_import\_module\_uri:nn.

```

2334 \str_new:N \l_stex_import_name_str
2335 \str_new:N \l_stex_import_archive_str
2336 \str_new:N \l_stex_import_path_str
2337 \str_new:N \l_stex_import_ns_str
```

(End definition for `\l_stex_import_name_str` and others. These variables are documented on page 89.)

```

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

2338 \cs_new_protected:Nn \stex_import_require_module:nnnn {
2339   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
2340
2341     \stex_debug:nn{requiremodule}{Here:\!\!\sim1:\sim1\!\!\sim2:\sim2\!\!\sim3:\sim3\!\!\sim4:\sim4}
2342
2343     \exp_args:NNxx \seq_set_split:Nnn \l_tmpa_seq {\tl_to_str:n{/}} {#4}
2344     \seq_get_left:NN \l_tmpa_seq \l_tmpc_str
2345
2346     %\stex_debug:nn{requiremodule}{Top-module:\l_tmpc_str}
2347
2348     % archive
2349     \str_set:Nx \l_tmpa_str { #2 }
2350     \str_if_empty:NTF \l_tmpa_str {
2351       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
2352       \seq_put_right:Nn \l_tmpa_seq {...}
2353     } {
2354       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
2355       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
2356       \seq_put_right:Nn \l_tmpa_seq { source }
2357     }
2358
2359     % path
2360     \str_set:Nx \l_tmpb_str { #3 }
2361     \str_if_empty:NTF \l_tmpb_str {
2362       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / \l_tmpc_str }
2363     }
2364     \ltx@ifpackageloaded{babel} {
2365       \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
2366         { \language_name } \l_tmpb_str {
2367         \msg_error:nnx{stex}{error/unknownlanguage}{\language_name}
2368       }
2369     } {
2370       \str_clear:N \l_tmpb_str
2371     }
2372
2373     \stex_debug:nn{modules}{Checking-a1~\l_tmpa_str.\l_tmpb_str.tex}
2374     \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2375       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2376     }{
2377       \stex_debug:nn{modules}{Checking-a2~\l_tmpa_str.tex}
2378       \IfFileExists{ \l_tmpa_str.tex }{
2379         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2380       }{
2381         % try english as default
2382         \stex_debug:nn{modules}{Checking-a3~\l_tmpa_str.en.tex}
2383         \IfFileExists{ \l_tmpa_str.en.tex }{
2384           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2385         }{
2386           \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2387         }
2388       }

```

```

2389     }
2390
2391   } {
2392     \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
2393     \seq_concat:NNN \l_tmpb_seq \l_tmpa_seq \l_tmpb_seq
2394
2395     \ltx@ifpackageloaded{babel} {
2396       \exp_args:Nn \prop_get:NnNF \c_stex_language_abbrevs_prop
2397         { \language } \l_tmpb_str {
2398         \msg_error:nnx{stex}{error/unknownlanguage}{\language}
2399       }
2400     } {
2401       \str_clear:N \l_tmpb_str
2402     }
2403
2404     \stex_path_canonicalize:N \l_tmpb_seq
2405     \stex_path_to_string:NN \l_tmpb_seq \l_tmpa_str
2406
2407     \stex_debug:nn{modules}{Checking~b1~\l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex}
2408     \IfFileExists{ \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }{
2409       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }
2410     }{
2411       \stex_debug:nn{modules}{Checking~b2~\l_tmpa_str/\l_tmpc_str.tex}
2412       \IfFileExists{ \l_tmpa_str/\l_tmpc_str.tex }{
2413         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.tex }
2414       }{
2415         % try english as default
2416         \stex_debug:nn{modules}{Checking~b3~\l_tmpa_str/\l_tmpc_str.en.tex}
2417         \IfFileExists{ \l_tmpa_str/\l_tmpc_str.en.tex }{
2418           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.en.tex }
2419         }{
2420           \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.\l_tmpb_str.tex}
2421           \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2422             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2423           }{
2424             \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.tex}
2425             \IfFileExists{ \l_tmpa_str.tex }{
2426               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2427             }{
2428               % try english as default
2429               \stex_debug:nn{modules}{Checking~b5~\l_tmpa_str.en.tex}
2430               \IfFileExists{ \l_tmpa_str.en.tex }{
2431                 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2432               }{
2433                 \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2434               }
2435             }
2436           }
2437         }
2438       }
2439     }
2440   }
2441
2442   \str_if_eq:eeF{\g__stex_importmodule_file_str}{\seq_use:Nn \g_stex_currentfile_seq /}{

```



```

2443 \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
2444 \seq_clear:N \l_stex_all_modules_seq
2445 \str_clear:N \l_stex_current_module_str
2446 \str_set:Nx \l_tmpb_str { #2 }
2447 \str_if_empty:NF \l_tmpb_str {
2448 \stex_set_current_repository:n { #2 }
2449 }
2450 \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
2451 }
2452
2453 \stex_if_module_exists:nF { #1 ? #4 } {
2454 \msg_error:nnx{stex}{error/unknownmodule}{
2455 #1?#4~(in~file~\g__stex_importmodule_file_str)
2456 }
2457 }
2458 }
2459
2460 }
2461 \stex_activate_module:n { #1 ? #4 }
2462 }

```

(End definition for `\stex_import_require_module:nnnn`. This function is documented on page 89.)

#### `\importmodule`

```

2463 \NewDocumentCommand \importmodule { 0{} m } {
2464 \stex_import_module_uri:nn { #1 } { #2 }
2465 \stex_debug:nn{modules}{Importing~module:~
2466 \l_stex_import_ns_str ? \l_stex_import_name_str
2467 }
2468 \stex_if_smsmode:F {
2469 \stex_annotate_invisible:nnn
2470 {import} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
2471 }
2472 \stex_execute_in_module:x {
2473 \stex_import_require_module:nnnn
2474 { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2475 { \l_stex_import_path_str } { \l_stex_import_name_str }
2476 }
2477 \exp_args:Nx \stex_add_import_to_current_module:n {
2478 \l_stex_import_ns_str ? \l_stex_import_name_str
2479 }
2480 \stex_smsmode_do:
2481 \ignorespacesandpars
2482 }
2483 \stex_deactivate_macro:Nn \importmodule {module-environments}

```

(End definition for `\importmodule`. This function is documented on page 88.)

#### `\usemodule`

```

2484 \NewDocumentCommand \usemodule { 0{} m } {
2485 \stex_if_smsmode:F {
2486 \stex_import_module_uri:nn { #1 } { #2 }
2487 \stex_import_require_module:nnnn
2488 { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2489 { \l_stex_import_path_str } { \l_stex_import_name_str }

```

```

2490     \stex_annotate_invisible:nnn
2491     {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
2492   }
2493   \stex_smsmode_do:
2494   \ignorespacesandpars
2495 }

(End definition for \usemodule. This function is documented on page 88.)

2496 \cs_new_protected:Nn \stex_csl_to_imports:Nn {
2497   \tl_if_empty:nF{#2}{
2498     \clist_set:Nn \l_tmpa_clist {#2}
2499     \clist_map_inline:Nn \l_tmpa_clist {
2500       \tl_if_head_eq_charcode:nNTF {##1}[{
2501         #1 ##1
2502       }{
2503         #1{##1}
2504       }
2505     }
2506   }
2507 }
2508 \cs_generate_variant:Nn \stex_csl_to_imports:Nn {No}
2509
2510
2511 \endpackage

```

## Chapter 31

# STEX -Symbols Implementation

```
2512 <*package>
2513
2514 %%%%%%%%%% symbols.dtx %%%%%%%%%%
2515
    Warnings and error messages
2516 \msg_new:nnn{stex}{error/wrongargs}{
2517   args~value~in~symbol~declaration~for~#1~
2518   needs~to~be~i,~a,~b~or~B,~but~#2~given
2519 }
2520 \msg_new:nnn{stex}{error/unknownsymbol}{
2521   No~symbol~#1~found!
2522 }
2523 \msg_new:nnn{stex}{error/seqlength}{
2524   Expected~#1~arguments;~got~#2!
2525 }
2526 \msg_new:nnn{stex}{error/unknownnotation}{
2527   Unknown~notation~#1~for~#2!
2528 }
```

### 31.1 Symbol Declarations

```
2529 <@@=stex_symdecl>
\stex_all_symbols:n Map over all available symbols
2530 \cs_new_protected:Nn \stex_all_symbols:n {
2531   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
2532   \seq_map_inline:Nn \l_stex_all_modules_seq {
2533     \seq_map_inline:cn{c_stex_module_##1_constants}{
2534       \__stex_symdecl_all_symbols_cs{##1?###1}
2535     }
2536   }
2537 }
```

(End definition for `\stex_all_symbols:n`. This function is documented on page 91.)

## `\STEXsymbol`

```
2538 \NewDocumentCommand \STEXsymbol { m } {
2539   \stex_get_symbol:n { #1 }
2540   \exp_args:No
2541   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
2542 }
```

(End definition for `\STEXsymbol`. This function is documented on page 92.)

`symdecl` arguments:

```
2543 \keys_define:nn { stex / symdecl } {
2544   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2545   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2546   type      .tl_set:N   = \l_stex_symdecl_type_tl ,
2547   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,
2548   align     .str_set:N   = \l_stex_symdecl_align_str , % TODO(?)
2549   gfc       .str_set:N   = \l_stex_symdecl_gfc_str , % TODO(?)
2550   def       .tl_set:N   = \l_stex_symdecl_definiens_tl ,
2551   reorder   .str_set_x:N = \l_stex_symdecl_reorder_str ,
2552   argnames  .clist_set:N = \l_stex_symdecl_argnames_clist ,
2553   assoc     .choices:nn =
2554             {bin,binl,binr,pre,conj,pwconj}
2555             {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}
2556 }
2557
2558 \bool_new:N \l_stex_symdecl_make_macro_bool
2559
2560 \cs_new_protected:Nn \__stex_symdecl_args:n {
2561   \str_clear:N \l_stex_symdecl_name_str
2562   \str_clear:N \l_stex_symdecl_args_str
2563   \str_clear:N \l_stex_symdecl_deprecate_str
2564   \str_clear:N \l_stex_symdecl_reorder_str
2565   \str_clear:N \l_stex_symdecl_assoctype_str
2566   \bool_set_false:N \l_stex_symdecl_local_bool
2567   \tl_clear:N \l_stex_symdecl_type_tl
2568   \tl_clear:N \l_stex_symdecl_definiens_tl
2569   \clist_clear:N \l_stex_symdecl_argnames_clist
2570
2571   \keys_set:nn { stex / symdecl } { #1 }
2572 }
```

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

```
2573
2574 \NewDocumentCommand \symdecl { s m O{} } {
2575   \__stex_symdecl_args:n { #3 }
2576   \IfBooleanTF #1 {
2577     \bool_set_false:N \l_stex_symdecl_make_macro_bool
2578   } {
2579     \bool_set_true:N \l_stex_symdecl_make_macro_bool
2580   }
2581   \stex_symdecl_do:n { #2 }
2582   \stex_smsmode_do:
2583 }
```

```

2584
2585 \cs_new_protected:Nn \stex_symdecl_do:nn {
2586   \__stex_symdecl_args:n{#1}
2587   \bool_set_false:N \l_stex_symdecl_make_macro_bool
2588   \stex_symdecl_do:n{#2}
2589 }
2590
2591 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

(End definition for \symdecl. This function is documented on page 90.)

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

```

2592 \cs_new_protected:Nn \stex_symdecl_do:n {
2593   \stex_if_in_module:F {
2594     % TODO throw error? some default namespace?
2595   }
2596
2597   \str_if_empty:NT \l_stex_symdecl_name_str {
2598     \str_set:Nx \l_stex_symdecl_name_str { #1 }
2599   }
2600
2601   \prop_if_exist:cT { l_stex_symdecl_
2602     \l_stex_current_module_str ?
2603     \l_stex_symdecl_name_str
2604     _prop
2605   }{
2606     % TODO throw error (beware of circular dependencies)
2607   }
2608
2609   \prop_clear:N \l_tmpa_prop
2610   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
2611   \seq_clear:N \l_tmpa_seq
2612   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
2613   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
2614
2615   \str_if_empty:NT \l_stex_symdecl_deprecate_str {
2616     \str_if_empty:NF \l_stex_module_deprecate_str {
2617       \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
2618     }
2619   }
2620   \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
2621
2622   \exp_args:No \stex_add_constant_to_current_module:n {
2623     \l_stex_symdecl_name_str
2624   }
2625
2626   % arity/args
2627   \int_zero:N \l_tmpb_int
2628
2629   \bool_set_true:N \l_tmpa_bool
2630   \str_map_inline:Nn \l_stex_symdecl_args_str {
2631     \token_case_meaning:NnF ##1 {
2632       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2633       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }

```

```

2634     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2635     {\tl_to_str:n a} {
2636         \bool_set_false:N \l_tmpa_bool
2637         \int_incr:N \l_tmpb_int
2638     }
2639     {\tl_to_str:n B} {
2640         \bool_set_false:N \l_tmpa_bool
2641         \int_incr:N \l_tmpb_int
2642     }
2643 }{
2644     \msg_error:nnxx{stex}{error/wrongargs}{
2645         \l_stex_current_module_str ?
2646         \l_stex_symdecl_name_str
2647     }{##1}
2648 }
2649 }
2650
2651 \bool_if:NTF \l_tmpa_bool {
2652     % possibly numeric
2653     \str_if_empty:NTF \l_stex_symdecl_args_str {
2654         \prop_put:Nnn \l_tmpa_prop { args } {}
2655         \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2656     }{
2657         \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
2658         \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2659         \str_clear:N \l_tmpa_str
2660         \int_step_inline:nn \l_tmpa_int {
2661             \str_put_right:Nn \l_tmpa_str i
2662         }
2663         \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
2664     }
2665 } {
2666     \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2667     \prop_put:Nnx \l_tmpa_prop { arity }
2668     { \str_count:N \l_stex_symdecl_args_str }
2669 }
2670 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2671
2672 \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
2673     \prop_put:Nnx \l_tmpa_prop { defined }{ false }
2674 }{
2675     \prop_put:Nnx \l_tmpa_prop { defined }{ true }
2676 }
2677
2678 % argnames
2679
2680 \clist_clear:N \l_tmpa_clist
2681 \int_step_inline:nn {\prop_item:Nn \l_tmpa_prop {arity}} {
2682     \clist_if_empty:NTF \l_stex_symdecl_argnames_clist {
2683         \clist_put_right:Nn \l_tmpa_clist {##1}
2684     }{
2685         \clist_pop:NN \l_stex_symdecl_argnames_clist \l_tmpa_tl
2686         \exp_args:NNx \clist_put_right:Nn \l_tmpa_clist {\c_dollar_str\l_tmpa_tl}
2687     }

```

```

2688 }
2689 \prop_put:Nn \l_tmpa_prop {argnames} {\clist_use:Nn \l_tmpa_clist ,}
2690
2691 % semantic macro
2692
2693 \bool_if:NT \l_stex_symdecl_make_macro_bool {
2694   \exp_args:Nx \stex_do_up_to_module:n {
2695     \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2696       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2697     }}
2698   }
2699 }
2700
2701 \stex_debug:nn{symbols}{New~symbol:~
2702   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2703   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2704   Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
2705   Definiens:~\exp_not:o {\l_stex_symdecl_definiens_tl}
2706 }
2707
2708 % circular dependencies require this:
2709 \stex_if_do_html:T {
2710   \stex_annotate_invisible:nnn {symdecl} {
2711     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2712   } {
2713     \tl_if_empty:NF \l_stex_symdecl_type_tl {
2714       \stex_annotate_invisible:nnn{type}{\l_stex_symdecl_type_tl$}
2715     }
2716     \stex_annotate_invisible:nnn{args}{\prop_item:Nn \l_tmpa_prop { args }}{}
2717     \stex_annotate_invisible:nnn{macroname}{#1}{}
2718     \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2719       \stex_annotate_invisible:nnn{definiens}{}
2720       {\l_stex_symdecl_definiens_tl$}
2721     }
2722     \str_if_empty:NF \l_stex_symdecl_assoc_type_str {
2723       \stex_annotate_invisible:nnn{assoc_type}{\l_stex_symdecl_assoc_type_str}{}
2724     }
2725     \str_if_empty:NF \l_stex_symdecl_reorder_str {
2726       \stex_annotate_invisible:nnn{reorderargs}{\l_stex_symdecl_reorder_str}{}
2727     }
2728   }
2729 }
2730 \prop_if_exist:cF {
2731   \l_stex_symdecl_
2732   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2733   _prop
2734 } {
2735   \bool_if:NTF \l_stex_symdecl_local_bool \stex_do_up_to_module:x \stex_execute_in_module:
2736     \__stex_symdecl_restore_symbol:nnnnnnnn
2737       {\l_stex_symdecl_name_str}
2738       { \prop_item:Nn \l_tmpa_prop {args} }
2739       { \prop_item:Nn \l_tmpa_prop {arity} }
2740       { \prop_item:Nn \l_tmpa_prop {assoc} }
2741       { \prop_item:Nn \l_tmpa_prop {defined} }

```

```

2742         {\bool_if:NT \l_stex_symdecl_make_macro_bool {#1} }
2743         {\l_stex_current_module_str}
2744         { \prop_item:Nn \l_tmpa_prop {argnames} }
2745     }
2746 }
2747 }
2748 \cs_new_protected:Nn \__stex_symdecl_restore_symbol:nnnnnnnn {
2749   \prop_clear:N \l_tmpa_prop
2750   \prop_put:Nnn \l_tmpa_prop { module } { #7 }
2751   \prop_put:Nnn \l_tmpa_prop { name } { #1 }
2752   \prop_put:Nnn \l_tmpa_prop { args } { #2 }
2753   \prop_put:Nnn \l_tmpa_prop { arity } { #3 }
2754   \prop_put:Nnn \l_tmpa_prop { assocs } { #4 }
2755   \prop_put:Nnn \l_tmpa_prop { defined } { #5 }
2756   \prop_put:Nnn \l_tmpa_prop { argnames } { #8 }
2757   \tl_if_empty:nF{#6}{
2758     \tl_set:cx{#6}{\stex_invoke_symbol:n{\detokenize{#7 ? #1}}}
2759   }
2760   \prop_set_eq:cN{\l_stex_symdecl_ \detokenize{#7 ? #1} _prop}\l_tmpa_prop
2761   \seq_clear:c{\l_stex_symdecl_ \detokenize{#7 ? #1} _notations}
2762 }

```

(End definition for `\stex_symdecl_do:n`. This function is documented on page 91.)

## `\textsymdecl`

```

2763
2764 \keys_define:nn { stex / textsymdecl } {
2765   name      .str_set_x:N = \l__stex_symdecl_name_str ,
2766   type      .tl_set:N    = \l__stex_symdecl_type_tl
2767 }
2768
2769 \cs_new_protected:Nn \stex_textsymdecl_args:n {
2770   \str_clear:N \l__stex_symdecl_name_str
2771   \tl_clear:N \l__stex_symdecl_type_tl
2772   \clist_clear:N \l_stex_symdecl_argnames_clist
2773   \keys_set:nn { stex / textsymdecl } { #1 }
2774 }
2775
2776 \NewDocumentCommand \textsymdecl {m O{} m} {
2777   \stex_textsymdecl_args:n { #2 }
2778   \str_if_empty:NTF \l__stex_symdecl_name_str {
2779     \__stex_symdecl_args:n{name=#1,#2}
2780   }{
2781     \__stex_symdecl_args:n{#2}
2782   }
2783   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2784   \stex_symdecl_do:n{#1-sym}
2785   \stex_execute_in_module:n{
2786     \cs_set_nopar:cpn{#1name}{
2787       \ifvmode\hbox_unpack:N\c_empty_box\fi
2788       \ifmmode\hbox{#3}\else#3\fi\hspace
2789     }
2790     \cs_set_nopar:cpn{#1}{
2791       \ifmmode\csname#1-sym\expandafter\endcsname\else

```



```

2792     \ifvmode\hbox_unpack:N\c_empty_box\fi
2793     \symref{#1-sym}{#3}\expandafter\xspace
2794     \fi
2795   }
2796 }
2797 \stex_execute_in_module:x{
2798   \__stex_notation_restore_notation:nnnnn
2799   {\l_stex_current_module_str?\tl_if_empty:NTF\l__stex_symdecl_name_str{#1}\l__stex_symdecl_name_str{#1}}{0}
2800   {\exp_not:n{\STEXInternalTermMathOMSiiii{\STEXInternalCurrentSymbolStr}{}}{\neginfprec}{\neginfprec}}{0}
2801   \comp{\hbox{#3}}\STEXInternalSymbolAfterInvokationTL
2802   }}}
2803   {}
2804 }
2805 }
2806 \stex_smsmode_do:
2807 }

```

(End definition for `\textsymdecl`. This function is documented on page 23.)

`\stex_get_symbol:n`

```

2808 \str_new:N \l_stex_get_symbol_uri_str
2809
2810 \cs_new_protected:Nn \stex_get_symbol:n {
2811   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2812     \tl_set:Nn \l_tmpa_tl { #1 }
2813     \__stex_symdecl_get_symbol_from_cs:
2814   }{
2815     % argument is a string
2816     % is it a command name?
2817     \cs_if_exist:cTF { #1 }{
2818       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2819       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2820       \str_if_empty:NTF \l_tmpa_str {
2821         \exp_args:Nx \cs_if_eq:NNTF {
2822           \tl_head:N \l_tmpa_tl
2823         } \stex_invoke_symbol:n {
2824           \__stex_symdecl_get_symbol_from_cs:
2825         }{
2826           \__stex_symdecl_get_symbol_from_string:n { #1 }
2827         }
2828       } {
2829         \__stex_symdecl_get_symbol_from_string:n { #1 }
2830       }
2831     }{
2832       % argument is not a command name
2833       \__stex_symdecl_get_symbol_from_string:n { #1 }
2834       % \l_stex_all_symbols_seq
2835     }
2836   }
2837   \str_if_eq:eeF {
2838     \prop_item:cn {
2839       \l_stex_symdecl\l_stex_get_symbol_uri_str _prop
2840     }{ deprecate }
2841   }{}{

```

```

2842     \msg_warning:nnxx{stex}{warning/deprecated}{
2843     Symbol~\l_stex_get_symbol_uri_str
2844     }{
2845     \prop_item:cn {l_stex_symdecl_l\l_stex_get_symbol_uri_str _prop}{ deprecate }
2846     }
2847 }
2848 }
2849
2850 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2851     \tl_set:Nn \l_tmpa_tl {
2852         \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2853     }
2854     \str_set:Nn \l_tmpa_str { #1 }
2855
2856     %\int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2857
2858     \str_if_in:NnTF \l_tmpa_str ? {
2859         \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
2860         \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
2861         \str_set:Nx \l_tmpb_str {\seq_use:Nn \l_tmpa_seq ?}
2862     }{
2863         \str_clear:N \l_tmpb_str
2864     }
2865     \str_if_empty:NnTF \l_tmpb_str {
2866         \seq_map_inline:Nn \l_stex_all_modules_seq {
2867             \seq_map_inline:cn{c_stex_module_##1_constants}{
2868                 \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2869                     \seq_map_break:n{\seq_map_break:n{
2870                         \tl_set:Nn \l_tmpa_tl {
2871                             \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2872                         }
2873                     }}
2874                 }
2875             }
2876         }
2877     }{
2878         \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpb_str }
2879         \seq_map_inline:Nn \l_stex_all_modules_seq {
2880             \str_if_eq:eeT{ \l_tmpb_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2881                 \seq_map_inline:cn{c_stex_module_##1_constants}{
2882                     \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2883                         \seq_map_break:n{\seq_map_break:n{
2884                             \tl_set:Nn \l_tmpa_tl {
2885                                 \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2886                             }
2887                         }}
2888                     }
2889                 }
2890             }
2891         }
2892     }
2893
2894     \l_tmpa_tl
2895 }

```

```

2896
2897 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
2898   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2899     { \tl_tail:N \l_tmpa_tl }
2900   \tl_if_single:NTF \l_tmpa_tl {
2901     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2902       \exp_after:wN \str_set:Nn \exp_after:wN
2903         \l_stex_get_symbol_uri_str \l_tmpa_tl
2904     }{
2905       % TODO
2906       % tail is not a single group
2907     }
2908   }{
2909     % TODO
2910     % tail is not a single group
2911   }
2912 }

```

(End definition for `\stex_get_symbol:n`. This function is documented on page 91.)

## 31.2 Notations

```

2913 <@@=stex_notation>
2914 notation arguments:
2915 \keys_define:nn { stex / notation } {
2916   % lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2917   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2918   prec     .str_set_x:N = \l__stex_notation_prec_str ,
2919   op       .tl_set:N = \l__stex_notation_op_tl ,
2920   primary .bool_set:N = \l__stex_notation_primary_bool ,
2921   primary .default:n = {true} ,
2922   hints    .str_set_x:N = \l__stex_notation_hints_str,
2923   unknown .code:n = \str_set:Nx
2924     \l__stex_notation_variant_str \l_keys_key_str
2925 }
2926
2927 \cs_new_protected:Nn \_stex_notation_args:n {
2928   % \str_clear:N \l__stex_notation_lang_str
2929   \str_clear:N \l__stex_notation_variant_str
2930   \str_clear:N \l__stex_notation_prec_str
2931   \str_clear:N \l__stex_notation_hints_str
2932   \tl_clear:N \l__stex_notation_op_tl
2933   \bool_set_false:N \l__stex_notation_primary_bool
2934
2935   \keys_set:nn { stex / notation } { #1 }
2936 }

```

**\notation**

```

2936 \NewDocumentCommand \notation { s m O{}} {
2937   \_stex_notation_args:n { #3 }
2938   \tl_clear:N \l_stex_symdecl_definiens_tl
2939   \stex_get_symbol:n { #2 }
2940   \tl_set:Nn \l_stex_notation_after_do_tl {

```

```

2941 \__stex_notation_final:
2942 \IfBooleanTF#1{
2943   \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2944 }{}
2945 \stex_smsmode_do:\ignorespacesandpars
2946 }
2947 \stex_notation_do:nnnnn
2948 { \prop_item:cn {\l_stex_symdecl\l_stex_get_symbol_uri_str _prop } { args } }
2949 { \prop_item:cn { \l_stex_symdecl\l_stex_get_symbol_uri_str _prop } { arity } }
2950 { \l__stex_notation_variant_str }
2951 { \l__stex_notation_prec_str }
2952 }
2953 \stex_deactivate_macro:Nn \notation {module-environments}

```

(End definition for \notation. This function is documented on page 91.)

\stex\_notation\_do:nnnnn

```

2954 \seq_new:N \l__stex_notation_precedences_seq
2955 \tl_new:N \l__stex_notation_opprec_tl
2956 \int_new:N \l__stex_notation_currarg_int
2957 \tl_new:N \STEXInternalSymbolAfterInvokationTL
2958
2959 \cs_new_protected:Nn \stex_notation_do:nnnnn {
2960   \let\STEXInternalCurrentSymbolStr\relax
2961   \seq_clear:N \l__stex_notation_precedences_seq
2962   \tl_clear:N \l__stex_notation_opprec_tl
2963   \str_set:Nx \l__stex_notation_args_str { #1 }
2964   \str_set:Nx \l__stex_notation_arity_str { #2 }
2965   \str_set:Nx \l__stex_notation_suffix_str { #3 }
2966   \str_set:Nx \l__stex_notation_prec_str { #4 }
2967
2968   % precedences
2969   \str_if_empty:NTF \l__stex_notation_prec_str {
2970     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2971       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2972     }{
2973       \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2974     }
2975   } {
2976     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2977       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2978       \int_step_inline:nn { \l__stex_notation_arity_str } {
2979         \exp_args:NNo
2980         \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2981       }
2982     }{
2983       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2984       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2985         \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2986         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2987           \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2988             \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2989           \seq_map_inline:Nn \l_tmpa_seq {
2990             \seq_put_right:Nn \l__stex_notation_precedences_seq { ##1 }

```

```

2991     }
2992   }
2993   }{
2994     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2995       \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2996     }{
2997       \tl_set:No \l__stex_notation_opprec_tl { 0 }
2998     }
2999   }
3000 }
3001 }
3002
3003 \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
3004 \int_step_inline:nn { \l__stex_notation_arity_str } {
3005   \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
3006     \exp_args:NNo
3007     \seq_put_right:No \l__stex_notation_precedences_seq {
3008       \l__stex_notation_opprec_tl
3009     }
3010   }
3011 }
3012 \tl_clear:N \l_stex_notation_dummyargs_tl
3013
3014 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
3015   \exp_args:NNe
3016   \cs_set:Npn \l_stex_notation_macrocode_cs {
3017     \STEXInternalTermMathOMSiiii { \STEXInternalCurrentSymbolStr }
3018     { \l__stex_notation_suffix_str }
3019     { \l__stex_notation_opprec_tl }
3020     { \exp_not:n { #5 } }
3021   }
3022   \l_stex_notation_after_do_tl
3023 }{
3024   \str_if_in:NnTF \l__stex_notation_args_str b {
3025     \exp_args:Nne \use:nn
3026     {
3027       \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
3028       \cs_set:Npn \l__stex_notation_arity_str } { {
3029         \STEXInternalTermMathOMBiiii { \STEXInternalCurrentSymbolStr }
3030         { \l__stex_notation_suffix_str }
3031         { \l__stex_notation_opprec_tl }
3032         { \exp_not:n { #5 } }
3033       }
3034     }{
3035       \str_if_in:NnTF \l__stex_notation_args_str B {
3036         \exp_args:Nne \use:nn
3037         {
3038           \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
3039           \cs_set:Npn \l__stex_notation_arity_str } { {
3040             \STEXInternalTermMathOMBiiii { \STEXInternalCurrentSymbolStr }
3041             { \l__stex_notation_suffix_str }
3042             { \l__stex_notation_opprec_tl }
3043             { \exp_not:n { #5 } }
3044           } }

```

```

3045     }{
3046       \exp_args:Nne \use:nn
3047       {
3048         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
3049         \cs_set:Npn \l__stex_notation_arity_str } { {
3050           \STEXInternalTermMathOMAiiiii { \STEXInternalCurrentSymbolStr }
3051           { \l__stex_notation_suffix_str }
3052           { \l__stex_notation_opprec_tl }
3053           { \exp_not:n { #5 } }
3054         } }
3055       }
3056     }
3057
3058     \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
3059     \int_zero:N \l__stex_notation_currarg_int
3060     \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
3061     \__stex_notation_arguments:
3062   }
3063 }

```

(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

```

3064 \cs_new_protected:Nn \__stex_notation_arguments: {
3065   \int_incr:N \l__stex_notation_currarg_int
3066   \str_if_empty:NTF \l__stex_notation_remaining_args_str {
3067     \l_stex_notation_after_do_tl
3068   }{
3069     \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
3070     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
3071     \str_if_eq:VnTF \l_tmpa_str a {
3072       \__stex_notation_argument_assoc:nn{a}
3073     }{
3074       \str_if_eq:VnTF \l_tmpa_str B {
3075         \__stex_notation_argument_assoc:nn{B}
3076       }{
3077         \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpb_str
3078         \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
3079           { \STEXInternalTermMathArgiii
3080             { \l_tmpa_str\int_use:N \l__stex_notation_currarg_int }
3081             { \l_tmpb_str }
3082             { ###\int_use:N \l__stex_notation_currarg_int }
3083           }
3084         }
3085         \__stex_notation_arguments:
3086       }
3087     }
3088   }
3089 }

```

(End definition for `\__stex_notation_arguments:.`)

`\_stex_notation_argument_assoc:nn`

```

3090 \cs_new_protected:Nn \__stex_notation_argument_assoc:nn {

```

```

3091
3092 \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
3093   {\l__stex_notation_arity_str}{
3094     #2
3095   }
3096 \int_zero:N \l_tmpa_int
3097 \tl_clear:N \l_tmpa_tl
3098 \str_map_inline:Nn \l__stex_notation_args_str {
3099   \int_incr:N \l_tmpa_int
3100   \tl_put_right:Nx \l_tmpa_tl {
3101     \str_if_eq:nnTF {##1}{a}{ } {} }{
3102     \str_if_eq:nnTF {##1}{B}{ } {} }{
3103     {\_stex_term_arg:nn{##1}\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa_int}
3104   }
3105 }
3106 }
3107 }
3108 \exp_after:wN\exp_after:wN\exp_after:wN \def
3109 \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
3110 \exp_after:wN\exp_after:wN\exp_after:wN ##
3111 \exp_after:wN\exp_after:wN\exp_after:wN 1
3112 \exp_after:wN\exp_after:wN\exp_after:wN ##
3113 \exp_after:wN\exp_after:wN\exp_after:wN 2
3114 \exp_after:wN\exp_after:wN\exp_after:wN {
3115   \exp_after:wN \exp_after:wN \exp_after:wN
3116   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
3117     \exp_after:wN \l_tmpa_cs \l_tmpa_tl
3118   }
3119 }
3120
3121 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
3122 \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
3123   \STEXInternalTermMathAssocArgiiii
3124   { \int_use:N \l__stex_notation_currarg_int }
3125   { \l_tmpa_str }
3126   { ####\int_use:N \l__stex_notation_currarg_int }
3127   { \l_tmpa_cs {####1} {####2} }
3128   {#1}
3129 } }
3130 \__stex_notation_arguments:
3131 }

```

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

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

```

3132 \cs_new_protected:Nn \__stex_notation_restore_notation:nnnnn {
3133   \cs_generate_from_arg_count:cNnn{stex_notation_\detokenize{#1} \c_hash_str \detokenize{#2}}
3134   \cs_set_nopar:Npn {#3}{#4}
3135   \tl_if_empty:nF {#5}{
3136     \tl_set:cn{stex_op_notation_\detokenize{#1} \c_hash_str \detokenize{#2}_cs}{\comp{ #5 }
3137   }
3138   \seq_if_exist:cT { l_stex_symdecl_\detokenize{#1} _notations }{
3139     \seq_put_right:cx { l_stex_symdecl_\detokenize{#1} _notations } { \detokenize{#2} }
3140   }

```

```

3141 }
3142
3143 \cs_new_protected:Nn \__stex_notation_final: {
3144
3145   \stex_execute_in_module:x {
3146     \__stex_notation_restore_notation:nnnnn
3147     {\l_stex_get_symbol_uri_str}
3148     {\l__stex_notation_suffix_str}
3149     {\l__stex_notation_arity_str}
3150     {
3151       \exp_after:wN \exp_after:wN \exp_after:wN
3152       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
3153       { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \STEXInt
3154     }
3155     {\exp_args:No \exp_not:n \l__stex_notation_op_tl }
3156   }
3157
3158   \stex_debug:nn{symbols}{
3159     Notation~\l__stex_notation_suffix_str
3160     ~for~\l_stex_get_symbol_uri_str^^J
3161     Operator~precedence:~\l__stex_notation_opprec_tl^^J
3162     Argument~precedences:~
3163     \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
3164     Notation: \cs_meaning:c {
3165       stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
3166       \l__stex_notation_suffix_str
3167       _cs
3168     }
3169   }
3170   % HTML annotations
3171   \stex_if_do_html:T {
3172     \stex_annotate_invisible:nnn { notation }
3173     { \l_stex_get_symbol_uri_str } {
3174       \stex_annotate_invisible:nnn { notationfragment }
3175       { \l__stex_notation_suffix_str }{}
3176       \stex_annotate_invisible:nnn { precedence }
3177       { \l__stex_notation_prec_str }{}
3178
3179       \int_zero:N \l_tmpa_int
3180       \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
3181       \tl_clear:N \l_tmpa_tl
3182       \int_step_inline:nn { \l__stex_notation_arity_str }{
3183         \int_incr:N \l_tmpa_int
3184         \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
3185         \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_rema
3186         \str_if_eq:VnTF \l_tmpb_str a {
3187           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3188             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3189             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3190           } }
3191         }{
3192           \str_if_eq:VnTF \l_tmpb_str B {
3193             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3194               \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,

```



```

3195         \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{\}
3196     } }
3197   }{
3198     \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3199       \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{\}
3200     } }
3201   }
3202 }
3203 }
3204 \stex_annotate_invisible:nnn { notationcomp }{ }{
3205   \str_set:Nx \STEXInternalCurrentSymbolStr {\l_stex_get_symbol_uri_str }
3206   $ \exp_args:Nno \use:nn { \use:c {
3207     stex_notation_ \STEXInternalCurrentSymbolStr
3208     \c_hash_str \l__stex_notation_suffix_str_cs
3209   } } { \l_tmpa_tl } $
3210 }
3211 \tl_if_empty:NF \l__stex_notation_op_tl {
3212   \stex_annotate_invisible:nnn { notationopcomp }{ }{
3213     $\l__stex_notation_op_tl$
3214   }
3215 }
3216 }
3217 }
3218 }

```

(End definition for `\_stex_notation_final:`)

## `\setnotation`

```

3219 \keys_define:nn { stex / setnotation } {
3220   % lang      .tl_set_x:N = \l__stex_notation_lang_str ,
3221   variant .tl_set_x:N = \l__stex_notation_variant_str ,
3222   unknown .code:n      = \str_set:Nx
3223     \l__stex_notation_variant_str \l_keys_key_str
3224 }
3225
3226 \cs_new_protected:Nn \_stex_setnotation_args:n {
3227   % \str_clear:N \l__stex_notation_lang_str
3228   \str_clear:N \l__stex_notation_variant_str
3229   \keys_set:nn { stex / setnotation } { #1 }
3230 }
3231
3232 \cs_new_protected:Nn \_stex_notation_setnotation:nn {
3233   \seq_if_exist:cT{l_stex_symdecl_#1_notations}{
3234     \seq_remove_all:cn { l_stex_symdecl_#1_notations }{ #2 }
3235     \seq_put_left:cn { l_stex_symdecl_#1_notations }{ #2 }
3236   }
3237 }
3238
3239 \cs_new_protected:Nn \stex_setnotation:n {
3240   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1_notations }
3241     { \l__stex_notation_variant_str }{
3242     \stex_execute_in_module:x{ \_stex_notation_setnotation:nn {#1}{\l__stex_notation_vari
3243     \stex_debug:nn {notations}{
3244       Setting~default~notation~

```

```

3245         {\l__stex_notation_variant_str }~for~
3246         #1 \\\
3247         \expandafter\meaning\csname
3248         l_stex_symdecl_#1 _notations\endcsname
3249     }
3250 }{
3251     \msg_error:nnxx{stex}{unknownnotation}{\l__stex_notation_variant_str}{#1}
3252 }
3253 }
3254
3255 \NewDocumentCommand \setnotation {m m} {
3256     \stex_get_symbol:n { #1 }
3257     \_stex_setnotation_args:n { #2 }
3258     \stex_setnotation:n{\l_stex_get_symbol_uri_str}
3259     \stex_smsmode_do:\ignorespacesandpars
3260 }
3261
3262 \cs_new_protected:Nn \stex_copy_notations:nn {
3263     \stex_debug:nn {notations}{
3264         Copying~notations~from~#2~to~#1\\
3265         \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
3266     }
3267     \tl_clear:N \l_tmpa_tl
3268     \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
3269         \tl_put_right:Nn \l_tmpa_tl { {##### #1} }
3270     }
3271     \seq_map_inline:cn {l_stex_symdecl_#2_notations}{\begingroup
3272         \stex_debug:nn{Here}{Here:~##1}
3273         \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
3274         \edef \l_tmpa_tl {
3275             \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
3276             \exp_after:wN\exp_after:wN\exp_after:wN {
3277                 \exp_after:wN \l_tmpa_cs \l_tmpa_tl
3278             }
3279         }
3280
3281         \exp_after:wN \def \exp_after:wN \l_tmpa_tl
3282         \exp_after:wN #####\exp_after:wN 1 \exp_after:wN #####\exp_after:wN 2
3283         \exp_after:wN { \l_tmpa_tl }
3284
3285         \edef \l_tmpa_tl {
3286             \exp_after:wN \exp_not:n \exp_after:wN {
3287                 \l_tmpa_tl {##### 1}{##### 2}
3288             }
3289         }
3290
3291         \stex_debug:nn{Here}{Here:~\expandafter\detokenize\expandafter{\l_tmpa_tl}}
3292
3293     \stex_execute_in_module:x {
3294         \_stex_notation_restore_notation:nnnnn
3295         {#1}{##1}
3296         { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }
3297         { \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl} }
3298         {

```

```

3299         \cs_if_exist:cT{stex_op_notation_ #2\c_hash_str ##1 _cs}{
3300             \exp_args:NNo\exp_args:No\exp_not:n{\csname stex_op_notation_ #2\c_hash_str ##1
3301             }
3302         }
3303     }\endgroup
3304 }
3305 }
3306
3307 \NewDocumentCommand \copynotation {m m} {
3308     \stex_get_symbol:n { #1 }
3309     \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
3310     \stex_get_symbol:n { #2 }
3311     \exp_args:Noo
3312     \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
3313     \stex_smsmode_do:\ignorespacesandpars
3314 }
3315

```

(End definition for \setnotation. This function is documented on page 23.)

## \symdef

```

3316 \keys_define:nn { stex / symdef } {
3317     name .str_set_x:N = \l_stex_symdecl_name_str ,
3318     args .str_set_x:N = \l_stex_symdecl_args_str ,
3319     type .tl_set:N = \l_stex_symdecl_type_tl ,
3320     def .tl_set:N = \l_stex_symdecl_definiens_tl ,
3321     reorder .str_set_x:N = \l_stex_symdecl_reorder_str ,
3322     op .tl_set:N = \l__stex_notation_op_tl ,
3323     % lang .str_set_x:N = \l__stex_notation_lang_str ,
3324     variant .str_set_x:N = \l__stex_notation_variant_str ,
3325     prec .str_set_x:N = \l__stex_notation_prec_str ,
3326     argnames .clist_set:N = \l_stex_symdecl_argnames_clist ,
3327     assoc .choices:nn =
3328         {bin,binl,binr,pre,conj,pwconj}
3329         {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
3330     unknown .code:n = \str_set:Nx
3331         \l__stex_notation_variant_str \l_keys_key_str
3332 }
3333
3334 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
3335     \str_clear:N \l_stex_symdecl_name_str
3336     \str_clear:N \l_stex_symdecl_args_str
3337     \str_clear:N \l_stex_symdecl_assoctype_str
3338     \str_clear:N \l_stex_symdecl_reorder_str
3339     \bool_set_false:N \l_stex_symdecl_local_bool
3340     \tl_clear:N \l_stex_symdecl_type_tl
3341     \tl_clear:N \l_stex_symdecl_definiens_tl
3342     \clist_clear:N \l_stex_symdecl_argnames_clist
3343     % \str_clear:N \l__stex_notation_lang_str
3344     \str_clear:N \l__stex_notation_variant_str
3345     \str_clear:N \l__stex_notation_prec_str
3346     \tl_clear:N \l__stex_notation_op_tl
3347
3348     \keys_set:nn { stex / symdef } { #1 }

```

```

3349 }
3350
3351 \NewDocumentCommand \symdef { m O{} } {
3352   \_stex_notation_symdef_args:n { #2 }
3353   \bool_set_true:N \l_stex_symdecl_make_macro_bool
3354   \stex_symdecl_do:n { #1 }
3355   \tl_set:Nn \l_stex_notation_after_do_tl {
3356     \_stex_notation_final:
3357     \stex_smsmode_do:\ignorespacesandpars
3358   }
3359   \str_set:Nx \l_stex_get_symbol_uri_str {
3360     \l_stex_current_module_str ? \l_stex_symdecl_name_str
3361   }
3362   \exp_args:Nx \stex_notation_do:nnnnn
3363     { \prop_item:cn {l_stex_symdecl\_l_stex_get_symbol_uri_str_prop } { args } }
3364     { \prop_item:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str_prop } { arity } }
3365     { \l__stex_notation_variant_str }
3366     { \l__stex_notation_prec_str}
3367 }
3368 \stex_deactivate_macro:Nn \symdef {module~environments}
3369
3370 \keys_define:nn { stex / mmtdef } {
3371   name .str_set_x:N = \l_stex_symdecl_name_str ,
3372   args .str_set_x:N = \l_stex_symdecl_args_str ,
3373   reorder .str_set_x:N = \l_stex_symdecl_reorder_str ,
3374   op .tl_set:N = \l__stex_notation_op_tl ,
3375   % lang .str_set_x:N = \l__stex_notation_lang_str ,
3376   variant .str_set_x:N = \l__stex_notation_variant_str ,
3377   prec .str_set_x:N = \l__stex_notation_prec_str ,
3378   argnames .clist_set:N = \l_stex_symdecl_argnames_clist ,
3379   assoc .choices:nn =
3380     {bin,binl,binr,pre,conj,pwconj}
3381     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
3382   unknown .code:n = \str_set:Nx
3383     \l__stex_notation_variant_str \l_keys_key_str
3384 }
3385 \cs_new_protected:Nn \_stex_mmtdef_args:n {
3386   \str_clear:N \l_stex_symdecl_name_str
3387   \str_clear:N \l_stex_symdecl_args_str
3388   \str_clear:N \l_stex_symdecl_assoctype_str
3389   \str_clear:N \l_stex_symdecl_reorder_str
3390   \bool_set_false:N \l_stex_symdecl_local_bool
3391   \clist_clear:N \l_stex_symdecl_argnames_clist
3392   % \str_clear:N \l__stex_notation_lang_str
3393   \str_clear:N \l__stex_notation_variant_str
3394   \str_clear:N \l__stex_notation_prec_str
3395   \tl_clear:N \l__stex_notation_op_tl
3396
3397   \keys_set:nn { stex / mmtdef } { #1 }
3398 }
3399
3400 \NewDocumentCommand \mmtdef {m O{} }{
3401   \_stex_mmtdef_args:n{ #2 }
3402   \bool_set_true:N \l_stex_symdecl_make_macro_bool

```

```

3403 \str_if_empty:NT \l_stex_symdecl_name_str {
3404   \str_set:Nx \l_stex_symdecl_name_str { #1 }
3405 }
3406 %\tl_set:Nx \l_stex_symdecl_definiens_tl {
3407 % \stex_annotate:nnn{ OMID }{
3408 %   \l_stex_module_mmtfor_str?\l_stex_symdecl_name_str
3409 % }{}
3410 %}
3411 \stex_symdecl_do:n { #1 }
3412 \stex_if_smsmode:F{
3413   \MMTrule{rules.stex.mmt.kwarc.info?SubstitutionRule}{
3414     \stex_annotate:nnn{ OMID }{
3415       \l_stex_current_module_str ? \l_stex_symdecl_name_str
3416     }{},
3417     \stex_annotate:nnn{ OMID }{
3418       \l_stex_module_mmtfor_str?\l_stex_symdecl_name_str
3419     }{}
3420   }
3421 }
3422 \tl_set:Nn \l_stex_notation_after_do_tl {
3423   \__stex_notation_final:
3424   \stex_smsmode_do:\ignorespacesandpars
3425 }
3426 \str_set:Nx \l_stex_get_symbol_uri_str {
3427   \l_stex_current_module_str ? \l_stex_symdecl_name_str
3428 }
3429 \exp_args:Nx \stex_notation_do:nnnnn
3430 { \prop_item:cn {l_stex_symdecl\_l_stex_get_symbol_uri_str _prop } { args } }
3431 { \prop_item:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _prop } { arity } }
3432 { \l__stex_notation_variant_str }
3433 { \l__stex_notation_prec_str}
3434 }

```

(End definition for `\symdef`. This function is documented on page 91.)

### 31.3 Variables

```

3435 <@@=stex_variables>
3436
3437 \keys_define:nn { stex / vardef } {
3438   name .str_set_x:N = \l__stex_variables_name_str ,
3439   args .str_set_x:N = \l__stex_variables_args_str ,
3440   type .tl_set:N = \l__stex_variables_type_tl ,
3441   def .tl_set:N = \l__stex_variables_def_tl ,
3442   op .tl_set:N = \l__stex_variables_op_tl ,
3443   prec .str_set_x:N = \l__stex_variables_prec_str ,
3444   reorder .str_set_x:N = \l__stex_variables_reorder_str ,
3445   argnames .clist_set:N = \l__stex_variables_argnames_clist ,
3446   assoc .choices:nn =
3447     {bin,binl,binr,pre,conj,pwconj}
3448     {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
3449   bind .choices:nn =
3450     {forall,exists}
3451     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}

```

```

3452 }
3453
3454 \cs_new_protected:Nn \__stex_variables_args:n {
3455   \str_clear:N \l__stex_variables_name_str
3456   \str_clear:N \l__stex_variables_args_str
3457   \str_clear:N \l__stex_variables_prec_str
3458   \str_clear:N \l__stex_variables_assoctype_str
3459   \str_clear:N \l__stex_variables_reorder_str
3460   \str_clear:N \l__stex_variables_bind_str
3461   \tl_clear:N \l__stex_variables_type_tl
3462   \tl_clear:N \l__stex_variables_def_tl
3463   \tl_clear:N \l__stex_variables_op_tl
3464   \clist_clear:N \l__stex_variables_argnames_clist
3465
3466   \keys_set:nn { stex / vardef } { #1 }
3467 }
3468
3469 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{} } {
3470   \__stex_variables_args:n {#2}
3471   \str_if_empty:NT \l__stex_variables_name_str {
3472     \str_set:Nx \l__stex_variables_name_str { #1 }
3473   }
3474   \prop_clear:N \l_tmpa_prop
3475   \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
3476
3477   \int_zero:N \l_tmpb_int
3478   \bool_set_true:N \l_tmpa_bool
3479   \str_map_inline:Nn \l__stex_variables_args_str {
3480     \token_case_meaning:NnF ##1 {
3481       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
3482       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
3483       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
3484       {\tl_to_str:n a} {
3485         \bool_set_false:N \l_tmpa_bool
3486         \int_incr:N \l_tmpb_int
3487       }
3488       {\tl_to_str:n B} {
3489         \bool_set_false:N \l_tmpa_bool
3490         \int_incr:N \l_tmpb_int
3491       }
3492     }{
3493       \msg_error:nnxx{stex}{error/wrongargs}{
3494         variable~\l__stex_variables_name_str
3495       }{##1}
3496     }
3497   }
3498   \bool_if:NTF \l_tmpa_bool {
3499     % possibly numeric
3500     \str_if_empty:NTF \l__stex_variables_args_str {
3501       \prop_put:Nnn \l_tmpa_prop { args } {}
3502       \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
3503     }{
3504       \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
3505       \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }

```

```

3506     \str_clear:N \l_tmpa_str
3507     \int_step_inline:nn \l_tmpa_int {
3508         \str_put_right:Nn \l_tmpa_str i
3509     }
3510     \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
3511     \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3512 }
3513 } {
3514     \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3515     \prop_put:Nnx \l_tmpa_prop { arity }
3516     { \str_count:N \l__stex_variables_args_str }
3517 }
3518 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
3519 \tl_set:cx { #1 } { \stex_invoke_variable:n { \l__stex_variables_name_str } }
3520
3521 % argnames
3522
3523 \clist_clear:N \l_tmpa_clist
3524 \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } } {
3525     \clist_if_empty:NTF \l__stex_variables_argnames_clist {
3526         \clist_put_right:Nn \l_tmpa_clist { ##1 }
3527     } {
3528         \clist_pop:NN \l__stex_variables_argnames_clist \l_tmpa_tl
3529         \exp_args:NNx \clist_put_right:Nn \l_tmpa_clist { \c_dollar_str \l_tmpa_tl }
3530     }
3531 }
3532 \prop_put:Nnx \l_tmpa_prop { argnames } { \clist_use:Nn \l_tmpa_clist , }
3533
3534
3535 \prop_set_eq:cN { l_stex_symdecl_var://\l__stex_variables_name_str _prop } \l_tmpa_prop
3536
3537 \tl_if_empty:NF \l__stex_variables_op_tl {
3538     \cs_set:cpx {
3539         stex_var_op_notation_ \l__stex_variables_name_str _cs
3540     } { \exp_not:N \comp { \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
3541 }
3542
3543 \tl_set:Nn \l_stex_notation_after_do_tl {
3544     \exp_args:Nne \use:nn {
3545         \cs_generate_from_arg_count:cNnn { stex_var_notation_ \l__stex_variables_name_str _cs }
3546         \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
3547     } { {
3548         \exp_after:wN \exp_after:wN \exp_after:wN
3549         \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
3550         { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \STEXInter
3551     } }
3552
3553 \stex_if_do_html:T {
3554     \stex_annotate_invisible:nnn { vardecl } { \l__stex_variables_name_str } {
3555         \stex_annotate_invisible:nnn { precedence }
3556         { \l__stex_variables_prec_str } { }
3557         \tl_if_empty:NF \l__stex_variables_type_tl { \stex_annotate_invisible:nnn { type } { } { $ \
3558         \stex_annotate_invisible:nnn { args } { \l__stex_variables_args_str } { }
3559         \stex_annotate_invisible:nnn { macroname } { #1 } { }
3560         \tl_if_empty:NF \l__stex_variables_def_tl {

```

```

3560         \stex_annotate_invisible:nnn{definiens}{}
3561         {${\l__stex_variables_def_tl$}
3562     }
3563     \str_if_empty:NF \l__stex_variables_assoctype_str {
3564         \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{}
3565     }
3566     \str_if_empty:NF \l__stex_variables_reorder_str {
3567         \stex_annotate_invisible:nnn{reorderargs}{\l__stex_variables_reorder_str}{}
3568     }
3569     \int_zero:N \l_tmpa_int
3570     \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
3571     \tl_clear:N \l_tmpa_tl
3572     \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
3573         \int_incr:N \l_tmpa_int
3574         \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
3575         \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables
3576         \str_if_eq:VnTF \l_tmpb_str a {
3577             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3578                 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3579                 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3580             } }
3581         }{
3582             \str_if_eq:VnTF \l_tmpb_str B {
3583                 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3584                     \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3585                     \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3586                 } }
3587             }{
3588                 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3589                     \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
3590                 } }
3591             }
3592         }
3593     }
3594     \stex_annotate_invisible:nnn { notationcomp }{}{
3595         \str_set:Nx \STEXInternalCurrentSymbolStr {var://\l__stex_variables_name_str }
3596         $ \exp_args:Nno \use:nn { \use:c {
3597             stex_var_notation_\l__stex_variables_name_str_cs
3598         } } { \l_tmpa_tl } $
3599     }
3600     \tl_if_empty:NF \l__stex_variables_op_tl {
3601         \stex_annotate_invisible:nnn { notationopcomp }{}{
3602             ${\l__stex_variables_op_tl$}
3603         }
3604     }
3605 }
3606 \str_if_empty:NF \l__stex_variables_bind_str {
3607     \stex_annotate_invisible:nnn {bindtype}{\l__stex_variables_bind_str,\l__stex_variabl
3608 }
3609 }\ignorespacesandpars
3610 }
3611
3612 \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
3613 }

```



```

3614
3615 \cs_new:Nn \_stex_reset:N {
3616   \tl_if_exist:NTF #1 {
3617     \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
3618   }{
3619     \let \exp_not:N #1 \exp_not:N \undefined
3620   }
3621 }
3622
3623 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
3624   \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
3625   \exp_args:Nnx \use:nn {
3626     % TODO
3627     \stex_annotate_invisible:nnn {vardecl}{\clist_use:Nn\l__stex_variables_names,}{
3628       #2
3629     }
3630   }{
3631     \_stex_reset:N \varnot
3632     \_stex_reset:N \vartype
3633     \_stex_reset:N \vardefi
3634   }
3635 }
3636
3637 \NewDocumentCommand \vardef { s } {
3638   \IfBooleanTF#1 {
3639     \__stex_variables_do_complex:nn
3640   }{
3641     \__stex_variables_do_simple:nnn
3642   }
3643 }
3644
3645 \NewDocumentCommand \svar { 0{} m }{
3646   \tl_if_empty:nTF {#1}{
3647     \str_set:Nn \l_tmpa_str { #2 }
3648   }{
3649     \str_set:Nn \l_tmpa_str { #1 }
3650   }
3651   \_stex_term_omv:nn {
3652     var://\l_tmpa_str
3653   }{
3654     \exp_args:Nnx \use:nn {
3655       \def\comp{\_varcomp}
3656       \str_set:Nx \STEXInternalCurrentSymbolStr { var://\l_tmpa_str }
3657       \comp{ #2 }
3658     }{
3659       \_stex_reset:N \comp
3660       \_stex_reset:N \STEXInternalCurrentSymbolStr
3661     }
3662   }
3663 }
3664
3665
3666
3667 \keys_define:nn { stex / varseq } {

```

```

3668 name      .str_set_x:N = \l__stex_variables_name_str ,
3669 args      .int_set:N = \l__stex_variables_args_int ,
3670 type      .tl_set:N = \l__stex_variables_type_tl ,
3671 mid       .tl_set:N = \l__stex_variables_mid_tl ,
3672 bind      .choices:nn =
3673           {forall,exists}
3674           {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
3675 }
3676
3677 \cs_new_protected:Nn \__stex_variables_seq_args:n {
3678   \str_clear:N \l__stex_variables_name_str
3679   \int_set:Nn \l__stex_variables_args_int 1
3680   \tl_clear:N \l__stex_variables_type_tl
3681   \str_clear:N \l__stex_variables_bind_str
3682
3683   \keys_set:nn { stex / varseq } { #1 }
3684 }
3685
3686 \NewDocumentCommand \varseq {m O{} m m m}{
3687   \__stex_variables_seq_args:n { #2 }
3688   \str_if_empty:NT \l__stex_variables_name_str {
3689     \str_set:Nx \l__stex_variables_name_str { #1 }
3690   }
3691   \prop_clear:N \l_tmpa_prop
3692   \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
3693
3694   \seq_set_from_clist:Nn \l_tmpa_seq {#3}
3695   \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
3696     \msg_error:nnxx{stex}{error/seqlength}
3697     {\int_use:N \l__stex_variables_args_int}
3698     {\seq_count:N \l_tmpa_seq}
3699   }
3700   \seq_set_from_clist:Nn \l_tmpb_seq {#4}
3701   \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
3702     \msg_error:nnxx{stex}{error/seqlength}
3703     {\int_use:N \l__stex_variables_args_int}
3704     {\seq_count:N \l_tmpb_seq}
3705   }
3706   \prop_put:Nnn \l_tmpa_prop {starts} {#3}
3707   \prop_put:Nnn \l_tmpa_prop {ends} {#4}
3708
3709   \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str_cs}
3710   \cs_set:Npn {\int_use:N \l__stex_variables_args_int} { #5 }
3711
3712   % argnames
3713
3714   \clist_clear:N \l_tmpa_clist
3715   \int_step_inline:nn {\l__stex_variables_args_int} {
3716     \clist_put_right:Nn \l_tmpa_clist {##1}
3717   }
3718   \prop_put:Nnx \l_tmpa_prop {argnames} {\clist_use:Nn \l_tmpa_clist ,}
3719
3720
3721

```

```

3722 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str_cs}}
3723 \int_step_inline:nn \l__stex_variables_args_int {
3724   \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
3725 }
3726 \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
3727 \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
3728 \tl_if_empty:NF \l__stex_variables_mid_tl {
3729   \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
3730   \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
3731 }
3732 \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq_\l__stex_variables_name_str_cs}}
3733 \int_step_inline:nn \l__stex_variables_args_int {
3734   \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
3735 }
3736 \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
3737 \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
3738
3739
3740
3741 \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl
3742
3743 \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}
3744
3745 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str_cs}}
3746
3747 \int_step_inline:nn \l__stex_variables_args_int {
3748   \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
3749     \STEXInternalTermMathArgiii{i##1}{0}{\exp_not:n{####}##1}
3750   }}
3751 }
3752
3753 \tl_set:Nx \l_tmpa_tl {
3754   \STEXInternalTermMathOMaiiii { varseq://\l__stex_variables_name_str}{0}{
3755     \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
3756   }
3757 }
3758
3759 \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \STEXInternalSymbolAfterInvokationTL} }
3760
3761 \exp_args:Nno \use:nn {
3762   \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str_cs}
3763   \cs_set:Npn {\int_use:N \l__stex_variables_args_int}{\l_tmpa_tl}
3764
3765   \stex_debug:nn{sequences}{New~Sequence:~
3766     \expandafter\meaning\csname stex_varseq_\l__stex_variables_name_str_cs\endcsname\\~\\
3767     \prop_to_keyval:N \l_tmpa_prop
3768   }
3769   \prop_set_eq:cN {\l_stex_symdecl_varseq://\l__stex_variables_name_str_prop}\l_tmpa_prop
3770
3771   \stex_if_do_html:T{\stex_annotate_invisible:nnn{varseq}{\l__stex_variables_name_str}{
3772     \tl_if_empty:NF \l__stex_variables_type_tl {
3773       \stex_annotate:nnn {type}{\l__stex_variables_type_tl$}
3774     }
3775     \stex_annotate:nnn {args}{\int_use:N \l__stex_variables_args_int}{

```

```

3776 \str_if_empty:NF \l__stex_variables_bind_str {
3777   \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{}
3778 }
3779 \stex_annotate:nnn{startindex}{}{${#3$}
3780 \stex_annotate:nnn{endindex}{}{${#4$}
3781
3782 \tl_clear:N \l_tmpa_tl
3783 \int_step_inline:nn \l__stex_variables_args_int {
3784   \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3785     \stex_annotate:nnn{argmarker}{##1}{}
3786   } }
3787 }
3788 \stex_annotate_invisible:nnn { notationcomp }{}{
3789   \str_set:Nx \STEXInternalCurrentSymbolStr {varseq://\l__stex_variables_name_str }
3790   $ \exp_args:Nno \use:nn { \use:c {
3791     stex_varseq\l__stex_variables_name_str _cs
3792   } } { \l_tmpa_tl } $
3793 }
3794 \stex_annotate_invisible:nnn { notationopcomp }{}{
3795   $ \prop_item:Nn \l_tmpa_prop { notation } $
3796 }
3797
3798 }}
3799
3800 \ignorespacesandpars
3801 }
3802
3803
3804 \keys_define:nn { stex / mmtdecl } {
3805   name      .str_set_x:N = \l_stex_symdecl_name_str ,
3806   args      .str_set_x:N = \l_stex_symdecl_args_str ,
3807   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,
3808   reorder   .str_set_x:N = \l_stex_symdecl_reorder_str ,
3809   argnames  .clist_set:N = \l_stex_symdecl_argnames_clist ,
3810   assoc     .choices:nn =
3811     {bin,binl,binr,pre,conj,pwconj}
3812     {\str_set:Nx \l_stex_symdecl_assoc_type_str {\l_keys_choice_tl}}
3813 }
3814
3815 \cs_new_protected:Nn \_stex_mmtdecl_args:n {
3816   \str_clear:N \l_stex_symdecl_name_str
3817   \str_clear:N \l_stex_symdecl_args_str
3818   \str_clear:N \l_stex_symdecl_deprecate_str
3819   \str_clear:N \l_stex_symdecl_reorder_str
3820   \str_clear:N \l_stex_symdecl_assoc_type_str
3821   \bool_set_false:N \l_stex_symdecl_local_bool
3822   \clist_clear:N \l_stex_symdecl_argnames_clist
3823
3824   \keys_set:nn { stex / symdecl } { #1 }
3825 }
3826
3827 \NewDocumentCommand \mmtdecl { s m O{} } {
3828   \_stex_mmtdecl_args:n{#3}
3829   \IfBooleanTF #1 {

```

```

3830   \bool_set_false:N \l_stex_symdecl_make_macro_bool
3831 } {
3832   \bool_set_true:N \l_stex_symdecl_make_macro_bool
3833 }
3834 \str_if_empty:NT \l_stex_symdecl_name_str {
3835   \str_set:Nx \l_stex_symdecl_name_str { #1 }
3836 }
3837 %\tl_set:Nx \l_stex_symdecl_definiens_tl {
3838 %   \stex_annotate:nnn{ OMID }{
3839 %     \l_stex_module_mmtfor_str?\l_stex_symdecl_name_str
3840 %   }{}
3841 %}
3842 \stex_symdecl_do:n{#2}
3843 \MMTrule{rules.stex.mmt.kwarc.info?SubstitutionRule}{
3844   \stex_annotate:nnn{ OMID }{
3845     \l_stex_current_module_str ? \l_stex_symdecl_name_str
3846   }{},
3847   \stex_annotate:nnn{ OMID }{
3848     \l_stex_module_mmtfor_str?\l_stex_symdecl_name_str
3849   }{}
3850 }
3851 \stex_smsmode_do:
3852 }
3853
3854 \stex_deactivate_macro:Nn \mmtdecl {mmtinterface~environments}
3855 \stex_deactivate_macro:Nn \mmtdef {mmtinterface~environments}
3856
3857 </package>

```

## Chapter 32

# STEX -Terms Implementation

```
3858 <*package>
3859
3860 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
3861
3862 <@@=stex_terms>
3863
3864   Warnings and error messages
3865   \msg_new:nnn{stex}{error/nonotation}{
3866     Symbol~#1~invoked,~but~has~no~notation#2!
3867   }
3868   \msg_new:nnn{stex}{error/notationarg}{
3869     Error~in~parsing~notation~#1
3870   }
3871   \msg_new:nnn{stex}{error/noop}{
3872     Symbol~#1~has~no~operator~notation~for~notation~#2
3873   }
3874   \msg_new:nnn{stex}{error/notallowed}{
3875     Symbol~invocation~#1~not~allowed~in~notation~component~of~#2
3876   }
3877   \msg_new:nnn{stex}{error/doubleargument}{
3878     Argument~#1~of~symbol~#2~already~assigned
3879   }
3880   \msg_new:nnn{stex}{error/overarity}{
3881     Argument~#1~invalid~for~symbol~#2~with~arity~#3
3882   }
```

### 32.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
3882
3883
3884 \bool_new:N \l_stex_allow_semantic_bool
3885 \bool_set_true:N \l_stex_allow_semantic_bool
3886
```

```

3887 \cs_new_protected:Nn \stex_invoke_symbol:n {
3888   \ifvmode\indent\fi
3889   \bool_if:NTF \l_stex_allow_semantic_bool {
3890     \str_if_eq:eeF {
3891       \prop_item:cn {
3892         l_stex_symdecl_#1_prop
3893       }{ deprecate }
3894     }{}{
3895       \msg_warning:nxxx{stex}{warning/deprecated}{
3896         Symbol~#1
3897       }{
3898         \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
3899       }
3900     }
3901     \if_mode_math:
3902       \exp_after:wN \__stex_terms_invoke_math:n
3903     \else:
3904       \exp_after:wN \__stex_terms_invoke_text:n
3905     \fi: { #1 }
3906   }{
3907     \msg_error:nxxx{stex}{error/notallowed}{#1}{\STEXInternalCurrentSymbolStr}
3908   }
3909 }
3910
3911 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
3912   \peek_charcode_remove:NTF ! {
3913     \__stex_terms_invoke_op_custom:nn {#1}
3914   }{
3915     \__stex_terms_invoke_custom:nn {#1}
3916   }
3917 }
3918
3919 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
3920   \peek_charcode_remove:NTF ! {
3921     % operator
3922     \peek_charcode_remove:NTF * {
3923       % custom op
3924       \__stex_terms_invoke_op_custom:nn {#1}
3925     }{
3926       % op notation
3927       \peek_charcode:NTF [ {
3928         \__stex_terms_invoke_op_notation:nw {#1}
3929       }{
3930         \__stex_terms_invoke_op_notation:nw {#1}[]
3931       }
3932     }
3933   }{
3934     \peek_charcode_remove:NTF * {
3935       \__stex_terms_invoke_custom:nn {#1}
3936       % custom
3937     }{
3938       % normal
3939       \peek_charcode:NTF [ {
3940         \__stex_terms_invoke_notation:nw {#1}

```

```

3941     }{
3942       \_stex_terms_invoke_notation:nw {#1}[]
3943     }
3944   }
3945 }
3946 }
3947
3948
3949 \cs_new_protected:Nn \_stex_terms_invoke_op_custom:nn {
3950   \exp_args:Nnx \use:nn {
3951     \def\comp{\_comp}
3952     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3953     \bool_set_false:N \l_stex_allow_semantic_bool
3954     \stex_mathml_intent:nn{#1}{
3955       \_stex_term_oms:nnn {#1}{#1 \c_hash_str CUSTOM-}{
3956         \comp{ #2 }
3957       }
3958     }
3959   }{
3960     \_stex_reset:N \comp
3961     \_stex_reset:N \STEXInternalCurrentSymbolStr
3962     \bool_set_true:N \l_stex_allow_semantic_bool
3963   }
3964 }
3965
3966 \keys_define:nn { stex / terms } {
3967   % lang      .tl_set_x:N = \l_stex_notation_lang_str ,
3968   variant .tl_set_x:N = \l_stex_notation_variant_str ,
3969   unknown .code:n      = \str_set:Nx
3970     \l_stex_notation_variant_str \l_keys_key_str
3971 }
3972
3973 \cs_new_protected:Nn \_stex_terms_args:n {
3974   % \str_clear:N \l_stex_notation_lang_str
3975   \str_clear:N \l_stex_notation_variant_str
3976
3977   \keys_set:nn { stex / terms } { #1 }
3978 }
3979
3980 \cs_new_protected:Nn \stex_find_notation:nn {
3981   \_stex_terms_args:n { #2 }
3982   \seq_if_empty:cTF {
3983     \l_stex_symdecl_ #1 _notations
3984   } {
3985     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
3986   } {
3987     \str_if_empty:NTF \l_stex_notation_variant_str {
3988       \seq_get_left:cN { \l_stex_symdecl_ #1 _notations } \l_stex_notation_variant_str
3989     } {
3990       \seq_if_in:cxTF { \l_stex_symdecl_ #1 _notations } {
3991         \l_stex_notation_variant_str
3992       } {
3993         % \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3994       } {

```



```

3995         \msg_error:nxxx{stex}{error/nonotation}{#1}{
3996         ~\l_stex_notation_variant_str
3997     }
3998 }
3999 }
4000 }
4001 }
4002
4003 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
4004     \exp_args:Nnx \use:nn {
4005         \def\comp{\_comp}
4006         \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
4007         \stex_find_notation:nn { #1 }{ #2 }
4008         \bool_set_false:N \l_stex_allow_semantic_bool
4009         \cs_if_exist:cTF {
4010             stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
4011         }{
4012             \_stex_term_oms:nnn { #1 }{
4013                 #1 \c_hash_str \l_stex_notation_variant_str
4014             }{
4015                 \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
4016             }
4017         }{
4018             \int_compare:nNnTF {\prop_item:cn {l_stex_symdecl_#1_prop}{arity}} = 0{
4019                 \cs_if_exist:cTF {
4020                     stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
4021                 }{
4022                     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4023                         \_stex_reset:N \comp
4024                         \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
4025                         \_stex_reset:N \STEXInternalCurrentSymbolStr
4026                         \bool_set_true:N \l_stex_allow_semantic_bool
4027                     }
4028                     \def\comp{\_comp}
4029                     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
4030                     \bool_set_false:N \l_stex_allow_semantic_bool
4031                     \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
4032                 }{
4033                     \msg_error:nxxx{stex}{error/nonotation}{#1}{
4034                     ~\l_stex_notation_variant_str
4035                     }
4036                 }
4037             }{
4038                 \msg_error:nxxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
4039             }
4040         }
4041     }{
4042         \_stex_reset:N \comp
4043         \_stex_reset:N \STEXInternalCurrentSymbolStr
4044         \bool_set_true:N \l_stex_allow_semantic_bool
4045     }
4046 }
4047
4048 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {

```

```

4049 \stex_find_notation:nn { #1 }{ #2 }
4050 \cs_if_exist:cTF {
4051   stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
4052 }{
4053   \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4054     \_stex_reset:N \comp
4055     \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
4056     \_stex_reset:N \STEXInternalCurrentSymbolStr
4057     \bool_set_true:N \l_stex_allow_semantic_bool
4058   }
4059   \def\comp{\_comp}
4060   \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
4061   \bool_set_false:N \l_stex_allow_semantic_bool
4062   \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
4063 }{
4064   \msg_error:nnxx{stex}{error/nonotation}{#1}{
4065     ~\l_stex_notation_variant_str
4066   }
4067 }
4068 }
4069
4070 \prop_new:N \l__stex_terms_custom_args_prop
4071 \clist_new:N \l_stex_argnames_seq
4072 \seq_new:N \l__stex_terms_tmp_seq
4073
4074 \cs_new_protected:Nn \__stex_terms_custom_comp:n{ \bool_set_false:N \l_stex_allow_semantic_bo
4075
4076 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
4077   \exp_args:Nnx \use:nn {
4078     \def\comp{\__stex_terms_custom_comp:n}
4079     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
4080     \prop_clear:N \l__stex_terms_custom_args_prop
4081     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
4082     \prop_get:cnN {
4083       l_stex_symdecl_#1 _prop
4084     }{ args } \l_tmpa_str
4085     \exp_args:NNx \seq_set_from_clist:Nn \l_stex_argnames_seq {
4086       \prop_item:cn {l_stex_symdecl_#1 _prop}{argnames}
4087     }
4088     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
4089     \tl_set:Nn \arg { \__stex_terms_arg: }
4090     \str_if_empty:NTF \l_tmpa_str {
4091       \stex_mathml_intent:nn{#1}{
4092         \_stex_term_oms:nnn {#1}{#1\c_hash_str CUSTOM-}{\ignorespaces#2}
4093       }
4094     }{
4095       \seq_clear:N \l__stex_terms_tmp_seq
4096       \exp_args:Nx\int_step_inline:nn{\prop_item:cn{l_stex_symdecl_#1 _prop}{arity}}{
4097         \tl_set:Nx \l__stex_terms_tmp_tl {\seq_item:Nn \l_stex_argnames_seq {##1}}
4098         \bool_lazy_or:nnT{
4099           \str_if_eq_p:nn{a}{\str_item:Nn \l_tmpa_str{##1}}
4100         }{
4101           \str_if_eq_p:nn{B}{\str_item:Nn \l_tmpa_str{##1}}
4102         }{

```

```

4103         \tl_put_right:Nn \l__stex_terms_tmp_tl +
4104     }
4105     \seq_put_right:No \l__stex_terms_tmp_seq \l__stex_terms_tmp_tl
4106 }
4107 \stex_mathml_intent:nn{
4108     #1[\prop_item:cn {l_stex_symdecl_#1 _prop}]{ args }(
4109     \seq_use:Nn \l__stex_terms_tmp_seq ,
4110 )
4111 ){
4112     \str_if_in:NnTF \l_tmpa_str b {
4113         \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
4114     }{
4115         \str_if_in:NnTF \l_tmpa_str B {
4116             \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
4117         }{
4118             \_stex_term_oma:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
4119         }
4120     }
4121 }
4122 }
4123 % TODO check that all arguments exist
4124 ){
4125     \_stex_reset:N \l_stex_argnames_seq
4126     \_stex_reset:N \STEXInternalCurrentSymbolStr
4127     \_stex_reset:N \arg
4128     \_stex_reset:N \comp
4129     \_stex_reset:N \l__stex_terms_custom_args_prop
4130     %\bool_set_true:N \l_stex_allow_semantic_bool
4131 }
4132 }
4133
4134 \NewDocumentCommand \__stex_terms_arg: { s O{} m}{
4135     \tl_if_empty:nTF {#2}{
4136         \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
4137         \bool_set_true:N \l_tmpa_bool
4138         \bool_do_while:Nn \l_tmpa_bool {
4139             \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int}
4140             \int_incr:N \l_tmpa_int
4141         }{
4142             \bool_set_false:N \l_tmpa_bool
4143         }
4144     }
4145     ){
4146         \int_set:Nn \l_tmpa_int { #2 }
4147     }
4148     \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
4149     \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
4150         \msg_error:nnxxx{stex}{error/overarity}
4151         {\int_use:N \l_tmpa_int}
4152         {\STEXInternalCurrentSymbolStr}
4153         {\str_count:N \l_tmpa_str}
4154     }
4155     \str_set:Nx \l_tmpa_str {\str_item:Nn \l_tmpa_str \l_tmpa_int}
4156     \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {

```

```

4157 \bool_lazy_any:nF {
4158   {\str_if_eq_p:Vn \l_tmpa_str {a}}
4159   {\str_if_eq_p:Vn \l_tmpa_str {B}}
4160 }{
4161   \msg_error:nnxx{stex}{error/doubleargument}
4162   {\int_use:N \l_tmpa_int}
4163   {\STEXInternalCurrentSymbolStr}
4164 }
4165 }
4166 \exp_args:NNx \prop_put:Nnn \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {\ign
4167 \bool_if:NTF \l_stex_allow_semantic_bool \use_i:nn {
4168   \bool_set_true:N \l_stex_allow_semantic_bool
4169   \use:nn
4170 }
4171 {
4172 \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq \l_tmpa_int}{
4173   \IfBooleanTF#1{
4174     \stex_annotate_invisible:n { %TODO
4175       \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
4176     }
4177   }{ %TODO
4178     \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
4179   }
4180 }}
4181 {\bool_set_false:N \l_stex_allow_semantic_bool}
4182 }
4183
4184
4185 \cs_new_protected:Nn \_stex_term_arg:nn {
4186   \bool_set_true:N \l_stex_allow_semantic_bool
4187   \stex_annotate:nnn{ arg }{ #1 }{ #2 }
4188   \bool_set_false:N \l_stex_allow_semantic_bool
4189 }
4190
4191 \cs_new_protected:Npn \STEXInternalTermMathArgiii #1#2#3 {
4192   \exp_args:Nnx \use:nn
4193   { \int_set:Nn \l__stex_terms_downprec { #2 }
4194     \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq \l_tmpa_int}{
4195       \_stex_term_arg:nn { #1 }{ #3 }
4196     }
4197   }
4198   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
4199 }

```

(End definition for `\stex_invoke_symbol:n`. This function is documented on page 92.)

`\STEXInternalTermMathAssocArgiiii`

```

4200 \cs_new_protected:Npn \STEXInternalTermMathAssocArgiiii #1#2#3#4#5 {
4201   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
4202   \tl_set:Nn \l_tmpb_tl {\STEXInternalTermMathArgiii{#5#1}{#2}}
4203   \tl_if_empty:nTF { #3 }{
4204     \STEXInternalTermMathArgiii{#5#1}{#2}{}
4205   }{
4206     \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 } }{

```

```

4207 \expandafter\if\expandafter\relax\noexpand#3
4208 \tl_set:Nn \l_tmpa_tl {\_stex_terms_math_assoc_arg_maybe_sequence:Nnn#3{#1}{#5}}
4209 \else
4210 \tl_set:Nn \l_tmpa_tl {\_stex_terms_math_assoc_arg_simple:nnn{#1}{#3}{#5}}
4211 \fi
4212 \l_tmpa_tl
4213 }{
4214 \_stex_terms_math_assoc_arg_simple:nnn{#1}{#3}{#5}
4215 }
4216 }
4217 }
4218
4219 \cs_new_protected:Nn \_stex_terms_math_assoc_arg_maybe_sequence:Nnn {
4220 \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
4221 \str_if_empty:NTF \l_tmpa_str {
4222 \exp_args:Nx \cs_if_eq:NNTF {
4223 \tl_head:N #1
4224 } \stex_invoke_sequence:n {
4225 \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
4226 \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
4227 \tl_set:Nx \l_tmpa_tl {\prop_item:cn {l_stex_symdecl_varseq://\l_tmpa_str _prop}{notat
4228 \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
4229 \tl_set:Nx \l_tmpa_tl {\exp_not:N \exp_not:n{
4230 \exp_not:n{\exp_args:Nnx \use:nn} {
4231 \exp_not:n {
4232 \def\comp{\_varcomp}
4233 \str_set:Nn \STEXInternalCurrentSymbolStr
4234 } {varseq://\l_tmpa_str}
4235 \exp_not:n{ ##1 }
4236 }{
4237 \exp_not:n {
4238 \_stex_reset:N \comp
4239 \_stex_reset:N \STEXInternalCurrentSymbolStr
4240 }
4241 }
4242 }}}
4243 \exp_args:Nno \use:n {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
4244 \seq_reverse:N \l_tmpa_seq
4245 \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
4246 \seq_map_inline:Nn \l_tmpa_seq {
4247 \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
4248 \exp_args:Nno
4249 \l_tmpa_cs { ##1 } \l_tmpa_tl
4250 }
4251 }
4252 \tl_set:Nx \l_tmpa_tl {
4253 \_stex_term_omv:nn {varseq://\l_tmpa_str}{
4254 \exp_args:No \exp_not:n \l_tmpa_tl
4255 }
4256 }
4257 \exp_args:No\l_tmpb_tl\l_tmpa_tl
4258 }{
4259 \_stex_terms_math_assoc_arg_simple:nnn{#2} { #1 }{#3}
4260 }

```

```

4261 } {
4262   \_stex_terms_math_assoc_arg_simple:nnn{#2} { #1 }{#3}
4263 }
4264
4265 }
4266
4267 \cs_new_protected:Nn \_stex_terms_math_assoc_arg_simple:nnn {
4268   \clist_set:Nn \l_tmpa_clist{ #2 }
4269   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
4270     \tl_set:Nn \l_tmpa_tl {
4271       \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq #1}{
4272         \_stex_term_arg:nn{A#3#1}{ #2 } }
4273     }
4274   }{
4275     \clist_reverse:N \l_tmpa_clist
4276     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
4277     \tl_set:Nx \l_tmpa_tl {
4278       \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq #1}{
4279         \_stex_term_arg:nn{A#3#1}{
4280           \exp_args:No \exp_not:n \l_tmpa_tl
4281         }
4282       }
4283     }
4284     \clist_map_inline:Nn \l_tmpa_clist {
4285       \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
4286         \l_tmpa_cs {
4287           \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq #1}{
4288             \_stex_term_arg:nn{A#3#1}{##1}
4289           }
4290         } \l_tmpa_tl
4291       }
4292     }
4293   }
4294   \exp_args:No\l_tmpb_tl\l_tmpa_tl
4295 }

```

(End definition for `\STEXInternalTermMathAssocArgiiii`. This function is documented on page 93.)

## 32.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
4296 \tl_const:Nx \infprec {\int_use:N \c_max_int}
4297 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
4298 \int_new:N \l__stex_terms_downprec
4299 \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 93.)

Bracketing:

```

\l__stex_terms_left_bracket_str
\l__stex_terms_right_bracket_str
4300 \tl_set:Nn \l__stex_terms_left_bracket_str (
4301 \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

```

4302 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
4303   \bool_if:NTF \l__stex_terms_brackets_done_bool {
4304     \bool_set_false:N \l__stex_terms_brackets_done_bool
4305     #2
4306   } {
4307     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
4308       \bool_if:NTF \l__stex_inarray_bool { #2 }{
4309         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
4310         \dobrackets { #2 }
4311       }
4312     }{ #2 }
4313   }
4314 }

```

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

**`\dobrackets`**

```

4315 \bool_new:N \l__stex_terms_brackets_done_bool
4316 %\RequirePackage{scalerel}
4317 \cs_new_protected:Npn \dobrackets #1 {
4318   %\ThisStyle{\if D\m@switch
4319   %   \exp_args:Nnx \use:nn
4320   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
4321   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
4322   %   \else
4323   %     \exp_args:Nnx \use:nn
4324   %     {
4325   %       \bool_set_true:N \l__stex_terms_brackets_done_bool
4326   %       \int_set:Nn \l__stex_terms_downprec \infpref
4327   %       \l__stex_terms_left_bracket_str
4328   %       #1
4329   %     }
4330   %     {
4331   %       \bool_set_false:N \l__stex_terms_brackets_done_bool
4332   %       \l__stex_terms_right_bracket_str
4333   %       \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
4334   %     }
4335   %\fi}
4336 }

```

(End definition for `\dobrackets`. This function is documented on page 93.)

**`\withbrackets`**

```

4337 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
4338   \exp_args:Nnx \use:nn
4339   {
4340     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
4341     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
4342     #3
4343   }
4344   {

```

```

4345 \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
4346 {\l__stex_terms_left_bracket_str}
4347 \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
4348 {\l__stex_terms_right_bracket_str}
4349 }
4350 }

```

(End definition for `\withbrackets`. This function is documented on page 93.)

## `\STEXinvisible`

```

4351 \cs_new_protected:Npn \STEXinvisible #1 {
4352   \stex_annotate_invisible:n { #1 }
4353 }

```

(End definition for `\STEXinvisible`. This function is documented on page 93.)

OMDoc terms:

## `\STEXInternalTermMathOMSiiii`

```

4354 \cs_new_protected:Nn \_stex_term_oms:nnn {
4355   \stex_annotate:nnn{ OMID }{ #2 }{
4356     #3
4357   }
4358 }
4359
4360 \cs_new_protected:Npn \STEXInternalTermMathOMSiiii #1#2#3#4 {
4361   \__stex_terms_maybe_brackets:nn { #3 }{
4362     \stex_mathml_intent:nn{#1} {
4363       \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
4364     }
4365   }
4366 }

```

(End definition for `\STEXInternalTermMathOMSiiii`. This function is documented on page 92.)

## `\_stex_term_math_omv:nn`

```

4367 \cs_new_protected:Nn \_stex_term_omv:nn {
4368   \stex_annotate:nnn{ OMV }{ #1 }{
4369     #2
4370   }
4371 }

```

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

## `\STEXInternalTermMathOMAiiii`

```

4372 \cs_new_protected:Nn \_stex_term_oma:nnn {
4373   \stex_annotate:nnn{ OMA }{ #2 }{
4374     #3
4375   }
4376 }
4377
4378 \cs_new_protected:Npn \STEXInternalTermMathOMAiiii #1#2#3#4 {
4379   \exp_args:Nnx \use:nn {
4380     \seq_clear:N \l__stex_terms_tmp_seq
4381     \prop_if_exist:cT{l_stex_symdecl_#1 _prop}{
4382       \exp_args:NNx \seq_set_from_clist:Nn \l_stex_argnames_seq {

```



```

4383     \prop_item:cn {l_stex_symdecl_#1 _prop}{argnames}
4384   }
4385   \exp_args:Nx\int_step_inline:nn{\prop_item:cn{l_stex_symdecl_#1 _prop}{arity}}{
4386     \tl_set:Nx \l__stex_terms_tmp_tl {\seq_item:Nn \l_stex_argnames_seq {##1}}
4387     \bool_lazy_or:nnT{
4388       \str_if_eq_p:nn{a}{\str_item:Nn\l_tmpa_str{##1}}
4389     }{
4390       \str_if_eq_p:nn{B}{\str_item:Nn\l_tmpa_str{##1}}
4391     }{
4392       \tl_put_right:Nn \l__stex_terms_tmp_tl +
4393     }
4394     \seq_put_right:No \l__stex_terms_tmp_seq \l__stex_terms_tmp_tl
4395   }
4396 }
4397 \__stex_terms_maybe_brackets:nn { #3 }{
4398   \stex_mathml_intent:nn{
4399     #1[\prop_item:cn {l_stex_symdecl_#1 _prop}{ args }](
4400       \seq_use:Nn \l__stex_terms_tmp_seq ,
4401     )
4402   }{
4403     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
4404   }
4405 }
4406 }{
4407   \_stex_reset:N \l_stex_argnames_seq
4408 }
4409 }

```

(End definition for `\STEXInternalTermMathOMAi`. This function is documented on page 92.)

`\STEXInternalTermMathOMBiiii`

```

4410 \cs_new_protected:Nn \_stex_term_ombind:nnn {
4411   \stex_annotate:nnn{ OMBIND }{ #2 }{
4412     #3
4413   }
4414 }
4415
4416 \cs_new_protected:Npn \STEXInternalTermMathOMBiiii #1#2#3#4 {
4417   \exp_args:Nnx \use:nn {
4418     \seq_clear:N \l__stex_terms_tmp_seq
4419     \prop_if_exist:cT{l_stex_symdecl_#1 _prop}{
4420       \exp_args:NNx \seq_set_from_clist:Nn \l_stex_argnames_seq {
4421         \prop_item:cn {l_stex_symdecl_#1 _prop}{argnames}
4422       }
4423       \exp_args:Nx\int_step_inline:nn{\prop_item:cn{l_stex_symdecl_#1 _prop}{arity}}{
4424         \tl_set:Nx \l__stex_terms_tmp_tl {\seq_item:Nn \l_stex_argnames_seq {##1}}
4425         \bool_lazy_or:nnT{
4426           \str_if_eq_p:nn{a}{\str_item:Nn\l_tmpa_str{##1}}
4427         }{
4428           \str_if_eq_p:nn{B}{\str_item:Nn\l_tmpa_str{##1}}
4429         }{
4430           \tl_put_right:Nn \l__stex_terms_tmp_tl +
4431         }
4432         \seq_put_right:No \l__stex_terms_tmp_seq \l__stex_terms_tmp_tl

```

```

4433     }
4434   }
4435   \__stex_terms_maybe_brackets:nn { #3 }{
4436     \stex_mathml_intent:nn{
4437       #1[\prop_item:cn {l_stex_symdecl_#1 _prop}{ args }](
4438         \seq_use:Nn \l__stex_terms_tmp_seq ,
4439       )
4440     }{
4441       \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
4442     }
4443   }
4444   }{
4445     \stex_reset:N \l_stex_argnames_seq
4446   }
4447 }

```

(End definition for `\STEXInternalTermMathOMBiiii`. This function is documented on page 92.)

`\symref`  
`\symname`

```

4448 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
4449
4450 \keys_define:nn { stex / symname } {
4451   pre      .tl_set_x:N      = \l__stex_terms_pre_tl ,
4452   post     .tl_set_x:N      = \l__stex_terms_post_tl ,
4453   root     .tl_set_x:N      = \l__stex_terms_root_tl
4454 }
4455
4456 \cs_new_protected:Nn \stex_symname_args:n {
4457   \tl_clear:N \l__stex_terms_post_tl
4458   \tl_clear:N \l__stex_terms_pre_tl
4459   \tl_clear:N \l__stex_terms_root_str
4460   \keys_set:nn { stex / symname } { #1 }
4461 }
4462
4463 \NewDocumentCommand \symref { m m }{
4464   \let\compemph_uri_prev:\compemph@uri
4465   \let\compemph@uri\symrefemph@uri
4466   \STEXsymbol{#1}!\{ #2 }
4467   \let\compemph@uri\compemph_uri_prev:
4468 }
4469
4470 \NewDocumentCommand \synonym { 0{} m m }{
4471   \stex_symname_args:n { #1 }
4472   \let\compemph_uri_prev:\compemph@uri
4473   \let\compemph@uri\symrefemph@uri
4474   % TODO
4475   \STEXsymbol{#2}!\{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
4476   \let\compemph@uri\compemph_uri_prev:
4477 }
4478
4479 \NewDocumentCommand \symname { 0{} m }{
4480   \stex_symname_args:n { #1 }
4481   \stex_get_symbol:n { #2 }
4482   \str_set:Nx \l_tmpa_str {

```

```

4483 \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4484 }
4485 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4486
4487 \let\compemph_uri_prev:\compemph@uri
4488 \let\compemph@uri\symrefemph@uri
4489 \exp_args:NNx \use:nn
4490 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmode*\fi{
4491 \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
4492 } }
4493 \let\compemph@uri\compemph_uri_prev:
4494 }
4495
4496 \NewDocumentCommand \Symname { 0{} m }{
4497 \stex_symname_args:n { #1 }
4498 \stex_get_symbol:n { #2 }
4499 \str_set:Nx \l_tmpa_str {
4500 \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4501 }
4502 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4503 \let\compemph_uri_prev:\compemph@uri
4504 \let\compemph@uri\symrefemph@uri
4505 \exp_args:NNx \use:nn
4506 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmode*\fi{
4507 \exp_after:wN \stex_capitalize:n \l_tmpa_str
4508 \l__stex_terms_post_tl
4509 } }
4510 \let\compemph@uri\compemph_uri_prev:
4511 }

```

(End definition for `\symref` and `\symname`. These functions are documented on page 92.)

## 32.3 Notation Components

```

4512 <@@=stex_notationcomps>
\comp
\compemph@uri
\compemph
\defemph
\defemph@uri
\symrefemph
\symrefemph@uri
\varemp
\varemp@uri
4513 \cs_new_protected:Npn \_comp #1 {
4514 \str_if_empty:NF \STEXInternalCurrentSymbolStr {
4515 \stex_html_backend:TF {
4516 \stex_annotate:nnn { comp }{ \STEXInternalCurrentSymbolStr }{ #1 }
4517 }{
4518 \exp_args:Nnx \compemph@uri { #1 } { \STEXInternalCurrentSymbolStr }
4519 }
4520 }
4521 }
4522
4523 \cs_new_protected:Npn \_varcomp #1 {
4524 \str_if_empty:NF \STEXInternalCurrentSymbolStr {
4525 \stex_html_backend:TF {
4526 \stex_annotate:nnn { varcomp }{ \STEXInternalCurrentSymbolStr }{ #1 }
4527 }{
4528 \exp_args:Nnx \varemp@uri { #1 } { \STEXInternalCurrentSymbolStr }
4529 }

```

```

4530 }
4531 }
4532
4533 \def\comp{\_comp}
4534
4535 \cs_new_protected:Npn \compemph@uri #1 #2 {
4536   \compemph{ #1 }
4537 }
4538
4539
4540 \cs_new_protected:Npn \compemph #1 {
4541   #1
4542 }
4543
4544 \cs_new_protected:Npn \defemph@uri #1 #2 {
4545   \defemph{#1}
4546 }
4547
4548 \cs_new_protected:Npn \defemph #1 {
4549   \textbf{#1}
4550 }
4551
4552 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
4553   \symrefemph{#1}
4554 }
4555
4556 \cs_new_protected:Npn \symrefemph #1 {
4557   \emph{#1}
4558 }
4559
4560 \cs_new_protected:Npn \varemp@uri #1 #2 {
4561   \varemp{#1}
4562 }
4563
4564 \cs_new_protected:Npn \varemp #1 {
4565   #1
4566 }

```

(End definition for `\comp` and others. These functions are documented on page 93.)

## `\ellipses`

```

4567 \NewDocumentCommand \ellipses {} { \ldots }

```

(End definition for `\ellipses`. This function is documented on page 93.)

```

\parray
\prmatrix 4568 \bool_new:N \l_stex_inparray_bool
\parrayline 4569 \bool_set_false:N \l_stex_inparray_bool
\parraylineh 4570 \NewDocumentCommand \parray { m m } {
\parraycell 4571   \begingroup
4572   \bool_set_true:N \l_stex_inparray_bool
4573   \begin{array}{#1}
4574     #2
4575   \end{array}
4576   \endgroup

```

```

4577 }
4578
4579 \NewDocumentCommand \prmatrix { m } {
4580   \begingroup
4581   \bool_set_true:N \l_stex_inarray_bool
4582   \begin{matrix}
4583     #1
4584   \end{matrix}
4585   \endgroup
4586 }
4587
4588 \def \maybepline {
4589   \bool_if:NT \l_stex_inarray_bool {\hline}
4590 }
4591
4592 \def \parrayline #1 #2 {
4593   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
4594 }
4595
4596 \def \pmrow #1 { \parrayline{}{ #1 } }
4597
4598 \def \parraylineh #1 #2 {
4599   #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
4600 }
4601
4602 \def \parraycell #1 {
4603   #1 \bool_if:NT \l_stex_inarray_bool {&}
4604 }

```

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

## 32.4 Variables

```

4605 <@@=stex_variables>

```

\stex\_invoke\_variable:n Invokes a variable

```

4606 \cs_new_protected:Nn \stex_invoke_variable:n {
4607   \if_mode_math:
4608     \exp_after:wN \__stex_variables_invoke_math:n
4609   \else:
4610     \exp_after:wN \__stex_variables_invoke_text:n
4611   \fi: {#1}
4612 }
4613
4614 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
4615   \peek_charcode_remove:NTF ! {
4616     \__stex_variables_invoke_op_custom:nn {#1}
4617   }{
4618     \__stex_variables_invoke_custom:nn {#1}
4619   }
4620 }
4621
4622
4623 \cs_new_protected:Nn \__stex_variables_invoke_math:n {

```

```

4624 \peek_charcode_remove:NTF ! {
4625   \peek_charcode_remove:NTF ! {
4626     \peek_charcode:NTF [ {
4627       % TODO throw error
4628     }{
4629       \__stex_variables_invoke_op_custom:nn
4630     }
4631   }{
4632     \__stex_variables_invoke_op:n { #1 }
4633   }
4634 }{
4635   \peek_charcode_remove:NTF * {
4636     \__stex_variables_invoke_custom:nn { #1 }
4637   }{
4638     \__stex_variables_invoke_math_ii:n { #1 }
4639   }
4640 }
4641 }
4642
4643 \cs_new_protected:Nn \__stex_variables_invoke_op_custom:nn {
4644   \exp_args:Nnx \use:nn {
4645     \def\comp{\_varcomp}
4646     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4647     \bool_set_false:N \l_stex_allow_semantic_bool
4648     \stex_term_omv:nn {var://#1}{
4649       \comp{ #2 }
4650     }
4651   }{
4652     \stex_reset:N \comp
4653     \stex_reset:N \STEXInternalCurrentSymbolStr
4654     \bool_set_true:N \l_stex_allow_semantic_bool
4655   }
4656 }
4657
4658 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
4659   \cs_if_exist:cTF {
4660     stex_var_op_notation_ #1 _cs
4661   }{
4662     \exp_args:Nnx \use:nn {
4663       \def\comp{\_varcomp}
4664       \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4665       \stex_term_omv:nn { var://#1 }{
4666         \use:c{stex_var_op_notation_ #1 _cs }
4667       }
4668     }{
4669       \stex_reset:N \comp
4670       \stex_reset:N \STEXInternalCurrentSymbolStr
4671     }
4672   }{
4673     \int_compare:nNnTF {\prop_item:cn {l_stex_symdecl_var://#1_prop}{arity}} = 0{
4674       \__stex_variables_invoke_math_ii:n {#1}
4675     }{
4676       \msg_error:nxxx{stex}{error/noop}{variable~#1}{}
4677     }

```

```

4678 }
4679 }
4680
4681 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n #1 {
4682   \cs_if_exist:cTF {
4683     stex_var_notation_#1_cs
4684   }{
4685     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4686       \_stex_reset:N \comp
4687       \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
4688       \_stex_reset:N \STEXInternalCurrentSymbolStr
4689       \bool_set_true:N \l_stex_allow_semantic_bool
4690     }
4691     \def\comp{\_varcomp}
4692     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4693     \bool_set_false:N \l_stex_allow_semantic_bool
4694     \use:c{stex_var_notation_#1_cs}
4695   }{
4696     \msg_error:nnxx{stex}{error/nonotation}{variable-#1}{s}
4697   }
4698 }
4699
4700 \cs_new_protected:Nn \__stex_variables_invoke_custom:nn {
4701   \exp_args:Nnx \use:nn {
4702     \def\comp{\_varcomp}
4703     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4704     \prop_clear:N \l__stex_terms_custom_args_prop
4705     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
4706     \prop_get:cnN {
4707       l_stex_symdecl_var://#1 _prop
4708     }{ args } \l_tmpa_str
4709     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
4710     \tl_set:Nn \arg { \_stex_terms_arg: }
4711     \str_if_empty:NTF \l_tmpa_str {
4712       \_stex_term_omv:nn {var://#1}{\ignorespaces#2}
4713     }{
4714       \str_if_in:NnTF \l_tmpa_str b {
4715         \_stex_term_ombind:nnn {var://#1}{\ignorespaces#2}
4716       }{
4717         \str_if_in:NnTF \l_tmpa_str B {
4718           \_stex_term_ombind:nnn {var://#1}{\ignorespaces#2}
4719         }{
4720           \_stex_term_oma:nnn {var://#1}{\ignorespaces#2}
4721         }
4722       }
4723     }
4724     % TODO check that all arguments exist
4725   }{
4726     \_stex_reset:N \STEXInternalCurrentSymbolStr
4727     \_stex_reset:N \arg
4728     \_stex_reset:N \comp
4729     \_stex_reset:N \l__stex_terms_custom_args_prop
4730     %\bool_set_true:N \l_stex_allow_semantic_bool
4731   }

```

```
4732 }
```

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

## 32.5 Sequences

```
4733 <@@=stex_sequences>
4734
4735 \cs_new_protected:Nn \stex_invoke_sequence:n {
4736   \peek_charcode_remove:NTF ! {
4737     \stex_term_omv:nn {varseq://#1}{
4738       \exp_args:Nnx \use:nn {
4739         \def\comp{\_varcomp}
4740         \str_set:Nn \STEXInternalCurrentSymbolStr {varseq://#1}
4741         \prop_item:cn{l_stex_symdecl_varseq://#1_prop}{notation}
4742       }{
4743         \stex_reset:N \comp
4744         \stex_reset:N \STEXInternalCurrentSymbolStr
4745       }
4746     }
4747   }{
4748     \bool_set_false:N \l_stex_allow_semantic_bool
4749     \def\comp{\_varcomp}
4750     \str_set:Nn \STEXInternalCurrentSymbolStr {varseq://#1}
4751     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4752       \stex_reset:N \comp
4753       \stex_reset:N \STEXInternalSymbolAfterInvokationTL
4754       \stex_reset:N \STEXInternalCurrentSymbolStr
4755       \bool_set_true:N \l_stex_allow_semantic_bool
4756     }
4757     \use:c { stex_varseq_#1_cs }
4758   }
4759 }
4760 </package>
```



## Chapter 33

# STEX -Structural Features Implementation

```
4761 <*package>
4762
4763 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4764
4765 Warnings and error messages
4766 \msg_new:nnn{stex}{error/copymodule/notallowed}{
4767   Symbol~#1~can~not~be~assigned~in~copymodule~#2
4768 }
4769 \msg_new:nnn{stex}{error/interpretmodule/nodfiniens}{
4770   Symbol~#1~not~assigned~in~interpretmodule~#2
4771 }
4772 \msg_new:nnn{stex}{error/unknownstructure}{
4773   No~structure~#1~found!
4774 }
4775
4776 \msg_new:nnn{stex}{error/unknownfield}{
4777   No~field~#1~in~instance~#2~found!\#3
4778 }
4779
4780 \msg_new:nnn{stex}{error/keyval}{
4781   Invalid~key=value~pair~#1
4782 }
4783 \msg_new:nnn{stex}{error/instantiate/missing}{
4784   Assignments~missing~in~instantiate:~#1
4785 }
4786 \msg_new:nnn{stex}{error/incompatible}{
4787   Incompatible~signature:~#1~(#2)~and~#3~(#4)
4788 }
4789
```

## 33.1 Imports with modification

```

4790 <@@=stex_copymodule>
4791 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
4792   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4793     \tl_set:Nn \l_tmpa_tl { #1 }
4794     \__stex_copymodule_get_symbol_from_cs:
4795   }{
4796     % argument is a string
4797     % is it a command name?
4798     \cs_if_exist:cTF { #1 }{
4799       \cs_set_eq:Nc \l_tmpa_tl { #1 }
4800       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4801       \str_if_empty:NNTF \l_tmpa_str {
4802         \exp_args:Nx \cs_if_eq:NNTF {
4803           \tl_head:N \l_tmpa_tl
4804         } \stex_invoke_symbol:n {
4805           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
4806         }{
4807           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4808         }
4809       } {
4810         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4811       }
4812     }{
4813       % argument is not a command name
4814       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4815       % \l_stex_all_symbols_seq
4816     }
4817   }
4818 }
4819
4820 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
4821   \str_set:Nn \l_tmpa_str { #1 }
4822   \bool_set_false:N \l_tmpa_bool
4823   \bool_if:NF \l_tmpa_bool {
4824     \tl_set:Nn \l_tmpa_tl {
4825       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
4826     }
4827     \str_set:Nn \l_tmpa_str { #1 }
4828     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4829     \seq_map_inline:Nn #2 {
4830       \str_set:Nn \l_tmpb_str { ##1 }
4831       \str_if_eq:eeT { \l_tmpa_str } {
4832         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4833       } {
4834         \seq_map_break:n {
4835           \tl_set:Nn \l_tmpa_tl {
4836             \str_set:Nn \l_stex_get_symbol_uri_str {
4837               ##1
4838             }
4839           }
4840         }
4841       }

```

```

4842     }
4843     \l_tmpa_tl
4844   }
4845 }
4846
4847 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
4848   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4849     { \tl_tail:N \l_tmpa_tl }
4850   \tl_if_single:NTF \l_tmpa_tl {
4851     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
4852       \exp_after:wN \str_set:Nn \exp_after:wN
4853         \l_stex_get_symbol_uri_str \l_tmpa_tl
4854       \__stex_copymodule_get_symbol_check:n { #1 }
4855     }{
4856       % TODO
4857       % tail is not a single group
4858     }
4859   }{
4860     % TODO
4861     % tail is not a single group
4862   }
4863 }
4864
4865 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
4866   \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
4867     \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
4868       :~\seq_use:Nn #1 {,~}
4869     }
4870   }
4871 }
4872
4873 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
4874   % import module
4875   \stex_import_module_uri:nn { #1 } { #2 }
4876   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
4877   \stex_import_require_module:nnnn
4878     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4879     { \l_stex_import_path_str } { \l_stex_import_name_str }
4880
4881   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
4882   \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
4883
4884   % fields
4885   \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
4886   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4887     \seq_map_inline:cn {c_stex_module_##1_constants}{
4888       \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
4889         ##1 ? #####1
4890       }
4891     }
4892   }
4893
4894   % setup prop
4895   \seq_clear:N \l_tmpa_seq

```

```

4896 \exp_args:Nn \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
4897   name      = \l_stex_current_copymodule_name_str ,
4898   module    = \l_stex_current_module_str ,
4899   from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
4900   includes  = \l_tmpa_seq %,
4901 % fields    = \l_tmpa_seq
4902 }
4903 \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
4904   as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
4905 \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_seq {
4906 \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
4907
4908 \stex_if_do_html:T {
4909   \begin{stex_annotate_env} {#4} {
4910     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4911   }
4912   \stex_annotate_invisible:nnn{domain}{\l_stex_import_ns_str ?\l_stex_import_name_str}{}
4913 }
4914 }
4915
4916 \cs_new_protected:Nn \stex_copymodule_end:n {
4917   % apply to every field
4918   \def \l_tmpa_cs ##1 ##2 {#1}
4919
4920   \tl_clear:N \__stex_copymodule_module_tl
4921   \tl_clear:N \__stex_copymodule_exec_tl
4922
4923   %\prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
4924   \seq_clear:N \__stex_copymodule_fields_seq
4925
4926   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4927     \seq_map_inline:cn {c_stex_module_##1_constants}{
4928
4929       \tl_clear:N \__stex_copymodule_curr_symbol_tl % <- wrap in current symbol html
4930       \l_tmpa_cs{##1}{####1}
4931
4932       \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
4933         \str_set_eq:Nc \__stex_copymodule_curr_name_str {l__stex_copymodule_copymodule_##1?#
4934         \stex_if_do_html:T {
4935           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4936             \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?###
4937           }
4938         }
4939       }{
4940         \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str /
4941       }
4942
4943       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
4944       \prop_put:Nnx \l_tmpa_prop { name } \__stex_copymodule_curr_name_str
4945       \prop_put:Nnx \l_tmpa_prop { module } \l_stex_current_module_str
4946
4947       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4948         \stex_if_do_html:T {
4949           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {

```

```

4950         $\stex_annotate_invisible:nnn{definiens}{\exp_after:wN \exp_not:N\csname l__st
4951     }
4952 }
4953 \prop_put:Nnn \l_tmpa_prop { defined } { true }
4954 }
4955
4956 \stex_add_constant_to_current_module:n \__stex_copymodule_curr_name_str
4957 \tl_put_right:Nx \__stex_copymodule_module_tl {
4958     \seq_clear:c {l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_n
4959     \prop_set_from_keyval:cn {
4960         l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_name_str _prop
4961     }{
4962         \prop_to_keyval:N \l_tmpa_prop
4963     }
4964 }
4965
4966 \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
4967     \stex_if_do_html:T {
4968         \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4969             \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1
4970         }
4971     }
4972     \tl_put_right:Nx \__stex_copymodule_module_tl {
4973         \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
4974             \stex_invoke_symbol:n {
4975                 \l_stex_current_module_str ? \__stex_copymodule_curr_name_str
4976             }
4977         }
4978     }
4979 }
4980
4981 \seq_put_right:Nx \__stex_copymodule_fields_seq {\l_stex_current_module_str ? \__stex_
4982
4983 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4984     \stex_copy_notations:nn {\l_stex_current_module_str ? \__stex_copymodule_curr_name_s
4985 }
4986
4987 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4988     \stex_if_do_html:TF{
4989         \stex_annotate_invisible:nnn{assignment} {##1?####1} { \exp_after:wN \exp_not:n \e
4990     }{
4991         \exp_after:wN \exp_not:n \exp_after:wN {\__stex_copymodule_curr_symbol_tl}
4992     }
4993 }
4994 }
4995 }
4996
4997
4998 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \__stex_copymodule_fields_seq
4999 \tl_put_left:Nx \__stex_copymodule_module_tl {
5000     \prop_set_from_keyval:cn {
5001         l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _pro
5002     }{
5003         \prop_to_keyval:N \l_stex_current_copymodule_prop

```

```

5004     }
5005   }
5006
5007   \seq_gput_right:cx{c_stex_module_\l_stex_current_module_str _copymodules}{
5008     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
5009   }
5010
5011   \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
5012   \stex_debug:nn{copymodule}{result:\meaning \__stex_copymodule_module_tl}
5013   \stex_debug:nn{copymodule}{output:\meaning \__stex_copymodule_exec_tl}
5014
5015   \__stex_copymodule_exec_tl
5016   \stex_if_do_html:T {
5017     \end{stex_annotate_env}
5018   }
5019 }
5020
5021 \NewDocumentEnvironment {copymodule} { 0{} m m}{
5022   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ copymodule }
5023   \stex_deactivate_macro:Nn \symdecl {module~environments}
5024   \stex_deactivate_macro:Nn \symdef {module~environments}
5025   \stex_deactivate_macro:Nn \notation {module~environments}
5026   \stex_reactivate_macro:N \assign
5027   \stex_reactivate_macro:N \renamedekl
5028   \stex_reactivate_macro:N \donotcopy
5029   \stex_smsmode_do:
5030 }{
5031   \stex_copymodule_end:n {}
5032 }
5033
5034 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
5035   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ interpretmodule }
5036   \stex_deactivate_macro:Nn \symdecl {module~environments}
5037   \stex_deactivate_macro:Nn \symdef {module~environments}
5038   \stex_deactivate_macro:Nn \notation {module~environments}
5039   \stex_reactivate_macro:N \assign
5040   \stex_reactivate_macro:N \renamedekl
5041   \stex_reactivate_macro:N \donotcopy
5042   \stex_smsmode_do:
5043 }{
5044   \stex_copymodule_end:n {
5045     \tl_if_exist:cF {
5046       l__stex_copymodule_copymodule_##1?##2_def_tl
5047     }{
5048       \str_if_eq:eeF {
5049         \prop_item:cn{
5050           l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
5051         }{ true }{
5052           \msg_error:nxxx{stex}{error/interpretmodule/nodéfiniens}{
5053             ##1?##2
5054           }{\l_stex_current_copymodule_name_str}
5055         }
5056       }
5057     }

```

```

5058 }
5059
5060 \iffalse \begin{stex_annotate_env} \fi
5061 \NewDocumentEnvironment {realization} { 0{} m}{
5062   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #2 }{ realize }
5063   \stex_deactivate_macro:Nn \symdecl {module~environments}
5064   \stex_deactivate_macro:Nn \symdef {module~environments}
5065   \stex_deactivate_macro:Nn \notation {module~environments}
5066   \stex_reactivate_macro:N \donotcopy
5067   \stex_reactivate_macro:N \assign
5068   \stex_smsmode_do:
5069 }{
5070   \stex_import_module_uri:nn { #1 } { #2 }
5071   \tl_clear:N \__stex_copymodule_exec_tl
5072   \tl_set:Nx \__stex_copymodule_module_tl {
5073     \stex_import_require_module:nnnn
5074     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
5075     { \l_stex_import_path_str } { \l_stex_import_name_str }
5076   }
5077   \exp_args:Nx \stex_add_import_to_current_module:n{
5078     \l_stex_import_ns_str ? \l_stex_import_name_str
5079   }
5080
5081   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
5082     \seq_map_inline:cn {c_stex_module_##1_constants}{
5083       \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str / #
5084       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?###1_def_tl}{
5085         \stex_if_do_html:T {
5086           \tl_put_right:Nx \__stex_copymodule_exec_tl {
5087             \stex_annotate_invisible:nnn{assignment} {##1?###1} {
5088               $\stex_annotate_invisible:nnn{definiens}{}{\exp_after:wN \exp_not:N\csname l__
5089             }
5090           }
5091         }
5092         \tl_put_right:Nx \__stex_copymodule_module_tl {
5093           \prop_put:cnn {l_stex_symdecl_##1?###1_prop}{ defined }{ true }
5094         }
5095       }
5096     }}
5097
5098   \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
5099
5100   \__stex_copymodule_exec_tl
5101   \stex_if_do_html:T {\end{stex_annotate_env}}
5102 }
5103
5104 \NewDocumentCommand \donotcopy { m }{
5105   \str_clear:N \l_stex_import_name_str
5106   \str_set:Nn \l_tmpa_str { #1 }
5107   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
5108   \seq_map_inline:Nn \l_stex_all_modules_seq {
5109     \str_set:Nn \l_tmpb_str { ##1 }
5110     \str_if_eq:eeT { \l_tmpa_str } {
5111       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }

```

```

5112 } {
5113   \seq_map_break:n {
5114     \stex_if_do_html:T {
5115       \stex_if_smsmode:F {
5116         \stex_annotate_invisible:nnn{donotcopy}{##1}{
5117           \stex_annotate:nnn{domain}{##1}{}}
5118       }
5119     }
5120   }
5121   \str_set_eq:NN \l_stex_import_name_str \l_tmpb_str
5122 }
5123 }
5124 \seq_map_inline:cn {c_stex_module_###1_copymodules}{
5125   \str_set:Nn \l_tmpb_str { ###1 }
5126   \str_if_eq:eeT { \l_tmpa_str } {
5127     \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
5128   } {
5129     \seq_map_break:n {\seq_map_break:n {
5130       \stex_if_do_html:T {
5131         \stex_if_smsmode:F {
5132           \stex_annotate_invisible:nnn{donotcopy}{####1}{
5133             \stex_annotate:nnn{domain}{
5134               \prop_item:cn {l_stex_copymodule_ ####1 _prop}{module}
5135             }{}}
5136         }
5137       }
5138     }
5139     \str_set:Nx \l_stex_import_name_str {
5140       \prop_item:cn {l_stex_copymodule_ ####1 _prop}{module}
5141     }
5142   }}
5143 }
5144 }
5145 }
5146 \str_if_empty:NTF \l_stex_import_name_str {
5147   % TODO throw error
5148 }{
5149   \stex_collect_imports:n {\l_stex_import_name_str }
5150   \seq_map_inline:Nn \l_stex_collect_imports_seq {
5151     \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
5152     \seq_map_inline:cn {c_stex_module_###1_constants}{
5153       \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? ####1 }
5154       \bool_lazy_any:nT {
5155         { \cs_if_exist_p:c {l__stex_copymodule_copymodule_###1?####1_name_str}}
5156         { \cs_if_exist_p:c {l__stex_copymodule_copymodule_###1?####1_macroname_str}}
5157         { \cs_if_exist_p:c {l__stex_copymodule_copymodule_###1?####1_def_tl}}
5158       }{
5159         % TODO throw error
5160       }
5161     }
5162   }
5163   \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
5164   \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_name_str }
5165   \prop_put:Nno \l_stex_current_copymodule_prop {includes} \l_tmpa_seq

```



```

5166 }
5167 \stex_smsmode_do:
5168 }
5169
5170 \NewDocumentCommand \assign { m m }{
5171   \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
5172   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
5173   \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _def_tl}{#2}
5174   \stex_smsmode_do:
5175 }
5176
5177 \keys_define:nn { stex / renamedec1 } {
5178   name          .str_set_x:N = \l_stex_renamedec1_name_str
5179 }
5180 \cs_new_protected:Nn \__stex_copymodule_renamedec1_args:n {
5181   \str_clear:N \l_stex_renamedec1_name_str
5182   \keys_set:nn { stex / renamedec1 } { #1 }
5183 }
5184
5185 \NewDocumentCommand \renamedec1 { 0{} m m }{
5186   \__stex_copymodule_renamedec1_args:n { #1 }
5187   \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
5188   \stex_debug:nn{renamedec1}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
5189   \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
5190   \str_if_empty:NTF \l_stex_renamedec1_name_str {
5191     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
5192       \l_stex_get_symbol_uri_str
5193     } }
5194   } {
5195     \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex
5196     \stex_debug:nn{renamedec1}{@~\l_stex_current_module_str ? \l_stex_renamedec1_name_str}
5197     \prop_set_eq:cc {l_stex_symdecl_
5198       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
5199     _prop
5200     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
5201     \seq_set_eq:cc {l_stex_symdecl_
5202       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
5203     _notations
5204     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
5205     \prop_put:cnx {l_stex_symdecl_
5206       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
5207     _prop
5208     }{ name }{ \l_stex_renamedec1_name_str }
5209     \prop_put:cnx {l_stex_symdecl_
5210       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
5211     _prop
5212     }{ module }{ \l_stex_current_module_str }
5213     \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
5214       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
5215     }
5216     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
5217       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
5218     } }
5219   }

```

```

5220 \stex_smsmode_do:
5221 }
5222
5223 \stex_deactivate_macro:Nn \assign {copymodules}
5224 \stex_deactivate_macro:Nn \renamedekl {copymodules}
5225 \stex_deactivate_macro:Nn \donotcopy {copymodules}
5226
5227

```

## 33.2 The feature environment

`structural@feature (env.)`

```

5228 <@@=stex_features>
5229
5230 \NewDocumentEnvironment{structural_feature_module}{ m m m }{
5231 \stex_if_in_module:F {
5232 \msg_set:nnn{stex}{error/nomodule}{
5233 Structural~Feature~has~to~occur~in~a~module:\\
5234 Feature~#2~of~type~#1\\
5235 In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
5236 }
5237 \msg_error:nn{stex}{error/nomodule}
5238 }
5239
5240 \str_set_eq:NN \l_stex_feature_parent_str \l_stex_current_module_str
5241
5242 \stex_module_setup:nn{meta=NONE}{#2 - #1}
5243
5244 \stex_if_do_html:T {
5245 \begin{stex_annotate_env}{ feature:#1 }{\l_stex_feature_parent_str ? #2 - #1}
5246 \stex_annotate_invisible:nnn{header}{}{ #3 }
5247 }
5248 }{
5249 \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
5250 \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
5251 \stex_debug:nn{features}{
5252 Feature: \l_stex_last_feature_str
5253 }
5254 \stex_if_do_html:T {
5255 \end{stex_annotate_env}
5256 }
5257 }

```

## 33.3 Structure

`structure (env.)`

```

5258 <@@=stex_structures>
5259 \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
5260 \prop_if_exist:cF {c_stex_module_ \l_stex_current_module_str _structures}{
5261 \prop_new:c {c_stex_module_ \l_stex_current_module_str _structures}
5262 }
5263 \prop_gput:cxn{c_stex_module_ \l_stex_current_module_str _structures}

```

```

5264     {#1}{#2}
5265 }
5266
5267 \keys_define:nn { stex / features / structure } {
5268   name          .str_set_x:N = \l__stex_structures_name_str ,
5269 }
5270
5271 \cs_new_protected:Nn \__stex_structures_structure_args:n {
5272   \str_clear:N \l__stex_structures_name_str
5273   \keys_set:nn { stex / features / structure } { #1 }
5274 }
5275 \NewDocumentEnvironment{mathstructure}{m O{}}{
5276   \begin{mathstructure_inner}{#1}[#2]
5277     \stex_smsmode_do:
5278     \ignorespacesandpars
5279   }\end{mathstructure_inner}}
5280 \NewDocumentEnvironment{mathstructure_inner}{m O{}}{
5281   \__stex_structures_structure_args:n { #2 }
5282   \str_if_empty:NT \l__stex_structures_name_str {
5283     \str_set:Nx \l__stex_structures_name_str { #1 }
5284   }
5285   \stex_suppress_html:n {
5286     \bool_set_true:N \l_stex_symdecl_make_macro_bool
5287     \exp_args:Nx \stex_symdecl_do:nn {
5288       name = \l__stex_structures_name_str ,
5289       def = {\STEXsymbol{module-type}}{
5290         \STEXInternalTermMathOMSiiii {
5291           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
5292             { ns } ?
5293           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
5294             { name } / \l__stex_structures_name_str - structure
5295         }{}{}{}
5296       }}
5297     }{ #1 }
5298   }
5299   \exp_args:Nnnx
5300   \begin{structural_feature_module}{ structure }
5301     { \l__stex_structures_name_str }{}
5302 }{
5303   \end{structural_feature_module}
5304   \_stex_reset_up_to_module:n \l_stex_last_feature_str
5305   \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
5306   \seq_clear:N \l_tmpa_seq
5307   \seq_map_inline:Nn \l_stex_collect_imports_seq {
5308     \seq_map_inline:cn{c_stex_module_##1_constants}{
5309       \seq_put_right:Nn \l_tmpa_seq { ##1 ? ####1 }
5310     }
5311   }
5312   \exp_args:Nnno
5313   \prop_gput:cn {c_stex_module_ \l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
5314   \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
5315   \stex_add_structure_to_current_module:nn
5316     \l__stex_structures_name_str
5317     \l_stex_last_feature_str

```

```

5318
5319 \stex_execute_in_module:x {
5320   \tl_set:cn { #1 }{
5321     \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures
5322   }
5323 }
5324 }
5325
5326 \cs_new:Nn \stex_invoke_structure:nn {
5327   \stex_invoke_symbol:n { #1?#2 }
5328 }
5329
5330 \cs_new_protected:Nn \stex_get_structure:n {
5331   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
5332     \tl_set:Nn \l_tmpa_tl { #1 }
5333     \__stex_structures_get_from_cs:
5334   }{
5335     \cs_if_exist:cTF { #1 }{
5336       \cs_set_eq:Nc \l_tmpa_cs { #1 }
5337       \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
5338       \str_if_empty:NNTF \l_tmpa_str {
5339         \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs} \stex_invoke_structure:nn {
5340           \__stex_structures_get_from_cs:
5341         }{
5342           \__stex_structures_get_from_string:n { #1 }
5343         }
5344       }{
5345         \__stex_structures_get_from_string:n { #1 }
5346       }
5347     }{
5348       \__stex_structures_get_from_string:n { #1 }
5349     }
5350   }
5351 }
5352
5353 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
5354   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
5355   { \tl_tail:N \l_tmpa_tl }
5356   \str_set:Nx \l_tmpa_str {
5357     \exp_after:wN \use_i:nn \l_tmpa_tl
5358   }
5359   \str_set:Nx \l_tmpb_str {
5360     \exp_after:wN \use_ii:nn \l_tmpa_tl
5361   }
5362   \str_set:Nx \l_stex_get_structure_str {
5363     \l_tmpa_str ? \l_tmpb_str
5364   }
5365   \str_set:Nx \l_stex_get_structure_module_str {
5366     \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
5367   }
5368 }
5369
5370 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
5371   \tl_set:Nn \l_tmpa_tl {

```

```

5372 \msg_error:nnn{stex}{error/unknownstructure}{#1}
5373 }
5374 \str_set:Nn \l_tmpa_str { #1 }
5375 \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
5376
5377 \seq_map_inline:Nn \l_stex_all_modules_seq {
5378   \prop_if_exist:cT {c_stex_module_##1_structures} {
5379     \prop_map_inline:cn {c_stex_module_##1_structures} {
5380       \exp_args:No \str_if_eq:nnT \l_tmpa_str {####1}{
5381         %\str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
5382         \prop_map_break:n{\seq_map_break:n{
5383           \tl_set:Nn \l_tmpa_tl {
5384             \str_set:Nn \l_stex_get_structure_str {##1?####1}
5385             \str_set:Nn \l_stex_get_structure_module_str {####2}
5386           }
5387         }}
5388       }
5389     }
5390   }
5391 }
5392 \l_tmpa_tl
5393 }

```

**\instantiate**

```

5394
5395 \NewDocumentEnvironment{usestructure}{m}{
5396   \stex_get_structure:n {#1}
5397   \exp_args:Nnx \stex_debug:nn{features}{using~structure:~\l_stex_get_structure_module_str}
5398   \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5399 }{}
5400
5401 \keys_define:nn { stex / instantiate } {
5402   name .str_set_x:N = \l__stex_structures_name_str
5403 }
5404 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
5405   \str_clear:N \l__stex_structures_name_str
5406   \keys_set:nn { stex / instantiate } { #1 }
5407 }
5408
5409 \NewDocumentEnvironment{extstructure}{m m O{}}{
5410   \begin{mathstructure_inner}{#1}[#3]
5411     \seq_set_split:Nnn\__stex_structures_extstructure_imports_seq,{#2}
5412     \seq_map_inline:Nn\__stex_structures_extstructure_imports_seq {
5413       \stex_get_structure:n {##1}
5414       \exp_args:Nnx \stex_debug:nn{features}{importing~structure:~\l_stex_get_structure_modu
5415       \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5416       \stex_if_smsmode:F {
5417         \stex_annotate_invisible:nnn
5418         {import} {\l_stex_get_structure_module_str} {}
5419       }
5420       \exp_args:Nx \stex_add_import_to_current_module:n {
5421         \l_stex_get_structure_module_str
5422       }
5423       \exp_args:Nx \stex_add_to_current_module:n {

```

```

5424         \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5425     }
5426 }
5427 \stex_smsmode_do:
5428 \ignorespacesandpars
5429 }{
5430 \end{mathstructure_inner}
5431 }
5432
5433 \NewDocumentEnvironment{extstructure*}{m m O{}}{
5434 % TODO
5435 \begin{extstructure}{#1}{#2}{#3}
5436 }{
5437 \end{extstructure}
5438 }
5439
5440 \NewDocumentCommand \instantiate {m O{ } m m O{}}{
5441 \begingroup
5442 \stex_get_structure:n {#3}
5443 \__stex_structures_instantiate_args:n { #2 }
5444 \str_if_empty:NT \l__stex_structures_name_str {
5445 \str_set:Nn \l__stex_structures_name_str { #1 }
5446 }
5447 \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5448 \seq_clear:N \l__stex_structures_fields_seq
5449 \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
5450 \seq_map_inline:Nn \l_stex_collect_imports_seq {
5451 \seq_map_inline:cn {c_stex_module_##1_constants}{
5452 \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? #####1 }
5453 }
5454 }
5455
5456 \tl_if_empty:nF{#5}{
5457 \seq_set_split:Nnn \l_tmpa_seq , {#5}
5458 \prop_clear:N \l_tmpa_prop
5459 \seq_map_inline:Nn \l_tmpa_seq {
5460 \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
5461 \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
5462 \msg_error:nnn{stex}{error/keyval}{##1}
5463 }
5464 \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
5465 \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
5466 \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
5467 \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}
5468 \exp_args:Nxx \str_if_eq:nnF
5469 {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5470 {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
5471 \msg_error:nnxxx{stex}{error/incompatible}
5472 {\l__stex_structures_dom_str}
5473 {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5474 {\l_stex_get_symbol_uri_str}
5475 {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
5476 }
5477 \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str

```

```

5478     }
5479 }
5480
5481 \seq_map_inline:Nn \l__stex_structures_fields_seq {
5482   \str_set:Nx \l_tmpa_str {field:\l__stex_structures_name_str . \prop_item:cn {l_stex_sy
5483   \stex_debug:nn{instantiate}{Field~\l_tmpa_str :~##1}
5484
5485   \stex_add_constant_to_current_module:n {\l_tmpa_str}
5486   \stex_execute_in_module:x {
5487     \prop_set_from_keyval:cn { l_stex_symdecl_ \l_stex_current_module_str?\l_tmpa_str _p
5488       name   = \l_tmpa_str ,
5489       args   = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
5490       arity  = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
5491       assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs} ,
5492       argnames = {\prop_item:cn {l_stex_symdecl_##1_prop}{argnames}}
5493   }
5494   \seq_clear:c {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notations}
5495 }
5496
5497 \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
5498   \stex_find_notation:nn{##1}{ }
5499   \stex_execute_in_module:x {
5500     \seq_put_right:cn {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notation
5501   }
5502
5503   \stex_copy_control_sequence_ii:ccN
5504     {stex_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_notation_
5505     {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5506     \l_tmpa_tl
5507   \exp_args:No \stex_execute_in_module:n \l_tmpa_tl
5508
5509
5510   \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
5511     \tl_set_eq:Nc \l_tmpa_cs {stex_op_notation_##1\c_hash_str \l_stex_notation_variant
5512     \stex_execute_in_module:x {
5513       \tl_set:cn
5514         {stex_op_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_not
5515         { \exp_args:No \exp_not:n \l_tmpa_cs}
5516     }
5517   }
5518
5519 }
5520
5521 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\l_stex_cur
5522 }
5523
5524 \stex_execute_in_module:x {
5525   \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
5526     domain = \l_stex_get_structure_module_str ,
5527     \prop_to_keyval:N \l_tmpa_prop
5528   }
5529   \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
5530 }
5531 \stex_debug:nn{instantiate}{

```

```

5532 Instance~\l_stex_current_module_str?\l__stex_structures_name_str \\  

5533 \prop_to_keyval:N \l_tmpa_prop  

5534 }  

5535 \exp_args:Nxx \stex_symdecl_do:nn {  

5536   type={\STEXsymbol{module-type}}{  

5537     \STEXInternalTermMathOMSiiii {  

5538       \l_stex_get_structure_module_str  

5539     }{}{0}{}  

5540   }}  

5541 }{\l__stex_structures_name_str}  

5542 % {  

5543   \str_set:Nx \l_stex_get_symbol_uri_str {\l_stex_current_module_str?\l__stex_structures  

5544   \tl_set:Nn \l_stex_notation_after_do_tl {\__stex_notation_final:}  

5545   \stex_notation_do:nnnnn{}{0}{}{\comp{#4}}  

5546 % }  

5547 %\exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}  

5548 \endgroup  

5549 \stex_smsmode_do:\ignorespacesandpars  

5550 }  

5551  

5552 \cs_new_protected:Nn \stex_symbol_or_var:n {  

5553   \cs_if_exist:cTF{#1}{  

5554     \cs_set_eq:Nc \l_tmpa_tl { #1 }  

5555     \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }  

5556     \str_if_empty:NTF \l_tmpa_str {  

5557       \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }  

5558       \stex_invoke_variable:n {  

5559         \bool_set_true:N \l_stex_symbol_or_var_bool  

5560         \bool_set_false:N \l_stex_instance_or_symbol_bool  

5561         \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}  

5562         \tl_set:Nx \l_tmpa_tl {\exp_after:wN \use:n \l_tmpa_tl}  

5563         \str_set:Nx \l_stex_get_symbol_uri_str {  

5564           \exp_after:wN \use:n \l_tmpa_tl  

5565         }  

5566       }{ % TODO \stex_invoke_varinstance:n  

5567         \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl } \stex_invoke_varinstance:n {  

5568           \bool_set_true:N \l_stex_symbol_or_var_bool  

5569           \bool_set_true:N \l_stex_instance_or_symbol_bool  

5570           \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}  

5571           \tl_set:Nx \l_tmpa_tl {\exp_after:wN \use:n \l_tmpa_tl}  

5572           \str_set:Nx \l_stex_get_symbol_uri_str {  

5573             \exp_after:wN \use:n \l_tmpa_tl  

5574           }  

5575         }{  

5576           \bool_set_false:N \l_stex_symbol_or_var_bool  

5577           \stex_get_symbol:n{#1}  

5578         }  

5579       }  

5580     }{  

5581       \__stex_structures_symbolorvar_from_string:n{ #1 }  

5582     }  

5583   }{  

5584     \__stex_structures_symbolorvar_from_string:n{ #1 }  

5585   }  


```



```

5586 }
5587
5588 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
5589   \prop_if_exist:cTF {l_stex_symdecl_var://#1 _prop}{
5590     \bool_set_true:N \l_stex_symbol_or_var_bool
5591     \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
5592   }{
5593     \bool_set_false:N \l_stex_symbol_or_var_bool
5594     \stex_get_symbol:n{#1}
5595   }
5596 }
5597
5598 \keys_define:nn { stex / varinstantiate } {
5599   name .str_set_x:N = \l__stex_structures_name_str,
5600   bind .choices:nn =
5601     {forall,exists}
5602     {\str_set:Nx \l__stex_structures_bind_str {\l_keys_choice_tl}}
5603 }
5604
5605 \cs_new_protected:Nn \__stex_structures_varinstantiate_args:n {
5606   \str_clear:N \l__stex_structures_name_str
5607   \str_clear:N \l__stex_structures_bind_str
5608   \keys_set:nn { stex / varinstantiate } { #1 }
5609 }
5610
5611 \NewDocumentCommand \varinstantiate {m O{}} m m O{}{}{
5612   \begingroup
5613     \stex_get_structure:n {#3}
5614     \__stex_structures_varinstantiate_args:n { #2 }
5615     \str_if_empty:NT \l__stex_structures_name_str {
5616       \str_set:Nn \l__stex_structures_name_str { #1 }
5617     }
5618     \stex_if_do_html:TF{
5619       \stex_annotate:nnn{varinstance}{\l__stex_structures_name_str}
5620     }{\use:n}
5621     {
5622       \stex_if_do_html:T{
5623         \stex_annotate_invisible:nnn{domain}{\l_stex_get_structure_module_str}{}
5624       }
5625       \seq_clear:N \l__stex_structures_fields_seq
5626       \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
5627       \seq_map_inline:Nn \l_stex_collect_imports_seq {
5628         \seq_map_inline:cn {c_stex_module_##1_constants}{
5629           \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
5630         }
5631       }
5632       \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5633       \prop_clear:N \l_tmpa_prop
5634       \tl_if_empty:nF {#5} {
5635         \seq_set_split:Nnn \l_tmpa_seq , {#5}
5636         \seq_map_inline:Nn \l_tmpa_seq {
5637           \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
5638           \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
5639             \msg_error:nnn{stex}{error/keyval}{##1}

```

```

5640     }
5641     \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_structures_dom_str
5642     \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
5643     \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_uri_str
5644     \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
5645     \stex_if_do_html:T{
5646         \stex_annotate:nnn{assign}{\l__stex_structures_dom_str,
5647         \bool_if:NTF\l_stex_symbol_or_var_bool{var://}{\l_stex_get_symbol_uri_str}{}}
5648     }
5649     \bool_if:NTF \l_stex_symbol_or_var_bool {
5650         \exp_args:Nxx \str_if_eq:nnF
5651             {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5652             {\prop_item:cn{l_stex_symdecl_var://\l_stex_get_symbol_uri_str _prop}{args}}}{
5653             \msg_error:nnxxxx{stex}{error/incompatible}
5654             {\l__stex_structures_dom_str}
5655             {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5656             {\l_stex_get_symbol_uri_str}
5657             {\prop_item:cn{l_stex_symdecl_var://\l_stex_get_symbol_uri_str _prop}{args}}
5658         }
5659         \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n {}}
5660     }{
5661         \exp_args:Nxx \str_if_eq:nnF
5662             {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5663             {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}}{
5664             \msg_error:nnxxxx{stex}{error/incompatible}
5665             {\l__stex_structures_dom_str}
5666             {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5667             {\l_stex_get_symbol_uri_str}
5668             {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
5669         }
5670         \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n {}}
5671     }
5672 }
5673 }
5674 \tl_gclear:N \g__stex_structures_aftergroup_tl
5675 \seq_map_inline:Nn \l__stex_structures_fields_seq {
5676     \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdecl_
5677     \stex_debug:nn{varinstantiate}{Field~\l_tmpa_str :~##1}
5678     \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
5679         \stex_find_notation:nn{##1}{}
5680         \cs_gset_eq:cc{g__stex_structures_tmpa_\l_tmpa_str _cs}
5681             {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5682         \stex_debug:nn{varinstantiate}{Notation:~\cs_meaning:c{g__stex_structures_tmpa_\l
5683         \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
5684             \cs_gset_eq:cc {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
5685             {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5686             \stex_debug:nn{varinstantiate}{Operator~Notation:~\cs_meaning:c{g__stex_struct
5687     }
5688 }
5689 }
5690 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
5691     \prop_set_from_keyval:cn { l_stex_symdecl_ var://\l_tmpa_str _prop}{
5692         name      = \l_tmpa_str ,
5693         args      = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,

```

```

5694         arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
5695         assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs} ,
5696         argnames = {\prop_item:cn {l_stex_symdecl_##1_prop}{argnames}} ,
5697     }
5698     \cs_set_eq:cc {stex_var_notation_\l_tmpa_str_cs}
5699     {g__stex_structures_tmpa_\l_tmpa_str_cs}
5700     \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str_cs}
5701     {g__stex_structures_tmpa_op_\l_tmpa_str_cs}
5702 }
5703 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_inv
5704 }
5705 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
5706     \prop_set_from_keyval:cn {l_stex_varinstance_\l__stex_structures_name_str_prop }{
5707         domain = \l_stex_get_structure_module_str ,
5708         \prop_to_keyval:N \l_tmpa_prop
5709     }
5710     \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
5711     \tl_set:cn {l_stex_varinstance_\l__stex_structures_name_str_op_tl}{
5712         \exp_args:Nnx \exp_not:N \use:nn {
5713             \str_set:Nn \exp_not:N \STEXInternalCurrentSymbolStr {var://\l__stex_structures_
5714             \_stex_term_omv:nn {var://\l__stex_structures_name_str}{
5715                 \exp_not:n{
5716                     \_varcomp{#4}
5717                 }
5718             }
5719             }{
5720                 \exp_not:n{\_stex_reset:N \STEXInternalCurrentSymbolStr}
5721             }
5722         }
5723     }
5724 }
5725 \stex_debug:nn{varinstantiate}{\expandafter\detokenize\expandafter{\g__stex_structures_a
5726 \aftergroup\g__stex_structures_aftergroup_tl
5727 \endgroup
5728 \stex_smsmode_do:\ignorespacesandpars
5729 }
5730
5731 \cs_new_protected:Nn \stex_invoke_instance:n {
5732     \peek_charcode_remove:NTF ! {
5733         \stex_invoke_symbol:n{#1}
5734     }{
5735         \_stex_invoke_instance:nn {#1}
5736     }
5737 }
5738
5739
5740 \cs_new_protected:Nn \stex_invoke_varinstance:n {
5741     \peek_charcode_remove:NTF ! {
5742         \exp_args:Nnx \use:nn {
5743             \def\comp{\_varcomp}
5744             \use:c{l_stex_varinstance_#1_op_tl}
5745         }{
5746             \_stex_reset:N \comp
5747         }

```

```

5748 }{
5749   \_stex_invoke_varinstance:nn {#1}
5750 }
5751 }
5752
5753 \cs_new_protected:Nn \_stex_invoke_instance:nn {
5754   \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
5755     \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
5756   }{
5757     \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
5758     \msg_error:nnxxx{stex}{error/unknownfield}{#2}{#1}{
5759       \prop_to_keyval:N \l_tmpa_prop
5760     }
5761   }
5762 }
5763
5764 \cs_new_protected:Nn \_stex_invoke_varinstance:nn {
5765   \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
5766     \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
5767     \l_tmpa_tl
5768   }{
5769     \msg_error:nnnnn{stex}{error/unknownfield}{#2}{#1}{
5770     }
5771   }

```

(End definition for `\instantiate`. This function is documented on page 38.)

`\stex_invoke_structure:nnn`

```

5772 % #1: URI of the instance
5773 % #2: URI of the instantiated module
5774 \cs_new_protected:Nn \stex_invoke_structure:nnn {
5775   \tl_if_empty:nTF{ #3 }{
5776     \prop_set_eq:Nc \l__stex_structures_structure_prop {
5777       c_stex_feature_ #2 _prop
5778     }
5779     \tl_clear:N \l_tmpa_tl
5780     \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
5781     \seq_map_inline:Nn \l_tmpa_seq {
5782       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
5783       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
5784       \cs_if_exist:cT {
5785         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
5786       }{
5787         \tl_if_empty:NF \l_tmpa_tl {
5788           \tl_put_right:Nn \l_tmpa_tl {,}
5789         }
5790         \tl_put_right:Nx \l_tmpa_tl {
5791           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
5792         }
5793       }
5794     }
5795     \exp_args:No \mathstrut \l_tmpa_tl
5796   }{
5797     \stex_invoke_symbol:n{#1/#3}

```

```

5798     }
5799 }

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

5800 \end{package}

```

## Chapter 34

# STEX -Statements Implementation

```
5801 <*package>
5802
5803 %%%%%%%%%%% features.dtx %%%%%%%%%%%
5804
5805 <@@=stex_statements>
    Warnings and error messages
5806
\titleemph
5807 \def\titleemph#1{\textbf{#1}}
    (End definition for \titleemph. This function is documented on page ??.)
```

### 34.1 Definitions

#### definiendum

```
5808 \keys_define:nn {stex / definiendum }{
5809   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
5810   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
5811   root     .str_set_x:N    = \l__stex_statements_definiendum_root_str,
5812   gfa      .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
5813 }
5814 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
5815   \str_clear:N \l__stex_statements_definiendum_root_str
5816   \tl_clear:N \l__stex_statements_definiendum_post_tl
5817   \str_clear:N \l__stex_statements_definiendum_gfa_str
5818   \keys_set:nn { stex / definiendum }{ #1 }
5819 }
5820 \NewDocumentCommand \definiendum { O{} m m } {
5821   \__stex_statements_definiendum_args:n { #1 }
5822   \stex_get_symbol:n { #2 }
5823   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5824   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
5825     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```

5826     \tl_set:Nn \l_tmpa_tl { #3 }
5827   } {
5828     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
5829     \tl_set:Nn \l_tmpa_tl {
5830       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
5831     }
5832   }
5833 } {
5834   \tl_set:Nn \l_tmpa_tl { #3 }
5835 }
5836
5837 % TODO root
5838 \stex_html_backend:TF {
5839   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
5840 } {
5841   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
5842 }
5843 }
5844 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

(End definition for definiendum. This function is documented on page 48.)

#### definame

```

5845
5846 \NewDocumentCommand \definame { 0{ } m } {
5847   \__stex_statements_definiendum_args:n { #1 }
5848   % TODO: root
5849   \stex_get_symbol:n { #2 }
5850   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5851   \str_set:Nx \l_tmpa_str {
5852     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5853   }
5854   \str_replace_all:Nnn \l_tmpa_str {-} {~}
5855   \stex_html_backend:TF {
5856     \stex_if_do_html:T {
5857       \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5858         \l_tmpa_str\l__stex_statements_definiendum_post_tl
5859       }
5860     }
5861   } {
5862     \exp_args:Nnx \defemph@uri {
5863       \l_tmpa_str\l__stex_statements_definiendum_post_tl
5864     } { \l_stex_get_symbol_uri_str }
5865   }
5866 }
5867 \stex_deactivate_macro:Nn \definame {definition~environments}
5868
5869 \NewDocumentCommand \Definame { 0{ } m } {
5870   \__stex_statements_definiendum_args:n { #1 }
5871   \stex_get_symbol:n { #2 }
5872   \str_set:Nx \l_tmpa_str {
5873     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5874   }
5875   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}

```

```

5876 \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5877 \stex_html_backend:TF {
5878   \stex_if_do_html:T {
5879     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5880       \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5881     }
5882   }
5883 } {
5884   \exp_args:Nnx \defemph@uri {
5885     \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5886   } { \l_stex_get_symbol_uri_str }
5887 }
5888 }
5889 \stex_deactivate_macro:Nn \Definame {definition-environments}
5890
5891 \NewDocumentCommand \premise { m }{
5892   \noindent\stex_annotate:nnn{ premise }{}{\ignorespaces #1 }
5893 }
5894 \NewDocumentCommand \conclusion { m }{
5895   \noindent\stex_annotate:nnn{ conclusion }{}{\ignorespaces #1 }
5896 }
5897 \NewDocumentCommand \definiens { 0{} m }{
5898   \str_clear:N \l_stex_get_symbol_uri_str
5899   \tl_if_empty:nF {#1} {
5900     \stex_get_symbol:n { #1 }
5901   }
5902   \str_if_empty:NT \l_stex_get_symbol_uri_str {
5903     \int_compare:nNnTF {\clist_count:N \l__stex_statements_sdefinition_for_clist} = 1 {
5904       \str_set:Nx \l_stex_get_symbol_uri_str {\clist_item:Nn \l__stex_statements_sdefinition
5905     }{
5906       % TODO throw error
5907     }
5908   }
5909   \str_if_eq:eeT {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{module}}
5910   {\l_stex_current_module_str}{
5911     \str_if_eq:eeF {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defin
5912   }{true}{
5913     \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5914     \exp_args:Nx \stex_add_to_current_module:n {
5915       \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5916     }
5917   }
5918 }
5919 \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
5920 }
5921
5922 \NewDocumentCommand \varbindforall {m}{
5923   \stex_symbol_or_var:n {#1}
5924   \bool_if:NTF\l_stex_symbol_or_var_bool{
5925     \stex_if_do_html:T {
5926       \stex_annotate_invisible:nnn {bindtype}{forall,\l_stex_get_symbol_uri_str}{}
5927     }
5928   }{
5929     % todo throw error

```



```

5930 }
5931 }
5932
5933 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
5934 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
5935 \stex_deactivate_macro:Nn \definiens {definition~environments}
5936 \stex_deactivate_macro:Nn \varbindforall {definition~or~assertion~environments}
5937

```

(End definition for definame. This function is documented on page 48.)

sdefinition (env.)

```

5938
5939 \keys_define:nn {stex / sdefinition }{
5940   type      .str_set_x:N = \sdefinitiontype,
5941   id        .str_set_x:N = \sdefinitionid,
5942   name      .str_set_x:N = \sdefinitionname,
5943   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5944   title     .tl_set:N     = \sdefinitiontitle
5945 }
5946 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
5947   \str_clear:N \sdefinitiontype
5948   \str_clear:N \sdefinitionid
5949   \str_clear:N \sdefinitionname
5950   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5951   \tl_clear:N \sdefinitiontitle
5952   \keys_set:nn { stex / sdefinition }{ #1 }
5953 }
5954
5955 \NewDocumentEnvironment{sdefinition}{0{}}{
5956   \__stex_statements_sdefinition_args:n{ #1 }
5957   \stex_reactivate_macro:N \definiendum
5958   \stex_reactivate_macro:N \definame
5959   \stex_reactivate_macro:N \Definame
5960   \stex_reactivate_macro:N \premise
5961   \stex_reactivate_macro:N \definiens
5962   \stex_reactivate_macro:N \varbindforall
5963   \stex_if_smsmode:F{
5964     \seq_clear:N \l_tmpb_seq
5965     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5966       \tl_if_empty:nF{ ##1 }{
5967         \stex_get_symbol:n { ##1 }
5968         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5969           \l_stex_get_symbol_uri_str
5970         }
5971       }
5972     }
5973     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5974     \exp_args:Nnnx
5975     \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
5976     \str_if_empty:NF \sdefinitiontype {
5977       \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{ }
5978     }
5979     \str_if_empty:NF \sdefinitionname {

```

```

5980     \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5981   }
5982   \clist_set:No \l_tmpa_clist \sdefinitiontype
5983   \tl_clear:N \l_tmpa_tl
5984   \clist_map_inline:Nn \l_tmpa_clist {
5985     \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
5986       \tl_set:Nn \l_tmpa_tl {
5987         \stex_patch_counters:
5988         \use:c{__stex_statements_sdefinition_##1_start:}
5989         \stex_unpatch_counters:
5990       }
5991     }
5992   }
5993   \tl_if_empty:NTF \l_tmpa_tl {
5994     \__stex_statements_sdefinition_start:
5995   }{
5996     \l_tmpa_tl
5997   }
5998 }
5999 \stex_ref_new_doc_target:n \sdefinitionid
6000 \stex_smsmode_do:
6001 }{
6002   \stex_suppress_html:n {
6003     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
6004   }
6005   \stex_if_smsmode:F {
6006     \clist_set:No \l_tmpa_clist \sdefinitiontype
6007     \tl_clear:N \l_tmpa_tl
6008     \clist_map_inline:Nn \l_tmpa_clist {
6009       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
6010         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
6011       }
6012     }
6013     \tl_if_empty:NTF \l_tmpa_tl {
6014       \__stex_statements_sdefinition_end:
6015     }{
6016       \l_tmpa_tl
6017     }
6018     \end{stex_annotate_env}
6019   }
6020 }

```

### **\stexpatchdefinition**

```

6021 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
6022   \stex_par:\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
6023     ~(\sdefinitiontitle)
6024   }~}
6025 }
6026 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\stex_par:\medskip}
6027
6028 \newcommand\stexpatchdefinition[3] [] {
6029   \str_set:Nx \l_tmpa_str{ #1 }
6030   \str_if_empty:NTF \l_tmpa_str {
6031     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }

```

```

6032     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
6033   }{
6034     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2
6035     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
6036   }
6037 }

```

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

`\inlinedef` inline:

```

6038 \keys_define:nn {stex / inlinedef }{
6039   type      .str_set_x:N = \sdefinitiontype,
6040   id        .str_set_x:N = \sdefinitionid,
6041   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
6042   name      .str_set_x:N = \sdefinitionname
6043 }
6044 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
6045   \str_clear:N \sdefinitiontype
6046   \str_clear:N \sdefinitionid
6047   \str_clear:N \sdefinitionname
6048   \clist_clear:N \l__stex_statements_sdefinition_for_clist
6049   \keys_set:nn { stex / inlinedef }{ #1 }
6050 }
6051 \NewDocumentCommand \inlinedef { 0{} m } {
6052   \beginngroup
6053   \__stex_statements_inlinedef_args:n{ #1 }
6054   \stex_reactivate_macro:N \definiendum
6055   \stex_reactivate_macro:N \definame
6056   \stex_reactivate_macro:N \Definame
6057   \stex_reactivate_macro:N \premise
6058   \stex_reactivate_macro:N \definiens
6059   \stex_reactivate_macro:N \varbindforall
6060   \stex_ref_new_doc_target:n \sdefinitionid
6061   \stex_if_smsmode:TF{\stex_suppress_html:n {
6062     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
6063   }}{
6064     \seq_clear:N \l_tmpb_seq
6065     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
6066       \tl_if_empty:nF{ ##1 }{
6067         \stex_get_symbol:n { ##1 }
6068         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6069           \l_stex_get_symbol_uri_str
6070         }
6071       }
6072     }
6073     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
6074     \ifvmode\noindent\fi
6075     \exp_args:Nnx
6076     \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpb_seq {,}}{
6077       \str_if_empty:NF \sdefinitiontype {
6078         \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{
6079         }
6080       }
6081       \str_if_empty:NF \sdefinitionname {

```

```

6082         \stex_suppress_html:n{\stex_symdecl_do:nn}{\sdefinitionname}}
6083         \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}}
6084     }
6085 }
6086 }
6087 \endgroup
6088 \stex_smsmode_do:
6089 }

```

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

## 34.2 Assertions

`sassertion (env.)`

```

6090
6091 \keys_define:nn {stex / sassertion }{
6092   type      .str_set_x:N = \sassertiontype,
6093   id        .str_set_x:N = \sassertionid,
6094   title     .tl_set:N     = \sassertiontitle ,
6095   for       .clist_set:N  = \l__stex_statements_sassertion_for_clist ,
6096   name      .str_set_x:N  = \sassertionname
6097 }
6098 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
6099   \str_clear:N \sassertiontype
6100   \str_clear:N \sassertionid
6101   \str_clear:N \sassertionname
6102   \clist_clear:N \l__stex_statements_sassertion_for_clist
6103   \tl_clear:N \sassertiontitle
6104   \keys_set:nn { stex / sassertion }{ #1 }
6105 }
6106
6107 %\tl_new:N \g__stex_statements_aftergroup_tl
6108
6109 \NewDocumentEnvironment{sassertion}{0{}}{
6110   \__stex_statements_sassertion_args:n{ #1 }
6111   \stex_reactivate_macro:N \premise
6112   \stex_reactivate_macro:N \conclusion
6113   \stex_reactivate_macro:N \varbindforall
6114   \stex_if_smsmode:F {
6115     \seq_clear:N \l_tmpb_seq
6116     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
6117       \tl_if_empty:nF{ ##1 }{
6118         \stex_get_symbol:n { ##1 }
6119         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6120           \l_stex_get_symbol_uri_str
6121         }
6122       }
6123     }
6124     \exp_args:Nnnx
6125     \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
6126     \str_if_empty:NF \sassertiontype {
6127       \stex_annotate_invisible:nnn{type}{\sassertiontype}{}}
6128   }

```

```

6129 \str_if_empty:NF \sassertionname {
6130 \stex_annotate_invisible:nnn{statementname}{\sassertionname}{\}
6131 }
6132 \clist_set:No \l_tmpa_clist \sassertiontype
6133 \tl_clear:N \l_tmpa_tl
6134 \clist_map_inline:Nn \l_tmpa_clist {
6135 \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
6136 \tl_set:Nn \l_tmpa_tl {
6137 \stex_patch_counters:
6138 \use:c{__stex_statements_sassertion_##1_start:}
6139 \stex_unpatch_counters:
6140 }
6141 }
6142 }
6143 \tl_if_empty:NTF \l_tmpa_tl {
6144 \__stex_statements_sassertion_start:
6145 }{
6146 \l_tmpa_tl
6147 }
6148 }
6149 \str_if_empty:NTF \sassertionid {
6150 \str_if_empty:NF \sassertionname {
6151 \stex_ref_new_doc_target:n {}
6152 }
6153 } {
6154 \stex_ref_new_doc_target:n \sassertionid
6155 }
6156 \stex_smsmode_do:
6157 }{
6158 \str_if_empty:NF \sassertionname {
6159 \stex_suppress_html:n{\stex_symdecl_do:nn}{\sassertionname}}
6160 \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
6161 }
6162 \stex_if_smsmode:F {
6163 \clist_set:No \l_tmpa_clist \sassertiontype
6164 \tl_clear:N \l_tmpa_tl
6165 \clist_map_inline:Nn \l_tmpa_clist {
6166 \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
6167 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
6168 }
6169 }
6170 \tl_if_empty:NTF \l_tmpa_tl {
6171 \__stex_statements_sassertion_end:
6172 }{
6173 \l_tmpa_tl
6174 }
6175 \end{stex_annotate_env}
6176 }
6177 }

```

**\stexpatchassertion**

```

6178
6179 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
6180 \stex_par:\noindent\titileemph{Assertion~\tl_if_empty:NF \sassertiontitle {

```

```

6181      (\sassertiontitle)
6182    }~}
6183  }
6184  \cs_new_protected:Nn \__stex_statements_sassertion_end: {\stex_par:\medskip}
6185
6186  \newcommand\stexpatchassertion[3] [] {
6187    \str_set:Nx \l_tmpa_str{ #1 }
6188    \str_if_empty:NTF \l_tmpa_str {
6189      \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
6190      \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
6191    }{
6192      \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
6193      \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
6194    }
6195  }

```

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

\inlineass inline:

```

6196  \keys_define:nn {stex / inlineass }{
6197    type      .str_set_x:N = \sassertiontype,
6198    id        .str_set_x:N = \sassertionid,
6199    for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
6200    name      .str_set_x:N = \sassertionname
6201  }
6202  \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
6203    \str_clear:N \sassertiontype
6204    \str_clear:N \sassertionid
6205    \str_clear:N \sassertionname
6206    \clist_clear:N \l__stex_statements_sassertion_for_clist
6207    \keys_set:nn { stex / inlineass }{ #1 }
6208  }
6209  \NewDocumentCommand \inlineass { 0{} m } {
6210    \begingroup
6211    \stex_reactivate_macro:N \premise
6212    \stex_reactivate_macro:N \conclusion
6213    \stex_reactivate_macro:N \varbindforall
6214    \__stex_statements_inlineass_args:n{ #1 }
6215    \str_if_empty:NTF \sassertionid {
6216      \str_if_empty:NF \sassertionname {
6217        \stex_ref_new_doc_target:n {}
6218      }
6219    } {
6220      \stex_ref_new_doc_target:n \sassertionid
6221    }
6222
6223    \stex_if_smsmode:TF{
6224      \str_if_empty:NF \sassertionname {
6225        \stex_suppress_html:n{\stex_symdecl_do:nn}{\sassertionname}}
6226        \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
6227      }
6228    }{
6229      \seq_clear:N \l_tmpb_seq
6230      \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {

```

```

6231     \tl_if_empty:nF{ ##1 }{
6232         \stex_get_symbol:n { ##1 }
6233         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6234             \l_stex_get_symbol_uri_str
6235         }
6236     }
6237 }
6238 \ifvmode\noindent\fi
6239 \exp_args:Nnx
6240 \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpb_seq {,}}{
6241     \str_if_empty:NF \sassertiontype {
6242         \stex_annotate_invisible:nnn{typestrings}{\sassertiontype}{}
6243     }
6244     #2
6245     \str_if_empty:NF \sassertionname {
6246         \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sassertionname}}
6247         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
6248         \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
6249     }
6250 }
6251 }
6252 \endgroup
6253 \stex_smsmode_do:
6254 }

```

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

## 34.3 Examples

`sexample (env.)`

```

6255
6256 \keys_define:nn {stex / sexample }{
6257     type      .str_set_x:N = \exampletype,
6258     id        .str_set_x:N = \sexampleid,
6259     title     .tl_set:N    = \sexampletitle,
6260     name      .str_set_x:N = \sexamplename ,
6261     for       .clist_set:N = \l__stex_statements_sexample_for_clist,
6262 }
6263 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
6264     \str_clear:N \sexampletype
6265     \str_clear:N \sexampleid
6266     \str_clear:N \sexamplename
6267     \tl_clear:N \sexampletitle
6268     \clist_clear:N \l__stex_statements_sexample_for_clist
6269     \keys_set:nn { stex / sexample }{ #1 }
6270 }
6271
6272 \NewDocumentEnvironment{sexample}{0{}}{
6273     \__stex_statements_sexample_args:n{ #1 }
6274     \stex_reactivate_macro:N \premise
6275     \stex_reactivate_macro:N \conclusion
6276     \stex_if_smsmode:F {
6277         \seq_clear:N \l_tmpb_seq

```

```

6278 \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
6279   \tl_if_empty:NF{ ##1 }{
6280     \stex_get_symbol:n { ##1 }
6281     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6282       \l_stex_get_symbol_uri_str
6283     }
6284   }
6285 }
6286 \exp_args:Nnnx
6287 \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpb_seq {,}}
6288 \str_if_empty:NF \sexamplotype {
6289   \stex_annotate_invisible:nnn{typestrings}{\sexampletype}{\}
6290 }
6291 \str_if_empty:NF \sexamplename {
6292   \stex_annotate_invisible:nnn{statementname}{\sexamplename}{\}
6293 }
6294 \clist_set:No \l_tmpa_clist \sexamplotype
6295 \tl_clear:N \l_tmpa_tl
6296 \clist_map_inline:Nn \l_tmpa_clist {
6297   \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
6298     \tl_set:Nn \l_tmpa_tl {
6299       \stex_patch_counters:
6300       \use:c{__stex_statements_sexample_##1_start:}
6301       \stex_unpatch_counters:
6302     }
6303   }
6304 }
6305 \tl_if_empty:NTF \l_tmpa_tl {
6306   \__stex_statements_sexample_start:
6307 }{
6308   \l_tmpa_tl
6309 }
6310 }
6311 \str_if_empty:NF \sexampleid {
6312   \stex_ref_new_doc_target:n \sexampleid
6313 }
6314 \stex_smsmode_do:
6315 }{
6316   \str_if_empty:NF \sexamplename {
6317     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
6318   }
6319   \stex_if_smsmode:F {
6320     \clist_set:No \l_tmpa_clist \sexamplotype
6321     \tl_clear:N \l_tmpa_tl
6322     \clist_map_inline:Nn \l_tmpa_clist {
6323       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
6324         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
6325       }
6326     }
6327     \tl_if_empty:NTF \l_tmpa_tl {
6328       \__stex_statements_sexample_end:
6329     }{
6330       \l_tmpa_tl
6331     }

```



```

6332     \end{stex_annotate_env}
6333   }
6334 }

```

## `\stexpatchexample`

```

6335
6336 \cs_new_protected:Nn \__stex_statements_sexample_start: {
6337   \stex_par:\noindent\titleemph{Example~\tl_if_empty:NF \sexampletile {
6338     (\sexampletile)
6339   }~}
6340 }
6341 \cs_new_protected:Nn \__stex_statements_sexample_end: {\stex_par:\medskip}
6342
6343 \newcommand\stexpatchexample[3] [] {
6344   \str_set:Nx \l_tmpa_str{ #1 }
6345   \str_if_empty:NTF \l_tmpa_str {
6346     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
6347     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
6348   }{
6349     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
6350     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
6351   }
6352 }

```

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

## `\inlineex` inline:

```

6353 \keys_define:nn {stex / inlineex }{
6354   type      .str_set_x:N = \sexamplotype,
6355   id        .str_set_x:N = \sexampleid,
6356   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
6357   name      .str_set_x:N = \sexamplename
6358 }
6359 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
6360   \str_clear:N \sexamplotype
6361   \str_clear:N \sexampleid
6362   \str_clear:N \sexamplename
6363   \clist_clear:N \l__stex_statements_sexample_for_clist
6364   \keys_set:nn { stex / inlineex }{ #1 }
6365 }
6366 \NewDocumentCommand \inlineex { 0{} m } {
6367   \begingroup
6368   \stex_reactivate_macro:N \premise
6369   \stex_reactivate_macro:N \conclusion
6370   \__stex_statements_inlineex_args:n{ #1 }
6371   \str_if_empty:NF \sexampleid {
6372     \stex_ref_new_doc_target:n \sexampleid
6373   }
6374   \stex_if_smsmode:TF{
6375     \str_if_empty:NF \sexamplename {
6376       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sexamplename}}
6377   }
6378 }{
6379   \seq_clear:N \l_tmpb_seq

```

```

6380 \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
6381   \tl_if_empty:NF{ ##1 }{
6382     \stex_get_symbol:n { ##1 }
6383     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6384       \l_stex_get_symbol_uri_str
6385     }
6386   }
6387 }
6388 \ifvmode\noindent\fi
6389 \exp_args:Nnx
6390 \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpb_seq {,}}{
6391   \str_if_empty:NF \sexamplotype {
6392     \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{}
6393   }
6394   #2
6395   \str_if_empty:NF \sexamplename {
6396     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
6397     \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
6398   }
6399 }
6400 }
6401 \endgroup
6402 \stex_smsmode_do:
6403 }

```

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

## 34.4 Logical Paragraphs

`sparagraph` (*env.*)

```

6404 \keys_define:nn { stex / spparagraph } {
6405   id      .str_set:N = \sparagraphid ,
6406   title   .tl_set:N  = \l_stex_sparagraph_title_tl ,
6407   type    .str_set:N = \sparagraphtype ,
6408   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
6409   from    .tl_set:N  = \sparagraphfrom ,
6410   to      .tl_set:N  = \sparagraphto ,
6411   start   .tl_set:N  = \l_stex_sparagraph_start_tl ,
6412   name    .str_set:N  = \sparagraphname ,
6413   imports .tl_set:N  = \l__stex_statements_sparagraph_imports_tl
6414 }
6415
6416 \cs_new_protected:Nn \stex_sparagraph_args:n {
6417   \tl_clear:N \l_stex_sparagraph_title_tl
6418   \tl_clear:N \sparagraphfrom
6419   \tl_clear:N \sparagraphto
6420   \tl_clear:N \l_stex_sparagraph_start_tl
6421   \tl_clear:N \l__stex_statements_sparagraph_imports_tl
6422   \str_clear:N \sparagraphid
6423   \str_clear:N \sparagraphtype
6424   \clist_clear:N \l__stex_statements_sparagraph_for_clist
6425   \str_clear:N \sparagraphname
6426   \keys_set:nn { stex / spparagraph }{ #1 }

```

```

6427 }
6428 \newif\if@in@omtext\@in@omtextfalse
6429
6430 \NewDocumentEnvironment {sparagraph} { 0{} } {
6431   \stex_sparagraph_args:n { #1 }
6432   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
6433     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
6434   }{
6435     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
6436   }
6437   \@in@omtexttrue
6438   \stex_if_smsmode:F {
6439     \seq_clear:N \l_tmpb_seq
6440     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
6441       \tl_if_empty:NF{ ##1 }{
6442         \stex_get_symbol:n { ##1 }
6443         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6444           \l_stex_get_symbol_uri_str
6445         }
6446       }
6447     }
6448     \exp_args:Nnnx
6449     \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}
6450     \str_if_empty:NF \sparagraphtype {
6451       \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{ }
6452     }
6453     \str_if_empty:NF \sparagraphfrom {
6454       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{ }
6455     }
6456     \str_if_empty:NF \sparagraphto {
6457       \stex_annotate_invisible:nnn{to}{\sparagraphto}{ }
6458     }
6459     \str_if_empty:NF \sparagraphname {
6460       \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{ }
6461     }
6462     \clist_set:No \l_tmpa_clist \sparagraphtype
6463     \tl_clear:N \l_tmpa_tl
6464     \clist_map_inline:Nn \sparagraphtype {
6465       \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
6466         \tl_set:Nn \l_tmpa_tl {
6467           \stex_patch_counters:
6468           \use:c{__stex_statements_sparagraph_##1_start:}
6469           \stex_unpatch_counters:
6470         }
6471       }
6472     }
6473     \stex_csl_to_imports:No \usemodule \l__stex_statements_sparagraph_imports_tl
6474     \tl_if_empty:NTF \l_tmpa_tl {
6475       \__stex_statements_sparagraph_start:
6476     }{
6477       \l_tmpa_tl
6478     }
6479   }
6480   \clist_set:No \l_tmpa_clist \sparagraphtype

```

```

6481 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}
6482 {
6483   \stex_reactivate_macro:N \definiendum
6484   \stex_reactivate_macro:N \definame
6485   \stex_reactivate_macro:N \Definame
6486   \stex_reactivate_macro:N \premise
6487   \stex_reactivate_macro:N \definiens
6488 }
6489 \str_if_empty:NTF \sparagraphid {
6490   \str_if_empty:NTF \sparagraphname {
6491     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
6492       \stex_ref_new_doc_target:n {}
6493     }
6494   } {
6495     \stex_ref_new_doc_target:n {}
6496   }
6497 } {
6498   \stex_ref_new_doc_target:n \sparagraphid
6499 }
6500 \exp_args:NNx
6501 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
6502   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
6503     \tl_if_empty:nF{ ##1 }{
6504       \stex_get_symbol:n { ##1 }
6505       \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
6506     }
6507   }
6508 }
6509 \stex_smsmode_do:
6510 \ignorespacesandpars
6511 }{
6512   \str_if_empty:NF \sparagraphname {
6513     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
6514     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
6515   }
6516   \stex_if_smsmode:F {
6517     \clist_set:No \l_tmpa_clist \sparagraphtype
6518     \tl_clear:N \l_tmpa_tl
6519     \clist_map_inline:Nn \l_tmpa_clist {
6520       \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
6521         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
6522       }
6523     }
6524     \tl_if_empty:NTF \l_tmpa_tl {
6525       \__stex_statements_sparagraph_end:
6526     }{
6527       \l_tmpa_tl
6528     }
6529     \end{stex_annotate_env}
6530   }
6531 }

```

**\stexpatchparagraph**

6532

```

6533 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
6534   \stex_par:\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
6535     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
6536       \titleemph{\l_stex_sparagraph_title_tl}:~
6537     }
6538   }{
6539     \titleemph{\l_stex_sparagraph_start_tl}~
6540   }
6541 }
6542 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\stex_par:\medskip}
6543
6544 \newcommand\stexpatchparagraph[3] [] {
6545   \str_set:Nx \l_tmpa_str{ #1 }
6546   \str_if_empty:NTF \l_tmpa_str {
6547     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
6548     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
6549   }{
6550     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
6551     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
6552   }
6553 }
6554
6555 \keys_define:nn { stex / inlinepara } {
6556   id      .str_set:x:N = \sparagraphid ,
6557   type    .str_set:x:N = \sparagraphtype ,
6558   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
6559   from    .tl_set:N    = \sparagraphfrom ,
6560   to      .tl_set:N    = \sparagraphto ,
6561   name    .str_set:N    = \sparagraphname
6562 }
6563 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
6564   \tl_clear:N \sparagraphfrom
6565   \tl_clear:N \sparagraphto
6566   \str_clear:N \sparagraphid
6567   \str_clear:N \sparagraphtype
6568   \clist_clear:N \l__stex_statements_sparagraph_for_clist
6569   \str_clear:N \sparagraphname
6570   \keys_set:nn { stex / inlinepara }{ #1 }
6571 }
6572 \NewDocumentCommand \inlinepara { 0{} m } {
6573   \begingroup
6574   \__stex_statements_inlinepara_args:n{ #1 }
6575   \clist_set:No \l_tmpa_clist \sparagraphtype
6576   \str_if_empty:NTF \sparagraphid {
6577     \str_if_empty:NTF \sparagraphname {
6578       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
6579         \stex_ref_new_doc_target:n {}
6580       }
6581     } {
6582       \stex_ref_new_doc_target:n {}
6583     }
6584   } {
6585     \stex_ref_new_doc_target:n \sparagraphid
6586   }

```

```

6587 \stex_if_smsmode:TF{
6588   \str_if_empty:NF \sparagraphname {
6589     \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
6590     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
6591   }
6592 }{
6593   \seq_clear:N \l_tmpb_seq
6594   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
6595     \tl_if_empty:NF{ ##1 }{
6596       \stex_get_symbol:n { ##1 }
6597       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6598         \l_stex_get_symbol_uri_str
6599       }
6600     }
6601   }
6602   \ifvmode\noindent\fi
6603   \exp_args:Nnx
6604   \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}{
6605     \str_if_empty:NF \sparagraphtype {
6606       \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
6607     }
6608     \str_if_empty:NF \sparagraphfrom {
6609       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
6610     }
6611     \str_if_empty:NF \sparagraphto {
6612       \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
6613     }
6614     \str_if_empty:NF \sparagraphname {
6615       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
6616       \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
6617       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
6618     }
6619     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
6620       \clist_map_inline:Nn \l_tmpb_seq {
6621         \stex_ref_new_sym_target:n {##1}
6622       }
6623     }
6624     #2
6625   }
6626 }
6627 \endgroup
6628 \stex_smsmode_do:
6629 }
6630

```

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

```

6631 \</package>

```

## Chapter 35

# The Implementation

```
6632 <*package>
6633 <@@=stex_sproof>
6634
6635 %%%%%%%%%%    sproof.dtx    %%%%%%%%%%
6636
```

### 35.1 Proofs

We first define some keys for the proof environment.

```
6637 \keys_define:nn { stex / spf } {
6638   id          .str_set_x:N = \spfid,
6639   for          .clist_set:N = \l__stex_sproof_spf_for_clist ,
6640   from         .tl_set:N    = \l__stex_sproof_spf_from_tl ,
6641   proofend     .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
6642   type         .str_set_x:N = \spftype,
6643   title        .tl_set:N    = \spftitle,
6644   continues    .tl_set:N    = \l__stex_sproof_spf_continues_tl,
6645   functions    .tl_set:N    = \l__stex_sproof_spf_functions_tl,
6646   term         .tl_set:N    = \l__stex_sproof_spf_term_tl,
6647   method       .tl_set:N    = \l__stex_sproof_spf_method_tl,
6648   hide         .bool_set:N  = \l__stex_sproof_spf_hide_bool
6649 }
6650 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
6651   \str_clear:N \spfid
6652   \tl_clear:N \l__stex_sproof_spf_for_tl
6653   \tl_clear:N \l__stex_sproof_spf_from_tl
6654   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
6655   \str_clear:N \spftype
6656   \tl_clear:N \spftitle
6657   \tl_clear:N \l__stex_sproof_spf_continues_tl
6658   \tl_clear:N \l__stex_sproof_spf_term_tl
6659   \tl_clear:N \l__stex_sproof_spf_functions_tl
6660   \tl_clear:N \l__stex_sproof_spf_method_tl
6661   \bool_set_false:N \l__stex_sproof_spf_hide_bool
6662   \keys_set:nn { stex / spf }{ #1 }
6663 }
6664 \bool_set_true:N \l__stex_sproof_inc_counter_bool
```

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

```
6665 \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.

```
6666 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
6667 \cs_new_protected:Npn \sproofnumber {
6668   \int_set:Nn \l_tmpa_int {1}
6669   \bool_while_do:nn {
6670     \int_compare_p:nNn {
6671       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6672     } > 0
6673   }{
6674     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
6675     \int_incr:N \l_tmpa_int
6676   }
6677 }
6678 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
6679   \int_set:Nn \l_tmpa_int {1}
6680   \bool_while_do:nn {
6681     \int_compare_p:nNn {
6682       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6683     } > 0
6684   }{
6685     \int_incr:N \l_tmpa_int
6686   }
6687   \int_compare:nNnF \l_tmpa_int = 1 {
6688     \int_decr:N \l_tmpa_int
6689   }
6690   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
6691     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1
6692   }
6693 }
6694
6695 \cs_new_protected:Npn \__stex_sproof_add_counter: {
6696   \int_set:Nn \l_tmpa_int {1}
6697   \bool_while_do:nn {
6698     \int_compare_p:nNn {
6699       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6700     } > 0
6701   }{
6702     \int_incr:N \l_tmpa_int
6703   }
6704   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 1 }
6705 }
6706
```



```

6707 \cs_new_protected:Npn \__stex_sproof_remove_counter: {
6708   \int_set:Nn \l_tmpa_int {1}
6709   \bool_while_do:nn {
6710     \int_compare_p:nNn {
6711       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6712     } > 0
6713   }{
6714     \int_incr:N \l_tmpa_int
6715   }
6716   \int_decr:N \l_tmpa_int
6717   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 0 }
6718 }

```

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

```

6719 \def\sproof@box{
6720   \ltx@ifpackageloaded{amssymb}{\square$}{
6721     \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
6722   }
6723 }
6724 \def\sproofend{
6725   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
6726     \hfil\hfill\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
6727   }
6728 }

```

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

**spf@\*kw**

```

6729 \def\spf@proofsketch@kw{Proof~Sketch}
6730 \def\spf@proof@kw{Proof}
6731 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

6732 \AddToHook{begindocument}{
6733   \ltx@ifpackageloaded{babel}{
6734     \makeatletter
6735     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6736     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6737       \input{sproof-ngerman.ldf}
6738     }
6739     \clist_if_in:NnT \l_tmpa_clist {finnish}{
6740       \input{sproof-finnish.ldf}
6741     }
6742     \clist_if_in:NnT \l_tmpa_clist {french}{
6743       \input{sproof-french.ldf}
6744     }
6745     \clist_if_in:NnT \l_tmpa_clist {russian}{
6746       \input{sproof-russian.ldf}
6747     }
6748     \makeatother
6749   }{}
6750 }

```

spfsketch

```

6751 \newcommand\spfsketch[2] [] {
6752   \begin{group}
6753   \let \premise \stex_proof_premise:
6754   \_stex_sproof_spf_args:n{#1}
6755   \stex_if_smsmode:TF {
6756     \str_if_empty:NF \spfid {
6757       \stex_ref_new_doc_target:n \spfid
6758     }
6759   }{
6760     \seq_clear:N \l_tmpa_seq
6761     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6762       \tl_if_empty:nF{ ##1 }{
6763         \stex_get_symbol:n { ##1 }
6764         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6765           \l_stex_get_symbol_uri_str
6766         }
6767       }
6768     }
6769     \exp_args:Nnx
6770     \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {},,}{
6771       \str_if_empty:NF \spftype {
6772         \stex_annotate_invisible:nnn{type}{\spftype}{
6773         }
6774       }
6775       \clist_set:Nn \l_tmpa_clist \spftype
6776       \tl_set:Nn \l_tmpa_tl {
6777         \titleemph{
6778           \tl_if_empty:NTF \spftitle {
6779             \spf@proofsketch@kw
6780           }{
6781             \spftitle
6782           }
6783         }
6784       }
6785       \clist_map_inline:Nn \l_tmpa_clist {
6786         \exp_args:Nn \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6787           \tl_clear:N \l_tmpa_tl
6788         }
6789       }
6790       \str_if_empty:NF \spfid {
6791         \stex_ref_new_doc_target:n \spfid
6792       }
6793       \l_tmpa_tl #2 \sproofend
6794     }
6795   }
6796   \endgroup
6797 }
6798

```

(End definition for *spfsketch*. This function is documented on page 54.)

```

\_stex_sproof_maybe_comment:
\_stex_sproof_maybe_comment_end:
\_stex_sproof_start_comment:
6799 \bool_set_false:N \l__stex_sproof_in_spfblock_bool

```

```

6800
6801 \cs_new_protected:Nn \__stex_sproof_maybe_comment: {
6802   \bool_if:NF \l__stex_sproof_in_spfblock_bool {
6803     \par \setbox \l_tmpa_box \vbox \bgroup \everypar{\__stex_sproof_start_comment:}
6804   }
6805 }
6806 \cs_new_protected:Nn \__stex_sproof_maybe_comment_end: {
6807   \bool_if:NF \l__stex_sproof_in_spfblock_bool { \egroup }
6808 }
6809 \cs_new_protected:Nn \__stex_sproof_start_comment: {
6810   \csname @ @ par\endcsname\egroup\item[]\bgroup\stexcommentfont
6811 }
6812

```

(End definition for \\_\_stex\_sproof\_maybe\_comment:, \\_\_stex\_sproof\_maybe\_comment\_end:, and \\_\_stex\_sproof\_start\_comment:.)

\stexcommentfont

```

6813 \cs_new_protected:Npn \stexcommentfont {
6814   \small\itshape
6815 }

```

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

**sproof (env.)** 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.

```

6816 \cs_new_protected:Nn \__stex_sproof_start_env:nnn {
6817   \seq_clear:N \l_tmpa_seq
6818   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6819     \tl_if_empty:NF{ ##1 }{
6820       \stex_get_symbol:n { ##1 }
6821       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6822         \l_stex_get_symbol_uri_str
6823       }
6824     }
6825   }
6826   \exp_args:Nnnx
6827   \begin{stex_annotate_env}{#1}{\seq_use:Nn \l_tmpa_seq {,}}
6828   \str_if_empty:NF \spftype {
6829     \stex_annotate_invisible:nnn{type}{\spftype}{}
6830   }
6831   #3 {\~\stex_annotate:nnn{spftitle}{}{#2}}
6832   \str_if_empty:NF \spfid {
6833     \stex_ref_new_doc_target:n \spfid
6834   }
6835   \begin{stex_annotate_env}{spfbody}{\bool_if:NTF \l__stex_sproof_spf_hide_bool {false}{true}
6836   \bool_if:NT \l__stex_sproof_spf_hide_bool{
6837     \stex_html_backend:F{\setbox\l_tmpa_box\vbox\bgroup}
6838   }
6839   \begin{list}{}{
6840     \setlength\topsep{0pt}
6841     \setlength\parsep{0pt}
6842     \setlength\rightmargin{0pt}

```

```

6843 } \__stex_sproof_maybe_comment:
6844 }
6845 \cs_new_protected:Nn \__stex_sproof_end_env:n {
6846   \stex_if_smsmode:F{
6847     \__stex_sproof_maybe_comment_end:
6848     \end{list}
6849     \bool_if:NT \l__stex_sproof_spf_hide_bool{
6850       \stex_html_backend:F{\egroup}
6851     }
6852     \clist_set:No \l_tmpa_clist \spftype
6853     #1
6854     \end{stex_annotate_env}
6855     \end{stex_annotate_env}
6856   }
6857 }
6858 }
6859 \NewDocumentEnvironment{sproof}{s O{} m}{
6860   \intarray_gzero:N \l__stex_sproof_counter_intarray
6861   \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
6862   \stex_reactivate_macro:N \yield
6863   \stex_reactivate_macro:N \eqstep
6864   \stex_reactivate_macro:N \assumption
6865   \stex_reactivate_macro:N \conclude
6866   \stex_reactivate_macro:N \spfstep
6867   \__stex_sproof_spf_args:n{#2}
6868   \stex_if_smsmode:TF {
6869     \str_if_empty:NF \spfid {
6870       \stex_ref_new_doc_target:n \spfid
6871     }
6872   }{
6873     \__stex_sproof_start_env:nnn{sproof}{#3}{
6874       \clist_set:No \l_tmpa_clist \spftype
6875       \tl_clear:N \l_tmpa_tl
6876       \clist_map_inline:Nn \l_tmpa_clist {
6877         \tl_if_exist:cT {\__stex_sproof_sproof_##1_start:}{
6878           \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_sproof_sproof_##1_start:}}
6879         }
6880         \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6881           \tl_set:Nn \l_tmpa_tl {\use:n{}}
6882         }
6883       }
6884       \tl_if_empty:NTF \l_tmpa_tl {
6885         \__stex_sproof_sproof_start:
6886       }{
6887         \l_tmpa_tl
6888       }
6889     }
6890   }
6891   \stex_smsmode_do:
6892 }{\__stex_sproof_end_env:n{
6893   \tl_clear:N \l_tmpa_tl
6894   \clist_map_inline:Nn \l_tmpa_clist {
6895     \tl_if_exist:cT {\__stex_sproof_sproof_##1_end:}{
6896       \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_sproof_sproof_##1_end:}}

```

```

6897     }
6898   }
6899   \tl_if_empty:NTF \l_tmpa_tl {
6900     \__stex_sproof_sproof_end:
6901   }{
6902     \l_tmpa_tl
6903   }
6904 }
6905 \NewDocumentEnvironment{subproof}{s O{} m}{
6906   \__stex_sproof_spf_args:n{#2}
6907   \stex_if_smsmode:TF {
6908     \str_if_empty:NF \spfid {
6909       \stex_ref_new_doc_target:n \spfid
6910     }
6911   }{
6912     \__stex_sproof_start_env:nnn{subproof}{\item[\sproofnumber]\ignorespacesandpars #3}{}
6913   }
6914   \__stex_sproof_add_counter:
6915   \stex_smsmode_do:
6916 }{\__stex_sproof_remove_counter:\__stex_sproof_end_env:n{
6917   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6918     \__stex_sproof_inc_counter:
6919   }
6920   \aftergroup\__stex_sproof_maybe_comment:
6921 }
6922 \AddToHook{env/subproof/before}{\__stex_sproof_maybe_comment_end:}
6923
6924 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
6925   \par\noindent\titllemph{
6926     \tl_if_empty:NTF \spftype {
6927       \spf@proof@kw
6928     }{
6929       \spftype
6930     }
6931   }:
6932 }
6933 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
6934
6935 \newcommand\stexpatchproof[3] [] {
6936   \str_set:Nx \l_tmpa_str{ #1 }
6937   \str_if_empty:NTF \l_tmpa_str {
6938     \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
6939     \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
6940   }{
6941     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
6942     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
6943   }
6944 }

\pstep
\conclude
\assumption
\have
\eqstep

```

```

6949 type .str_set_x:N = \spftype,
6950 title .tl_set:N = \spftitle,
6951 method .tl_set:N = \l__stex_sproof_spf_method_tl,
6952 term .tl_set:N = \l__stex_sproof_spf_term_tl
6953 }
6954 \cs_new_protected:Nn \__stex_sproof_spfstep_args:n {
6955 \str_clear:N \spfstepid
6956 \clist_clear:N \l__stex_sproof_spf_for_clist
6957 \str_clear:N \spftype
6958 \tl_clear:N \l__stex_sproof_spf_method_tl
6959 \tl_clear:N \l__stex_sproof_spf_term_tl
6960 %\bool_set_false:N \l__stex_sproof_inc_counter_bool
6961 \keys_set:nn { stex / spfsteps }{ #1 }
6962 }
6963
6964 \cs_new_protected:Nn \__stex_sproof_make_step_macro:Nnnnn {
6965 \NewDocumentCommand #1 {s O{} +m} {
6966 \__stex_sproof_maybe_comment_end:
6967
6968 \__stex_sproof_spfstep_args:n{##2}
6969 \stex_annotate:nnn{spfstep}{#2}{
6970 \tl_if_empty:NF \l__stex_sproof_spf_term_tl {
6971 \stex_annotate_invisible:nnn{spfyield}{}{\l__stex_sproof_spf_term_tl$}
6972 }
6973 \bool_if:NTF \l__stex_sproof_in_spfblock_bool {
6974 #4
6975 }{
6976 \item[\IfBooleanTF ##1 {}{#3}]
6977 }
6978 \ignorespacesandpars ##3
6979 }
6980 \bool_if:NF \l__stex_sproof_in_spfblock_bool { \IfBooleanTF ##1 {}{ #5 } }
6981 \__stex_sproof_maybe_comment:
6982 }
6983 \stex_deactivate_macro:Nn #1 {sproof~environments}
6984 }
6985
6986 \__stex_sproof_make_step_macro:Nnnnn \assumption {assumption} \sproofnumber {} \__stex_sproof
6987 \__stex_sproof_make_step_macro:Nnnnn \conclude {conclusion} {\$ \Rightarrow$} {} {}
6988 \__stex_sproof_make_step_macro:Nnnnn \spfstep {} \sproofnumber {} \__stex_sproof_inc_counter
6989
6990 \NewDocumentCommand \eqstep {s m}{
6991 \__stex_sproof_maybe_comment_end:
6992 \bool_if:NTF \l__stex_sproof_in_spfblock_bool {
6993 $=$
6994 }{
6995 \item[$=$]
6996 }
6997 $\stex_annotate:nnn{spfstep}{eq}{ #2 }$
6998 \__stex_sproof_maybe_comment:
6999 }
7000 \stex_deactivate_macro:Nn \eqstep {sproof~environments}
7001
7002 \NewDocumentCommand \yield {+m}{

```

```

7003 \stex_annotate:nnn{spfyield}{\}{ #1 }
7004 }
7005 \stex_deactivate_macro:Nn \yield {sproof~environments}
7006
7007 \NewDocumentEnvironment{spfblock}{\}{\{
7008 \item[]
7009 \bool_set_true:N \l__stex_sproof_in_spfblock_bool
7010 }\{
7011 \aftergroup\__stex_sproof_maybe_comment:
7012 }
7013 \AddToHook{env/spfblock/before}{\__stex_sproof_maybe_comment_end:}
7014

```

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

### `\spfidea`

```

7015 \NewDocumentCommand\spfidea{0\} +m\{
7016 \__stex_sproof_spf_args:n{#1}
7017 \titleemph{
7018 \tl_if_empty:NTF \spftype {Proof~Idea}{
7019 \spftype
7020 }~#2
7021 }~#2
7022 \sproofend
7023 }

```

(End definition for `\spfidea`. This function is documented on page 54.)

```

7024 \newcommand\spfjust[1]{
7025 #1
7026 }
7027 \</package>

```

Some auxiliary code, and clean up to be executed at the end of the package.

## Chapter 36

# STEX -Others Implementation

```
7028 <*package>
7029
7030 %%%%%%%%%%% others.dtx %%%%%%%%%%%
7031
7032 <@@=stex_others>
    Warnings and error messages
7033 % None

\MSC Math subject classifier

7034 \NewDocumentCommand \MSC {m} {
7035 % TODO
7036 }

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

7037 \@ifpackageloaded{tikzinput}{
7038 \RequirePackage{stex-tikzinput}
7039 }{}
7040
7041 \bool_if:NT \c_stex_persist_mode_bool {
7042 \let__stex_notation_restore_notation_old:nnnnn
7043 \__stex_notation_restore_notation:nnnnn
7044 \def__stex_notation_restore_notation_new:nnnnn#1#2#3#4#5{
7045 \__stex_notation_restore_notation_old:nnnnn{#1}{#2}{#3}{#4}{#5}
7046 \ExplSyntaxOn
7047 }
7048 \def__stex_notation_restore_notation:nnnnn{
7049 \ExplSyntaxOff
7050 \catcode'\sim10
7051 \__stex_notation_restore_notation_new:nnnnn
7052 }
7053 \input{\jobname.sms}
7054 \let__stex_notation_restore_notation:nnnnn
7055 \__stex_notation_restore_notation_old:nnnnn
7056 \prop_if_exist:NT\c_stex_mathhub_main_manifest_prop{
```



```

7057     \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
7058     \l_tmpa_str
7059     \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str_manifest_prop }
7060     \c_stex_mathhub_main_manifest_prop
7061     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
7062   }
7063 }
7064
7065 \stex_get_document_uri:
7066 </package>

```

## Chapter 37

# STEX -Metatheory Implementation

```
7067 <*package>
7068 <@@=stex_modules>
7069
7070 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
7071
7072 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX/meta}
7073 \begingroup
7074 \stex_module_setup:nn{
7075   ns=\c_stex_metatheory_ns_str,
7076   meta=NONE
7077 }{Metatheory}
7078 \stex_reactivate_macro:N \symdecl
7079 \stex_reactivate_macro:N \notation
7080 \stex_reactivate_macro:N \symdef
7081 \ExplSyntaxOff
7082 \csname stex_suppress_html:n\endcsname{
7083   % is-a (a:A, a \in A, a is an A, etc.)
7084   \symdecl{isa}[args=ai]
7085   \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
7086   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
7087   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
7088
7089   % bind (\forall, \Pi, \lambda etc.)
7090   \symdecl{bind}[args=Bi,assoc=pre]
7091   \notation{bind}[depfun,prec=nobrackets,op={(\cdot)\;\to\;\cdot}]{\comp( #1 \comp{}\;\to\;)}
7092   \notation{bind}[forall]{\comp\forall #1.\;#2}{##1 \comp, ##2}
7093   \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
7094
7095   % implicit bind
7096   \symdecl{implicitbind}[args=Bi,assoc=pre]
7097   \notation{implicitbind}[braces,prec=nobrackets,op={(\cdot\_I\;\cdot)}]{\comp\{ #1 \comp{
7098     \notation{implicitbind}[depfun,prec=nobrackets]{\comp( #1 \comp{}\;\to\_I\; } #2}{##1 \comp,
7099     \notation{implicitbind}[Pi]{\comp\prod^I_{#1}#2}{##1\comp,##2}
7100
7101   % dummy variable
```

```

7102 \symdecl{dummyvar}
7103 \notation{dummyvar}[underscore]{\comp\_}
7104 \notation{dummyvar}[dot]{\comp\cdot}
7105 \notation{dummyvar}[dash]{\comp{\rm --}}
7106
7107 %fromto (function space, Hom-set, implication etc.)
7108 \symdecl{fromto}[args=ai]
7109 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
7110 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
7111
7112 % mapto (lambda etc.)
7113 \symdecl{mapto}[args=Bi]
7114 %\notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
7115 %\notation{mapto}[lambda]{\comp\lambda #1 \comp.\; #2}{#1 \comp, #2}
7116 %\notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.\; #2}{#1 \comp, #2}
7117
7118 % function/operator application
7119 \symdecl{apply}[args=ia]
7120 \notation{apply}[prec=0;0x\infpres,parens,op=\cdot(\cdot)]{#1 \comp( #2 \comp)}{##1 \comp,
7121 \notation{apply}[prec=0;0x\infpres,lambda]{#1 \; #2 }{##1 \; ; ##2}
7122
7123 % collection of propositions/booleans/truth values
7124 \symdecl{prop}[name=proposition]
7125 \notation{prop}[prop]{\comp{\rm prop}}
7126 \notation{prop}[BOOL]{\comp{\rm BOOL}}
7127
7128 \symdecl{judgmentholds}[args=1]
7129 \notation{judgmentholds}[vdash,op=\vdash]{\comp\vdash\; #1}
7130
7131 % sequences
7132 \symdecl{seqtype}[args=1]
7133 \notation{seqtype}[kleene]{#1^{\comp\ast}}
7134
7135 \symdecl{seqexpr}[args=a]
7136 \notation{seqexpr}[angle,prec=nobrackets]{\comp\langle #1\comp\rangle}{##1\comp,##2}
7137
7138 \symdef{seqmap}[args=abi,setlike]{\comp\{#3 \comp| #2\comp\in \dobrackets{#1} \comp\}}{##1
7139 \symdef{seqprepend}[args=ia]{#1 \comp{::} #2}{##1 \comp, ##2}
7140 \symdef{seqappend}[args=ai]{#1 \comp{::} #2}{##1 \comp, ##2}
7141 \symdef{seqfoldleft}[args=iabbi]{ \comp{foldl}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\comp}
7142 \symdef{seqfoldright}[args=iabbi,op=foldr]{ \comp{foldr}\dobrackets{#1,#2}\dobrackets{#3\comp
7143 \symdef{seqhead}[args=a]{\comp{head}\dobrackets{#1}}{##1 \comp, ##2}
7144 \symdef{seqtail}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
7145 \symdef{seqlast}[args=a]{\comp{last}\dobrackets{#1}}{##1 \comp, ##2}
7146 \symdef{seqinit}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
7147
7148 \symdef{sequence-index}[args=2,li,prec=nobrackets]{#{#1}_{#2}}
7149 \notation{sequence-index}[ui,prec=nobrackets]{#{#1}^{\#2}}
7150
7151 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{,\ellipses}}{##1\comp,##2}
7152 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{,\ellipses,}#2}{##1\comp,##2}
7153 \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{,\ellipses,}#2\comp{,\ellipses,}#
7154
7155 % nat literals

```

```

7156 \symdef{natliteral}{\comp{\mathtt{Ord}}}
7157
7158 % letin (''let'', local definitions, variable substitution)
7159 \symdecl{letin}[args=bii]
7160 \notation{letin}{let}{\comp{\rm let}}\;#1\comp{=}\#2\;\comp{\rm in}}\;#3}
7161 \notation{letin}{subst}{#3 \comp[ #1 \comp/ #2 \comp]}
7162 \notation{letin}{frac}{#3 \comp[ \frac{#2}{#1} \comp]}
7163
7164 % structures
7165 \symdecl*{module-type}[args=1]
7166 \notation{module-type}{\comp{\mathtt{MOD}}} #1}
7167 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
7168 \notation{mathstruct}[angle,prec=nobrackets]{\comp\angle #1 \comp\rangle}{##1 \comp, ##2}
7169
7170 % objects
7171 \symdecl{object}
7172 \notation{object}{\comp{\mathtt{OBJECT}}}
7173
7174 }
7175
7176 % The following are abbreviations in the sTeX corpus that are left over from earlier
7177 % developments. They will eventually be phased out.
7178
7179 \ExplSyntaxOn
7180 \stex_add_to_current_module:n{
7181   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
7182   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
7183   \def\livar{\csname sequence-index\endcsname[li]}
7184   \def\uivar{\csname sequence-index\endcsname[ui]}
7185   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
7186   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
7187 }
7188 \__stex_modules_end_module:
7189 \endgroup
7190
7191
7192 \str_set:Nn \l_stex_metatheory_str {http://mathhub.info/sTeX/meta?Metatheory}
7193
7194 \NewDocumentCommand \setmetatheory {0{} m}{
7195   \stex_import_module_uri:nn { #1 } { #2 }
7196   \stex_import_require_module:nnnn
7197   { \l_stex_import_ns_str } { \l_stex_import_archive_str }
7198   { \l_stex_import_path_str } { \l_stex_import_name_str }
7199   \str_set:Nx \l_stex_metatheory_str { \l_stex_import_ns_str ? \l_stex_import_name_str }
7200   \stex_smsmode_do:
7201 }
7202
7203 </package>

```

## Chapter 38

# Tikzinput Implementation

```
7204 <@@=tikzinput>
7205 <*package>
7206
7207 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
7208
7209 \ProvidesExplPackage{tikzinput}{2022/09/14}{3.2.0}{tikzinput package}
7210 \RequirePackage{l3keys2e}
7211
7212 \keys_define:nn { tikzinput } {
7213   image .bool_set:N = \c_tikzinput_image_bool,
7214   image .default:n = false ,
7215   unknown .code:n = {}
7216 }
7217
7218 \ProcessKeysOptions { tikzinput }
7219
7220 \bool_if:NTF \c_tikzinput_image_bool {
7221   \RequirePackage{graphicx}
7222
7223   \providecommand\usetikzlibrary[]{}
7224   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
7225 }{
7226   \RequirePackage{tikz}
7227   \RequirePackage{standalone}
7228
7229   \newcommand \tikzinput [2] [] {
7230     \setkeys{Gin}{#1}
7231     \ifx \Gin@ewidth \Gin@exclamation
7232       \ifx \Gin@eheight \Gin@exclamation
7233         \input { #2 }
7234       \else
7235         \resizebox{!}{ \Gin@eheight }{
7236           \input { #2 }
7237         }
7238       \fi
7239     \else
7240       \ifx \Gin@eheight \Gin@exclamation
7241         \resizebox{ \Gin@ewidth }{!}{
```

```

7242         \input { #2 }
7243     }
7244     \else
7245         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
7246             \input { #2 }
7247         }
7248     \fi
7249 \fi
7250 }
7251 }
7252
7253 \newcommand \ctikzinput [2] [] {
7254     \begin{center}
7255         \tikzinput [#1] {#2}
7256     \end{center}
7257 }
7258
7259 \@ifpackageloaded{stex}{
7260     \RequirePackage{stex-tikzinput}
7261 }{}
7262
7263 </package>
7264 <*stex>
7265 \ProvidesExplPackage{stex-tikzinput}{2022/09/14}{3.2.0}{stex-tikzinput}
7266 \RequirePackage{stex}
7267 \RequirePackage{tikzinput}
7268
7269 \newcommand\mhtikzinput[2] [] {%
7270     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
7271     \stex_in_repository:nn\Gin@mhrepos{
7272         \tikzinput[#1]{\mhp@hpath{##1}{#2}}
7273     }
7274 }
7275 \newcommand\cmhtikzinput[2] [] {\begin{center}\mhtikzinput[#1]{#2}\end{center}}
7276
7277 \cs_new_protected:Nn \__tikzinput_usetikzlibrary:nn {
7278     \pgfkeys@spdef\pgf@temp{#1}
7279     \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
7280     \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgf@temp
7281     \expandafter\edef\csname tikz@library@#1@atcode\endcsname{\the\catcode'\@}
7282     \expandafter\edef\csname tikz@library@#1@barcode\endcsname{\the\catcode'\|}
7283     \expandafter\edef\csname tikz@library@#1@dollarcode\endcsname{\the\catcode'\$}
7284     \catcode'\@=11
7285     \catcode'\|=12
7286     \catcode'\$=3
7287     \pgfutil@InputIfFileExists{#2}{-}{-}
7288     \catcode'\@=\csname tikz@library@#1@atcode\endcsname
7289     \catcode'\|=\csname tikz@library@#1@barcode\endcsname
7290     \catcode'\$=\csname tikz@library@#1@dollarcode\endcsname
7291 }
7292
7293
7294 \newcommand\libusetikzlibrary[1]{

```

```

7295 \prop_if_exist:NF \l_stex_current_repository_prop {
7296   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
7297 }
7298 \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
7299   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
7300 }
7301 \seq_clear:N \l__tikzinput_libinput_files_seq
7302 \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
7303 \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
7304
7305 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
7306   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / tikzlibrary
7307   \IfFileExists{ \l_tmpa_str }{
7308     \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
7309   }{}
7310   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
7311   \seq_put_right:No \l_tmpa_seq \l_tmpa_str
7312 }
7313
7314 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / tikzlibrary #1 .code.t
7315 \IfFileExists{ \l_tmpa_str }{
7316   \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
7317 }{}
7318
7319 \seq_if_empty:NTF \l__tikzinput_libinput_files_seq {
7320   \msg_error:nxxx{stex}{error/nofile}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .code.t
7321 }{
7322   \int_compare:nNnTF {\seq_count:N \l__tikzinput_libinput_files_seq} = 1 {
7323     \seq_map_inline:Nn \l__tikzinput_libinput_files_seq {
7324       \__tikzinput_usetikzlibrary:nn{#1}{ ##1 }
7325     }
7326   }{
7327     \msg_error:nxxx{stex}{error/twofiles}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .co
7328   }
7329 }
7330 }
7331 </stex>

```

## Chapter 39

# document-structure.sty Implementation

```
7332 <*package>
7333 <@@=document_structure>
7334 \ProvidesExplPackage{document-structure}{2022/09/14}{3.2.0}{Modular Document Structure}
7335 \RequirePackage{13keys2e}
```

### 39.1 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).

```
7336
7337 \keys_define:nn{ document-structure }{
7338   class      .str_set_x:N = \c_document_structure_class_str,
7339   topsect    .str_set_x:N = \c_document_structure_topsect_str,
7340   unknown    .code:n      = {
7341     \PassOptionsToClass{\CurrentOption}{stex}
7342     \PassOptionsToClass{\CurrentOption}{tikzinput}
7343   }
7344   % showignores .bool_set:N = \c_document_structure_showignores_bool,
7345 }
7346 \ProcessKeysOptions{ document-structure }
7347 \str_if_empty:NT \c_document_structure_class_str {
7348   \str_set:Nn \c_document_structure_class_str {article}
7349 }
7350 \str_if_empty:NT \c_document_structure_topsect_str {
7351   \str_set:Nn \c_document_structure_topsect_str {section}
7352 }
```

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

```
7353 \RequirePackage{xspace}
7354 \RequirePackage{comment}
7355 \RequirePackage{stex}
7356 \AddToHook{begindocument}{
```



```

7357 \ltx@ifpackageloaded{babel}{
7358   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7359   \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
7360     \makeatletter\input{document-structure-ngerman.ldf}\makeatother
7361   }
7362 }{}
7363 }

```

`\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.

```

7364 \int_new:N \l_document_structure_section_level_int
7365 \str_case:NnF \c_document_structure_topsect_str {
7366   {part}}{
7367     \int_set:Nn \l_document_structure_section_level_int {0}
7368   }
7369   {chapter}{
7370     \int_set:Nn \l_document_structure_section_level_int {1}
7371   }
7372 }{
7373   \str_case:NnF \c_document_structure_class_str {
7374     {book}{
7375       \int_set:Nn \l_document_structure_section_level_int {0}
7376     }
7377     {report}{
7378       \int_set:Nn \l_document_structure_section_level_int {0}
7379     }
7380   }{
7381     \int_set:Nn \l_document_structure_section_level_int {2}
7382   }
7383 }

```

## 39.2 Document Structure

The structure of the document is given by the `sfragment` environment. The hierarchy is adjusted automatically according to the L<sup>A</sup>T<sub>E</sub>X class in effect.

`\currentsectionlevel` For the `\currentsectionlevel` and `\Currentsectionlevel` macros we use an internal macro `\current@section@level` that only contains the keyword (no markup). We initialize it with “document” as a default. In the generated OMDoc, we only generate a text element of class `omdoc_currentsectionlevel`, which will be instantiated by CSS later.<sup>9</sup>

EdN:9

```

7384 \def\current@section@level{document}%
7385 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
7386 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%

```

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

`\skipfragment`

```

7387 \cs_new_protected:Npn \skipfragment {

```

---

<sup>9</sup>EdNOTE: MK: we may have to experiment with the more powerful uppercasing macro from `mfirstuc.sty` once we internationalize.

```

7388 \ifcase\l_document_structure_section_level_int
7389 \or\stepcounter{part}
7390 \or\stepcounter{chapter}
7391 \or\stepcounter{section}
7392 \or\stepcounter{subsection}
7393 \or\stepcounter{subsubsection}
7394 \or\stepcounter{paragraph}
7395 \or\stepcounter{subparagraph}
7396 \fi
7397 }

```

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

`blindfragment (env.)`

```

7398 \newcommand\at@begin@blindsfragment[1]{
7399 \newenvironment{blindfragment}
7400 {
7401 \int_incr:N\l_document_structure_section_level_int
7402 \at@begin@blindsfragment\l_document_structure_section_level_int
7403 }{}

```

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

```

7404 \newcommand\sfragment@nonum[2]{
7405 \ifx\hyper@anchor\@undefined\else\phantomsection\fi
7406 \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
7407 }

```

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

`\sfragment@num` convenience macro: `\sfragment@num{<level>}{<title>}` makes numbered sectioning with title `<title>` at level `<level>`. We have to check the `short` key was given in the `sfragment` environment and – if it is use it. But how to do that depends on whether the `rdfmata` package has been loaded. In the end we call `\sref@label@id` to enable crossreferencing.

```

7408 \newcommand\sfragment@num[2]{
7409 \tl_if_empty:NTF \l__document_structure_sfragment_short_tl {
7410 \@nameuse{#1}{#2}
7411 }{
7412 \cs_if_exist:NTF\rdfmata@sectioning{
7413 \@nameuse{rdfmata@#1@old}[\l__document_structure_sfragment_short_tl]{#2}
7414 }{
7415 \@nameuse{#1}[\l__document_structure_sfragment_short_tl]{#2}
7416 }
7417 }
7418 %\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\sfragment@id
7419 }

```

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

`sfragment (env.)`

```

7420 \keys_define:nn { document-structure / sfragment }{
7421 id .str_set_x:N = \l__document_structure_sfragment_id_str,
7422 date .str_set_x:N = \l__document_structure_sfragment_date_str,

```

```

7423 creators      .clist_set:N = \l__document_structure_sfragment_creators_clist,
7424 contributors   .clist_set:N = \l__document_structure_sfragment_contributors_clist,
7425 srccite        .tl_set:N    = \l__document_structure_sfragment_srccite_tl,
7426 type          .tl_set:N    = \l__document_structure_sfragment_type_tl,
7427 short         .tl_set:N    = \l__document_structure_sfragment_short_tl,
7428 intro         .tl_set:N    = \l__document_structure_sfragment_intro_tl,
7429 imports       .tl_set:N    = \l__document_structure_sfragment_imports_tl,
7430 loadmodules    .bool_set:N  = \l__document_structure_sfragment_loadmodules_bool
7431 }
7432 \cs_new_protected:Nn \l__document_structure_sfragment_args:n {
7433   \str_clear:N \l__document_structure_sfragment_id_str
7434   \str_clear:N \l__document_structure_sfragment_date_str
7435   \clist_clear:N \l__document_structure_sfragment_creators_clist
7436   \clist_clear:N \l__document_structure_sfragment_contributors_clist
7437   \tl_clear:N \l__document_structure_sfragment_srccite_tl
7438   \tl_clear:N \l__document_structure_sfragment_type_tl
7439   \tl_clear:N \l__document_structure_sfragment_short_tl
7440   \tl_clear:N \l__document_structure_sfragment_imports_tl
7441   \tl_clear:N \l__document_structure_sfragment_intro_tl
7442   \bool_set_false:N \l__document_structure_sfragment_loadmodules_bool
7443   \keys_set:nn { document-structure / sfragment } { #1 }
7444 }

```

we define a switch for numbering lines and a hook for the beginning of groups: The `\at@begin@sfragment` macro allows customization. It is run at the beginning of the `sfragment`, i.e. after the section heading.

```

7445 \newif\if@mainmatter\@mainmattertrue
7446 \newcommand\at@begin@sfragment[3][]{ }

```

Then we define a helper macro that takes care of the sectioning magic. It comes with its own key/value interface for customization.

```

7447 \keys_define:nn { document-structure / sectioning }{
7448   name      .str_set_x:N = \l__document_structure_sect_name_str ,
7449   ref       .str_set_x:N = \l__document_structure_sect_ref_str  ,
7450   clear     .bool_set:N  = \l__document_structure_sect_clear_bool ,
7451   clear     .default:n   = {true} ,
7452   num       .bool_set:N  = \l__document_structure_sect_num_bool  ,
7453   num       .default:n   = {true}
7454 }
7455 \cs_new_protected:Nn \l__document_structure_sect_args:n {
7456   \str_clear:N \l__document_structure_sect_name_str
7457   \str_clear:N \l__document_structure_sect_ref_str
7458   \bool_set_false:N \l__document_structure_sect_clear_bool
7459   \bool_set_false:N \l__document_structure_sect_num_bool
7460   \keys_set:nn { document-structure / sectioning } { #1 }
7461 }
7462 \newcommand\omdoc@sectioning[3][]{
7463   \l__document_structure_sect_args:n {#1 }
7464   \let\omdoc@sect@name\l__document_structure_sect_name_str
7465   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
7466   \if@mainmatter% numbering not overridden by frontmatter, etc.
7467     \bool_if:NTF \l__document_structure_sect_num_bool {
7468       \sfragment@num{#2}{#3}
7469     }{

```

```

7470     \sfragment@nonum{#2}{#3}
7471   }
7472   \def\current@section@level{\omdoc@sect@name}
7473   \else
7474     \sfragment@nonum{#2}{#3}
7475   \fi
7476 }% 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.

```

7477 \newcommand\sfragment@redefine@addtocontents[1]{%
7478 %\edef\__document_structureimport{#1}%
7479 %\@for\@I:=\__document_structureimport\do{%
7480 %\edef\@path{\csname module@\@I @path\endcsname}%
7481 %\@ifundefined{tf@toc}\relax%
7482 %   {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
7483 %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
7484 %\def\addcontentsline##1##2##3{%
7485 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
7486 %\else% hyperref.sty not loaded
7487 %\def\addcontentsline##1##2##3{%
7488 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
7489 %\fi
7490 }% hypreref.sty loaded?

```

now the `sfragment` environment itself. This takes care of the table of contents via the helper macro above and then selects the appropriate sectioning command from `article.cls`. It also registers the current level of sfragments in the `\sfragment@level` counter.

```

7491 \newenvironment{sfragment}[2][ ]% keys, title
7492 {
7493   \__document_structure_sfragment_args:n { #1 }\sref@target%

```

If the `loadmodules` key is set on `\begin{sfragment}`, we redefine the `\addcontetsline` macro that determines how the sectioning commands below construct the entries for the table of contents.

```

7494   \stex_csl_to_imports:No \usemodule \l__document_structure_sfragment_imports_tl
7495
7496   \bool_if:NT \l__document_structure_sfragment_loadmodules_bool {
7497     \sfragment@redefine@addtocontents{
7498       %\@ifundefined{module@id}\used@modules%
7499       %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}
7500     }
7501   }

```

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

```

7502
7503   \stex_document_title:n { #2 }
7504
7505   \stex_patch_counters:
7506   \int_incr:N\l__document_structure_section_level_int
7507   \ifcase\l__document_structure_section_level_int
7508     \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
7509     \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
7510     \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}

```

```

7511 \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
7512 \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
7513 \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#1}
7514 \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#1}
7515 \fi
7516 \at@begin@sfragment[#1]\l__document_structure_section_level_int{#2}
7517 \str_if_empty:NF \l__document_structure_sfragment_id_str {
7518   \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
7519 }
7520 \stex_unpatch_counters:
7521 }% for customization
7522 {}

```

and finally, we localize the sections

```

7523 \newcommand\omdoc@part@kw{Part}
7524 \newcommand\omdoc@chapter@kw{Chapter}
7525 \newcommand\omdoc@section@kw{Section}
7526 \newcommand\omdoc@subsection@kw{Subsection}
7527 \newcommand\omdoc@subsubsection@kw{Subsubsection}
7528 \newcommand\omdoc@paragraph@kw{paragraph}
7529 \newcommand\omdoc@subparagraph@kw{subparagraph}

```

### 39.3 Front and Backmatter

Index markup is provided by the `omtext` package [[Kohlhase:smmf:git](#)], so in the `document-structure` package we only need to supply the corresponding `\printindex` command, if it is not already defined

`\printindex`

```

7530 \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).

```

7531 \cs_if_exist:NTF\frontmatter{
7532   \let\__document_structure_orig_frontmatter\frontmatter
7533   \let\frontmatter\relax
7534 }{
7535   \tl_set:Nn\__document_structure_orig_frontmatter{
7536     \clearpage
7537     \@mainmatterfalse
7538     \pagenumbering{roman}
7539   }
7540 }
7541 \cs_if_exist:NTF\backmatter{
7542   \let\__document_structure_orig_backmatter\backmatter
7543   \let\backmatter\relax
7544 }{
7545   \tl_set:Nn\__document_structure_orig_backmatter{
7546     \clearpage
7547     \@mainmatterfalse

```

```

7548     \pagenumbering{roman}
7549   }
7550 }

```

Using these, we can now define the `frontmatter` and `backmatter` environments

`frontmatter (env.)` we use the `\orig@frontmatter` macro defined above and `\mainmatter` if it exists, otherwise we define it.

```

7551 \newenvironment{frontmatter}{
7552   \_document_structure_orig_frontmatter
7553 }{
7554   \cs_if_exist:NTF\mainmatter{
7555     \mainmatter
7556   }{
7557     \clearpage
7558     \@mainmattertrue
7559     \pagenumbering{arabic}
7560   }
7561 }

```

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

```

7562 \newenvironment{backmatter}{
7563   \_document_structure_orig_backmatter
7564 }{
7565   \cs_if_exist:NTF\mainmatter{
7566     \mainmatter
7567   }{
7568     \clearpage
7569     \@mainmattertrue
7570     \pagenumbering{arabic}
7571   }
7572 }

```

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

```

7573 \@mainmattertrue\pagenumbering{arabic}

```

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

```

7574 \def \c__document_structure_document_str{document}
7575 \newcommand\afterprematurestop{}
7576 \def\prematurestop@endsfragment{
7577   \unless\ifx\@currenvir\c__document_structure_document_str
7578     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
7579     \expandafter\prematurestop@endsfragment
7580   }
7581 }
7582 \providecommand\prematurestop{
7583   \message{Stopping~sTeX~processing~prematurely}
7584   \prematurestop@endsfragment
7585   \afterprematurestop
7586   \end{document}
7587 }

```

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

## 39.4 Global Variables

**\setSGvar** set a global variable

```
7588 \RequirePackage{etoolbox}
7589 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}
```

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

**\useSGvar** use a global variable

```
7590 \newrobustcmd\useSGvar[1]{%
7591   \@ifundefined{sTeX@Gvar@#1}
7592   {\PackageError{document-structure}
7593    {The sTeX Global variable #1 is undefined}
7594    {set it with \protect\setSGvar}}
7595   \@nameuse{sTeX@Gvar@#1}}
```

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

**\ifSGvar** execute something conditionally based on the state of the global variable.

```
7596 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
7597   \@ifundefined{sTeX@Gvar@#1}
7598   {\PackageError{document-structure}
7599    {The sTeX Global variable #1 is undefined}
7600    {set it with \protect\setSGvar}}
7601   {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}
```

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

## Chapter 40

# NotesSlides – Implementation

### 40.1 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.

```
7602 \*cls)
7603 \@@=notesslides)
7604 \ProvidesExplClass{notesslides}{2022/09/14}{3.2.0}{notesslides Class}
7605 \RequirePackage{13keys2e}
7606
7607 \keys_define:nn{notesslides / cls}{
7608   class .str_set_x:N = \c__notesslides_class_str,
7609   notes .bool_set:N = \c__notesslides_notes_bool ,
7610   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
7611   docopt .str_set_x:N = \c__notesslides_docopt_str,
7612   unknown .code:n = {
7613     \PassOptionsToPackage{\CurrentOption}{document-structure}
7614     \PassOptionsToClass{\CurrentOption}{beamer}
7615     \PassOptionsToPackage{\CurrentOption}{notesslides}
7616     \PassOptionsToPackage{\CurrentOption}{stex}
7617   }
7618 }
7619 \ProcessKeysOptions{ notesslides / cls }
7620
7621 \str_if_empty:NF \c__notesslides_class_str {
7622   \PassOptionsToPackage{class=\c__notesslides_class_str}{document-structure}
7623 }
7624
7625 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{book}{
7626   \PassOptionsToPackage{defaulttopsect=part}{notesslides}
7627 }
7628 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{report}{
7629   \PassOptionsToPackage{defaulttopsect=part}{notesslides}
7630 }
7631
7632 \RequirePackage{stex}
```



```

7633 \stex_html_backend:T {
7634   \bool_set_true:N\c__notesslides_notes_bool
7635 }
7636
7637 \bool_if:NTF \c__notesslides_notes_bool {
7638   \PassOptionsToPackage{notes=true}{notesslides}
7639   \message{notesslides.cls:~Formatting~course~materials~in~notes~mode}
7640 }{
7641   \PassOptionsToPackage{notes=false}{notesslides}
7642   \message{notesslides.cls:~Formatting~course~materials~in~slides~mode}
7643 }
7644 \</cls>

```

now we do the same for the notesslides package.

```

7645 <*package>
7646 \ProvidesExplPackage{notesslides}{2022/09/14}{3.2.0}{notesslides Package}
7647 \RequirePackage{l3keys2e}
7648
7649 \keys_define:nn{notesslides / pkg}{
7650   topsect          .str_set_x:N = \c__notesslides_topsect_str,
7651   defaulttopsect   .str_set_x:N = \c__notesslides_defaulttopsec_str,
7652   notes            .bool_set:N = \c__notesslides_notes_bool ,
7653   slides           .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
7654   sectocframes     .bool_set:N = \c__notesslides_sectocframes_bool ,
7655   frameimages      .bool_set:N = \c__notesslides_frameimages_bool ,
7656   fiboxed          .bool_set:N = \c__notesslides_fiboxed_bool ,
7657   nopproblems      .bool_set:N = \c__notesslides_nopproblems_bool,
7658   unknown          .code:n      = {
7659     \PassOptionsToClass{\CurrentOption}{stex}
7660     \PassOptionsToClass{\CurrentOption}{tikzinput}
7661   }
7662 }
7663 \ProcessKeysOptions{ notesslides / pkg }
7664
7665 \RequirePackage{stex}
7666 \stex_html_backend:T {
7667   \bool_set_true:N\c__notesslides_notes_bool
7668 }
7669
7670 \newif\ifnotes
7671 \bool_if:NTF \c__notesslides_notes_bool {
7672   \notesttrue
7673 }{
7674   \notesfalse
7675 }
7676

```

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

```

7677 \str_if_empty:NTF \c__notesslides_topsect_str {
7678   \str_set_eq:NN \__notesslides_topsect \c__notesslides_defaulttopsec_str
7679 }{
7680   \str_set_eq:NN \__notesslides_topsect \c__notesslides_topsect_str
7681 }
7682 \PassOptionsToPackage{topsect=\__notesslides_topsect}{document-structure}

```

```
7683 </package>
```

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

```
7684 <*cls>
7685 \bool_if:NTF \c__notesslides_notes_bool {
7686   \str_if_empty:NT \c__notesslides_class_str {
7687     \str_set:Nn \c__notesslides_class_str {article}
7688   }
7689   \exp_after:wN\LoadClass\exp_after:wN[\c__notesslides_dcopt_str]
7690     {\c__notesslides_class_str}
7691 }{
7692   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
7693   \newcounter{Item}
7694   \newcounter{paragraph}
7695   \newcounter{subparagraph}
7696   \newcounter{Hfootnote}
7697 }
7698 \RequirePackage{document-structure}
```

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

```
7699 \RequirePackage{notesslides}
7700 </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  $\text{\TeX}$  packages. The first batch of packages we want are loaded on `notesslides.sty`. These are the general ones, we will load the  $\text{\TeX}$ -specific ones after we have done some work (e.g. defined the counters `m*`). Only the `stex-logo` package is already needed now for the default theme.

```
7701 <*package>
7702 \bool_if:NT \c__notesslides_notes_bool {
7703   \RequirePackage{a4wide}
7704   \RequirePackage{marginnote}
7705   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
7706   \RequirePackage{mdframed}
7707   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
7708   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
7709 }
7710 \RequirePackage{stex-tikzinput}
7711 \RequirePackage{comment}
7712 \RequirePackage{url}
7713 \RequirePackage{graphicx}
7714 \RequirePackage{pgf}
7715 \RequirePackage{bookmark}
```

## 40.2 Notes and Slides

For the lecture notes cases, we also provide the `\usetheme` macro that would otherwise come from the `beamer` class.

```
7716 \bool_if:NT \c__notesslides_notes_bool {
7717   \renewcommand\usetheme[2][\usepackage{#1}{beamertheme#2}]
7718 }
```

```

7719 \NewDocumentCommand \libusetheme {0{} m} {
7720   \libusepackage[#1]{beamertheme#2}
7721 }
7722

```

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

```

7723 \newcounter{slide}
7724 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
7725 \newlength{\slideheight}\setlength{\slideheight}{9cm}

```

**note** (*env.*) 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.

```

7726 \bool_if:NTF \c__notesslides_notes_bool {
7727   \renewenvironment{note}{\ignorespaces}{}
7728 }{
7729   \excludecomment{note}
7730 }

```

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.

```

7731 \bool_if:NT \c__notesslides_notes_bool {
7732   \newlength{\slideframewidth}
7733   \setlength{\slideframewidth}{1.5pt}

```

**frame** (*env.*) We first define the keys.

```

7734 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
7735   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
7736     \bool_set_true:N #1
7737   }{
7738     \bool_set_false:N #1
7739   }
7740 }
7741 \keys_define:nn{notesslides / frame}{
7742   label .str_set_x:N = \l__notesslides_frame_label_str,
7743   allowframebreaks .code:n = {
7744     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
7745   },
7746   allowdisplaybreaks .code:n = {
7747     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
7748   },
7749   fragile .code:n = {
7750     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
7751   },
7752   shrink .code:n = {
7753     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
7754   },
7755   squeeze .code:n = {
7756     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
7757   },
7758   t .code:n = {

```

```

7759     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
7760   },
7761   unknown    .code:n      = {}
7762 }
7763 \cs_new_protected:Nn \__notesslides_frame_args:n {
7764   \str_clear:N \l__notesslides_frame_label_str
7765   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
7766   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
7767   \bool_set_true:N \l__notesslides_frame_fragile_bool
7768   \bool_set_true:N \l__notesslides_frame_shrink_bool
7769   \bool_set_true:N \l__notesslides_frame_squeeze_bool
7770   \bool_set_true:N \l__notesslides_frame_t_bool
7771   \keys_set:nn { notesslides / frame }{ #1 }
7772 }

```

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

```

7773 \renewenvironment{frame}[1][]{
7774   \__notesslides_frame_args:n{#1}
7775   \sffamily
7776   \stepcounter{slide}
7777   \def\@currentlabel{\theslide}
7778   \str_if_empty:NF \l__notesslides_frame_label_str {
7779     \label{\l__notesslides_frame_label_str}
7780   }

```

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

```

7781   \def\itemize@level{outer}
7782   \def\itemize@outer{outer}
7783   \def\itemize@inner{inner}
7784   \renewcommand\newpage{\addtocounter{framenum}{1}}
7785   %\newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
7786   \renewenvironment{itemize}{
7787     \ifx\itemize@level\itemize@outer
7788       \def\itemize@label{$\rhd$}
7789     \fi
7790     \ifx\itemize@level\itemize@inner
7791       \def\itemize@label{$\scriptstyle\rhd$}
7792     \fi
7793     \begin{list}
7794       {\itemize@label}
7795       {\setlength{\labelsep}{.3em}
7796        \setlength{\labelwidth}{.5em}
7797        \setlength{\leftmargin}{1.5em}
7798       }
7799     \edef\itemize@level{\itemize@inner}
7800   }{
7801     \end{list}
7802   }

```

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

```

7803   \stex_html_backend:TF {
7804     \begin{stex_annotate_env}{frame}{}\vbox\bgroup
7805     \mdf@patchamsthm
7806   }{
7807     \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwid

```

```

7808     }
7809   }{
7810     \stex_html_backend:TF {
7811       \miko@slidelabel\egroup\end{stex_annotate_env}
7812     }\medskip\miko@slidelabel\end{mdframed}}
7813   }

```

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

`\frametitle`

```

7814   \renewcommand{\frametitle}[1]{
7815     \stex_document_title:n { #1 }
7816     {\Large\bf\sf\color{blue}{#1}}\medskip
7817   }
7818 }

```

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

EdN:10

`\pause` 10

```

7819 \bool_if:NT \c__notesslides_notes_bool {
7820   \newcommand\pause{}
7821 }

```

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

`nparagraph (env.)`

```

7822 \bool_if:NTF \c__notesslides_notes_bool {
7823   \newenvironment{nparagraph}[1] [] {\begin{sparagraph}[#1]}\end{sparagraph}}
7824 }{
7825   \excludecomment{nparagraph}
7826 }

```

`nfragment (env.)`

```

7827 \bool_if:NTF \c__notesslides_notes_bool {
7828   \newenvironment{nfragment}[2] [] {\begin{sfragment}[#1]{#2}}\end{sfragment}}
7829 }{
7830   \excludecomment{nfragment}
7831 }

```

`ndefinition (env.)`

```

7832 \bool_if:NTF \c__notesslides_notes_bool {
7833   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}\end{sdefinition}}
7834 }{
7835   \excludecomment{ndefinition}
7836 }

```

`nassertion (env.)`

```

7837 \bool_if:NTF \c__notesslides_notes_bool {
7838   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}\end{sassertion}}
7839 }{
7840   \excludecomment{nassertion}
7841 }

```

---

<sup>10</sup>EDNOTE: MK: fake it in notes mode for now

`nsproof (env.)`

```
7842 \bool_if:NTF \c__notesslides_notes_bool {
7843   \newenvironment{nproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
7844 }{
7845   \excludecomment{nproof}
7846 }
```

`nexample (env.)`

```
7847 \bool_if:NTF \c__notesslides_notes_bool {
7848   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}{\end{sexample}}
7849 }{
7850   \excludecomment{nexample}
7851 }
```

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

```
7852 \def\inputref@preskip{\smallskip}
7853 \def\inputref@postskip{\medskip}
```

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

**`\inputref*`**

```
7854 \let\orig@inputref\inputref
7855 \def\inputref{\@ifstar\ninputref\orig@inputref}
7856 \newcommand\ninputref[2] []{
7857   \bool_if:NT \c__notesslides_notes_bool {
7858     \orig@inputref[#1]{#2}
7859   }
7860 }
```

*(End definition for \inputref\*. This function is documented on page 64.)*

## 40.3 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 `STEX` logo. Customization can be done by `\setslidelogo{<logo name>}`.

```
7861 \newlength{\slidelogoheight}
7862
7863 \RequirePackage{graphicx}
7864
7865 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7866 \providecommand\mhgraphics[2] []{
7867   \def\Gin@mhrepos{}\setkeys{Gin}{#1}
7868   \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}
7869 }
7870
7871 \bool_if:NTF \c__notesslides_notes_bool {
7872   \setlength{\slidelogoheight}{.4cm}
7873 }{
7874   \setlength{\slidelogoheight}{.25cm}
7875 }
```

```

7876 \ifcsname slidelogo\endcsname\else
7877 \newsavebox{\slidelogo}
7878 \sbox{\slidelogo}{\sTeX}
7879 \fi
7880 \newrobustcmd{\setslidelogo}[2][\{
7881 \tl_if_empty:nTF{#1}{
7882 \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#2}}
7883 }{
7884 \sbox{\slidelogo}{\mhgraphics[height=\slidelogoheight,mhrepos=#1]{#2}}
7885 }
7886 }

```

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

**\author** In notes mode, we redefine the \author macro so that it does not disregard the optional argument (as beamerarticle does). We want to use it to set the source later.

```

7887 \bool_if:NT \c__notesslides_notes_bool {
7888 \def\author{\@dblarg\@ns@author}
7889 \long\def\@ns@author[#1]#2{%
7890 \def\c__notesslides_shortauthor{#1}%
7891 \def\@author{#2}
7892 }
7893 }

```

(End definition for \author. 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.

```

7894 \newrobustcmd{\setsource}[1]{\def\source{#1}}

```

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

**\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.

```

7895 \def\copyrightnotice{%
7896 \footnotesize\copyright : \hspace{.3ex}%
7897 \ifcsname source\endcsname\source\else%
7898 \ifcsname c__notesslides_shortauthor\endcsname\c__notesslides_shortauthor\else%
7899 \PackageWarning{notesslides}{Author/Source-undefined-in-copyright-notice}%
7900 ?source/author?\fi%
7901 \fi}
7902 \newsavebox{\cclogo}
7903 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
7904 \newif\ifcchref\cchreffalse
7905 \AtBeginDocument{
7906 \@ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
7907 }
7908 \def\licensing{
7909 \ifcchref
7910 \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
7911 \else
7912 {\usebox{\cclogo}}

```

```

7913 \fi
7914 }
7915 \newrobustcmd{\setlicensing}[2][]{
7916   \def\@url{#1}
7917   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
7918   \ifx\@url\@empty
7919     \def\licensing{\usebox{\cclogo}}
7920   \else
7921     \def\licensing{
7922       \ifcchref
7923         \href{#1}{\usebox{\cclogo}}
7924       \else
7925         {\usebox{\cclogo}}
7926     \fi
7927   }
7928 \fi
7929 }

```

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

EdN:11

```

\slidelabel Now, we set up the slide label for the article mode.11
7930 \newrobustcmd\miko@slidelabel{
7931   \vbox to \slidelogoheight{
7932     \vss\hbox to \slidewidth
7933       {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
7934   }
7935 }

```

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

## 40.4 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.

```

7936 \def\Gin@mhrepos{}
7937 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7938 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
7939 \newrobustcmd\frameimage[2][]{
7940   \stepcounter{slide}
7941   \bool_if:NT \c__notesslides_frameimages_bool {
7942     \def\Gin@ewidth{}\setkeys{Gin}{#1}
7943     \bool_if:NF \c__notesslides_notes_bool { \vfill }
7944     \begin{center}
7945       \bool_if:NTF \c__notesslides_fiboxed_bool {
7946         \fbox{
7947           \ifx\Gin@ewidth\@empty
7948             \ifx\Gin@mhrepos\@empty
7949               \mhgraphics[width=\slidewidth,#1]{#2}
7950             \else
7951               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7952             \fi
7953           \else% Gin@ewidth empty

```

<sup>11</sup>EdNOTE: see that we can use the themes for the slides some day. This is all fake.



```

7954         \ifx\Gin@mhrepos\@empty
7955         \mhgraphics[#1]{#2}
7956     \else
7957         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7958     \fi
7959 \fi% Gin@ewidth empty
7960 }
7961 }{
7962     \ifx\Gin@ewidth\@empty
7963     \ifx\Gin@mhrepos\@empty
7964         \mhgraphics[width=\slidewidth,#1]{#2}
7965     \else
7966         \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7967     \fi
7968     \ifx\Gin@mhrepos\@empty
7969         \mhgraphics[#1]{#2}
7970     \else
7971         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7972     \fi
7973     \fi% Gin@ewidth empty
7974 }
7975 \end{center}
7976 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
7977 \bool_if:NF \c__notesslides_notes_bool { \vfill }
7978 }
7979 } % ifmks@sty@frameimages

```

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

## 40.5 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`.

```

7980 \stex_html_backend:F {
7981     \bool_if:NT \c__notesslides_sectocframes_bool {
7982         \str_if_eq:VnTF \__notesslidestopsect{part}{
7983             \newcounter{chapter}\counterwithin*{section}{chapter}
7984         }{
7985             \str_if_eq:VnT\__notesslidestopsect{chapter}{
7986                 \newcounter{chapter}\counterwithin*{section}{chapter}
7987             }
7988         }
7989     }
7990 }

```

`\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
7991 \def\part@prefix{}
7992 \@ifpackageloaded{document-structure}{
7993     \str_case:VnF \__notesslidestopsect {

```

```

7994 {part}{
7995   \int_set:Nn \l_document_structure_section_level_int {0}
7996   \def\thesection{\arabic{chapter}.\arabic{section}}
7997   \def\part@prefix{\arabic{chapter}.}
7998 }
7999 {chapter}{
8000   \int_set:Nn \l_document_structure_section_level_int {1}
8001   \def\thesection{\arabic{chapter}.\arabic{section}}
8002   \def\part@prefix{\arabic{chapter}.}
8003 }
8004 }{
8005   \int_set:Nn \l_document_structure_section_level_int {2}
8006   \def\part@prefix{}
8007 }
8008 }
8009
8010 \bool_if:NF \c__notesslides_notes_bool { % only in slides

```

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

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

`sfragment (env.)`

```

8011 \renewenvironment{sfragment}[2][]{
8012   \__document_structure_sfragment_args:n { #1 }
8013   \int_incr:N \l_document_structure_section_level_int
8014   \bool_if:NT \c__notesslides_sectocframes_bool {
8015     \stepcounter{slide}
8016     \begin{frame}[noframenumbering]
8017     \vfill\Large\centering
8018     \red{
8019       \ifcase\l_document_structure_section_level_int\or
8020         \stepcounter{part}
8021         \def\__notesslideslabel{\omdoc@part@kw}~\Roman{part}}
8022         \addcontentsline{toc}{part}{\protect\numberline{\thepart}#2}
8023         \pdfbookmark[0]{\thepart\ #2}{part.\thepart}
8024         \def\currentsectionlevel{\omdoc@part@kw}
8025       \or
8026         \stepcounter{chapter}
8027         \def\__notesslideslabel{\omdoc@chapter@kw}~\arabic{chapter}}
8028         \addcontentsline{toc}{chapter}{\protect\numberline{\thechapter}#2}
8029         \pdfbookmark[1]{\thechapter\ #2}{chapter.\cs_if_exist:cT{\thepart}\thepart.\thechapter}
8030         \def\currentsectionlevel{\omdoc@chapter@kw}
8031       \or
8032         \stepcounter{section}
8033         \def\__notesslideslabel{\part@prefix\arabic{section}}
8034         \addcontentsline{toc}{section}{\protect\numberline{\thesection}#2}
8035         \pdfbookmark[2]{\cs_if_exist:cT{\thechapter}{\thechapter}.\thesection\ #2}
8036         {section.\cs_if_exist:cT{\thepart}{\thepart}.\cs_if_exist:cT{\thechapter}{\thechapter}.\thepart}
8037         \def\currentsectionlevel{\omdoc@section@kw}
8038       \or
8039         \stepcounter{subsection}
8040         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
8041         \addcontentsline{toc}{subsection}{\protect\numberline{\thesubsection}#2}

```

```

8042         \pdfbookmark[3]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection
8043         {subsection.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection}
8044         \def\currentsectionlevel{\omdoc@subsection@kw}
8045     \or
8046         \stepcounter{subsubsection}
8047         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
8048         \addcontentsline{toc}{subsubsection}{\protect\numberline{\thesubsubsection}#2}
8049         \pdfbookmark[4]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection.\thesubsubsection}
8050         {subsubsection.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsubsection}
8051         \def\currentsectionlevel{\omdoc@subsubsection@kw}
8052     \or
8053         \stepcounter{paragraph}
8054         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{paragraph}}
8055         \addcontentsline{toc}{paragraph}{\protect\numberline{\theparagraph}#2}
8056         \pdfbookmark[5]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection.\theparagraph}
8057         {paragraph.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\theparagraph}
8058         \def\currentsectionlevel{\omdoc@paragraph@kw}
8059     \else
8060         \def\__notesslideslabel{}
8061         \def\currentsectionlevel{\omdoc@paragraph@kw}
8062     \fi% end ifcase
8063     \__notesslideslabel\quad #2%
8064 }%
8065 \vfill%
8066 \end{frame}%
8067 }
8068 \str_if_empty:NF \l__document_structure_sfragment_id_str {
8069     \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
8070 }
8071 }{}
8072 }

```

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

```

8073 \def\inserttheorembodyfont{\normalfont}
8074 %\bool_if:NF \c__notesslides_notes_bool {
8075 % \defbeamertemplate{theorem begin}{miko}
8076 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
8077 % \ifx\inserttheoremaddition@empty\else\ (\inserttheoremaddition)\fi%
8078 % \inserttheorempunctuation\inserttheorembodyfont\xspace}
8079 % \defbeamertemplate{theorem end}{miko}{\fi}

```

and we set it as the default one.

```

8080 % \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.

```

8081 % \expandafter\def\csname Parent2\endcsname{}
8082 %}
8083
8084 \AddToHook{begindocument}{% this does not work for some reason
8085     \setbeamertemplate{theorems}[ams style]
8086 }
8087 \bool_if:NT \c__notesslides_notes_bool {
8088     \renewenvironment{columns}[1][\fi]{}{}

```

```

8089     \par\noindent%
8090     \begin{minipage}%
8091     \slidewidth\centering\leavevmode%
8092   }{%
8093     \end{minipage}\par\noindent%
8094   }%
8095   \newsavebox\columnbox%
8096   \renewenvironment<>{column}[2][]{%
8097     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
8098   }{%
8099     \end{minipage}\end{lrbox}\usebox\columnbox%
8100   }%
8101 }

8102 \bool_if:NTF \c__notesslides_noproblems_bool {
8103   \newenvironment{problems}{}{}
8104 }{
8105   \excludecomment{problems}
8106 }

```

## 40.6 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.

```

8107 \gdef\printexcursions{}
8108 \newcommand\excursionref[2]{% label, text
8109   \bool_if:NT \c__notesslides_notes_bool {
8110     \begin{sparagraph}[title=Excursion]
8111       #2 \sref[fallback=the appendix]{#1}.
8112     \end{sparagraph}
8113   }
8114 }
8115 \newcommand\activate@excursion[2][{}{
8116   \gappto\printexcursions{\inputref{#1}{#2}}
8117 }
8118 \newcommand\excursion[4][{}{ repos, label, path, text
8119   \bool_if:NT \c__notesslides_notes_bool {
8120     \activate@excursion{#1}{#3}\excursionref{#2}{#4}
8121   }
8122 }

```

(End definition for `\excursion`. This function is documented on page 66.)

**\excursiongroup**

```

8123 \keys_define:nn{notesslides / excursiongroup }{
8124   id          .str_set_x:N = \l__notesslides_excursion_id_str,
8125   intro       .tl_set:N   = \l__notesslides_excursion_intro_tl,
8126   mhrepos     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
8127 }
8128 \cs_new_protected:Nn \__notesslides_excursion_args:n {
8129   \tl_clear:N \l__notesslides_excursion_intro_tl
8130   \str_clear:N \l__notesslides_excursion_id_str

```

```

8131 \str_clear:N \l__notesslides_excursion_mhrepos_str
8132 \keys_set:nn {notesslides / excursiongroup }{ #1 }
8133 }
8134 \newcommand\excursiongroup[1][ ]{
8135   \__notesslides_excursion_args:n{ #1 }
8136   \ifdefempty\printexcursions{}% only if there are excursions
8137   {\begin{note}
8138     \begin{sfragment}[#1]{Excursions}%
8139     \ifdefempty\l__notesslides_excursion_intro_tl}{
8140       \inputref[\l__notesslides_excursion_mhrepos_str]{
8141         \l__notesslides_excursion_intro_tl
8142       }
8143     }
8144     \printexcursions%
8145     \end{sfragment}
8146   \end{note}}
8147 }
8148 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
8149 \end{package}

```

(End definition for \excursiongroup. This function is documented on page 66.)

## Chapter 41

# The Implementation

### 41.1 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.

```
8150 <*package>
8151 <@@=problems>
8152 \ProvidesExplPackage{problem}{2022/09/14}{3.2.0}{Semantic Markup for Problems}
8153 \RequirePackage{13keys2e}
8154 \RequirePackage{amssymb}% for \Box
8155
8156 \keys_define:nn { problem / pkg }{
8157   notes      .default:n    = { true },
8158   notes      .bool_set:N   = \c__problems_notes_bool,
8159   gnotes     .default:n    = { true },
8160   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
8161   hints      .default:n    = { true },
8162   hints      .bool_set:N   = \c__problems_hints_bool,
8163   solutions  .default:n    = { true },
8164   solutions  .bool_set:N   = \c__problems_solutions_bool,
8165   pts        .default:n    = { true },
8166   pts        .bool_set:N   = \c__problems_pts_bool,
8167   min        .default:n    = { true },
8168   min        .bool_set:N   = \c__problems_min_bool,
8169   boxed      .default:n    = { true },
8170   boxed      .bool_set:N   = \c__problems_boxed_bool,
8171   test       .default:n    = { true },
8172   test       .bool_set:N   = \c__problems_test_bool,
8173   unknown    .code:n       = {
8174     \PassOptionsToPackage{\CurrentOption}{stex}
8175   }
8176 }
8177 \newif\ifsolutions
8178
8179 \ProcessKeysOptions{ problem / pkg }
8180 \bool_if:NTF \c__problems_solutions_bool {
8181   \solutionstrue
```

```

8182 }{
8183   \solutionsfalse
8184 }
8185 \RequirePackage{stex}

```

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

```

8186 \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.

```

8187 \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.

```

8188 \def\prob@problem@kw{Problem}
8189 \def\prob@solution@kw{Solution}
8190 \def\prob@hint@kw{Hint}
8191 \def\prob@note@kw{Note}
8192 \def\prob@gnote@kw{Grading}
8193 \def\prob@pt@kw{pt}
8194 \def\prob@min@kw{min}
8195 \def\prob@correct@kw{Correct}
8196 \def\prob@wrong@kw{Wrong}

```

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

For the other languages, we set up triggers

```

8197 \AddToHook{begindocument}{
8198   \ltx@ifpackageloaded{babel}{
8199     \makeatletter
8200     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
8201     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
8202       \input{problem-ngerman.ldf}
8203     }
8204     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
8205       \input{problem-finnish.ldf}
8206     }
8207     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
8208       \input{problem-french.ldf}
8209     }
8210     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
8211       \input{problem-russian.ldf}
8212     }
8213     \makeatother
8214   }{ }
8215 }

```

## 41.2 Problems and Solutions

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

```

8216 \keys_define:nn{ problem / problem }{
8217   id .str_set_x:N = \l__problems_prob_id_str,

```

```

8218 pts      .tl_set:N      = \l__problems_prob_pts_tl,
8219 min      .tl_set:N      = \l__problems_prob_min_tl,
8220 title    .tl_set:N      = \l__problems_prob_title_tl,
8221 type     .tl_set:N      = \l__problems_prob_type_tl,
8222 imports  .tl_set:N      = \l__problems_prob_imports_tl,
8223 name     .str_set_x:N     = \l__problems_prob_name_str,
8224 refnum   .int_set:N      = \l__problems_prob_refnum_int
8225 }
8226 \cs_new_protected:Nn \__problems_prob_args:n {
8227   \str_clear:N \l__problems_prob_id_str
8228   \str_clear:N \l__problems_prob_name_str
8229   \tl_clear:N \l__problems_prob_pts_tl
8230   \tl_clear:N \l__problems_prob_min_tl
8231   \tl_clear:N \l__problems_prob_title_tl
8232   \tl_clear:N \l__problems_prob_type_tl
8233   \tl_clear:N \l__problems_prob_imports_tl
8234   \int_zero_new:N \l__problems_prob_refnum_int
8235   \keys_set:nn { problem / problem }{ #1 }
8236   \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
8237     \let\l__problems_prob_refnum_int\undefined
8238   }
8239 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

8240 \newcounter{problem}[section]
8241 \newcommand\numberproblemsin[1]{\@addtoreset{problem}{#1}}
8242 \def\theplainsproblem{\arabic{problem}}
8243 \def\thesproblem{\thesection.\theplainsproblem}

```

(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.

```

8244 \newcommand\prob@label[1]{\thesection.#1}

```

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

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

```

8245 \newcommand\prob@number{
8246   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
8247     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
8248   }{
8249     \int_if_exist:NTF \l__problems_prob_refnum_int {
8250       \prob@label{\int_use:N \l__problems_prob_refnum_int }
8251     }{
8252       \prob@label\theplainsproblem
8253     }
8254   }
8255 }
8256 \def\sproblemautorefname{\prob@problem@kw}

```

(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.

```

8257 \newcommand\prob@title[3]{%
8258   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
8259     #2 \l__problems_inclprob_title_tl #3
8260   }{
8261     \tl_if_empty:NTF \l__problems_prob_title_tl {
8262       #1
8263     }{
8264       #2 \l__problems_prob_title_tl #3
8265     }
8266   }
8267 }

```

(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.

```

8268 \def\prob@heading{
8269   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
8270   %\sref@label@id{\prob@problem@kw~\prob@number}{~}
8271 }

```

(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 (env.)`

```

8272 \newenvironment{sproblem}[1][]{
8273   \__problems_prob_args:n{#1}%\sref@target%
8274   \@in@omtexttrue% we are in a statement (for inline definitions)
8275   \refstepcounter{sproblem}\record@problem
8276   \def\current@section@level{\prob@problem@kw}
8277
8278   \str_if_empty:NT \l__problems_prob_name_str {
8279     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
8280     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
8281     \seq_get_left:NN \l_tmpa_seq \l__problems_prob_name_str
8282   }
8283
8284   \stex_if_do_html:T{
8285     \tl_if_empty:NF \l__problems_prob_title_tl {
8286       \exp_args:No \stex_document_title:n \l__problems_prob_title_tl
8287     }
8288   }
8289
8290   \exp_args:Nno\stex_module_setup:nn{type=problem}\l__problems_prob_name_str
8291
8292   \stex_reactivate_macro:N \STEXexport
8293   \stex_reactivate_macro:N \importmodule

```

```

8294 \stex_reactivate_macro:N \symdecl
8295 \stex_reactivate_macro:N \notation
8296 \stex_reactivate_macro:N \symdef
8297
8298 \stex_if_do_html:T{
8299   \begin{stex_annotate_env} {problem} {
8300     \l_stex_module_ns_str ? \l_stex_module_name_str
8301   }
8302
8303   \stex_annotate_invisible:nnn{header}{ } {
8304     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{ }
8305     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{ }
8306     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
8307       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{ }
8308     }
8309   }
8310 }
8311
8312 \stex_csl_to_imports:No \importmodule \l__problems_prob_imports_tl
8313
8314
8315 \tl_if_exist:NTF \l__problems_inclprob_type_tl {
8316   \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
8317 }{
8318   \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
8319 }
8320 \str_if_exist:NTF \l__problems_inclprob_id_str {
8321   \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
8322 }{
8323   \str_set_eq:NN \sproblemid \l__problems_prob_id_str
8324 }
8325
8326
8327 \stex_if_smsmode:F {
8328   \clist_set:No \l_tmpa_clist \sproblemtype
8329   \tl_clear:N \l_tmpa_tl
8330   \clist_map_inline:Nn \l_tmpa_clist {
8331     \tl_if_exist:cT {__problems_sproblem_##1_start:}{
8332       \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_start:}}
8333     }
8334   }
8335   \tl_if_empty:NTF \l_tmpa_tl {
8336     \__problems_sproblem_start:
8337   }{
8338     \l_tmpa_tl
8339   }
8340 }
8341 \stex_ref_new_doc_target:n \sproblemid
8342 \stex_if_smsmode:TF \stex_smsmode_do: \ignorespacesandpars
8343 }{
8344   \__stex_modules_end_module:
8345   \stex_if_smsmode:F{
8346     \clist_set:No \l_tmpa_clist \sproblemtype
8347     \tl_clear:N \l_tmpa_tl

```

```

8348 \clist_map_inline:Nn \l_tmpa_clist {
8349 \tl_if_exist:cT {__problems_sproblem_##1_end:}{
8350 \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_end:}}
8351 }
8352 }
8353 \tl_if_empty:NTF \l_tmpa_tl {
8354 \__problems_sproblem_end:
8355 }{
8356 \l_tmpa_tl
8357 }
8358 }
8359 \stex_if_do_html:T{
8360 \end{stex_annotate_env}
8361 }
8362
8363 \smallskip
8364 }
8365
8366 \seq_put_right:Nx\g_stex_smsmode_allowedenvs_seq{\tl_to_str:n{sproblem}}
8367
8368
8369
8370 \cs_new_protected:Nn \__problems_sproblem_start: {
8371 \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
8372 }
8373 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
8374
8375 \newcommand\stexpatchproblem[3][] {
8376 \str_set:Nx \l_tmpa_str{ #1 }
8377 \str_if_empty:NTF \l_tmpa_str {
8378 \tl_set:Nn \__problems_sproblem_start: { #2 }
8379 \tl_set:Nn \__problems_sproblem_end: { #3 }
8380 }{
8381 \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
8382 \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
8383 }
8384 }
8385
8386
8387 \bool_if:NT \c__problems_boxed_bool {
8388 \surroundwithmdframed{problem}
8389 }

```

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

```

8390 \def\record@problem{
8391 \protected@write\@auxout{}
8392 {
8393 \string\@problem{\prob@number}
8394 {
8395 \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
8396 \l__problems_inclprob_pts_tl
8397 }{
8398 \l__problems_prob_pts_tl
8399 }

```

```

8400 }%
8401 {
8402   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
8403     \l__problems_inclprob_min_tl
8404   }{
8405     \l__problems_prob_min_tl
8406   }
8407 }
8408 }
8409 }

```

(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).

```

8410 \def\@problem#1#2#3{}

```

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

**`solution (env.)`** 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.

```

8411 \keys_define:nn { problem / solution }{
8412   id          .str_set_x:N = \l__problems_solution_id_str ,
8413   for         .str_set_x:N = \l__problems_solution_for_str ,
8414   type       .str_set_x:N = \l__problems_solution_type_str ,
8415   title      .tl_set:N     = \l__problems_solution_title_tl
8416 }
8417 \cs_new_protected:Nn \__problems_solution_args:n {
8418   \str_clear:N \l__problems_solution_id_str
8419   \str_clear:N \l__problems_solution_type_str
8420   \str_clear:N \l__problems_solution_for_str
8421   \tl_clear:N \l__problems_solution_title_tl
8422   \keys_set:nn { problem / solution }{ #1 }
8423 }

```

**`\startolutions`** for the `\startolutions` 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.

```

8424 \box_new:N \l__problems_solution_box
8425 \newenvironment{solution}[1][{}]{
8426   \__problems_solution_args:n{#1}
8427   \stex_html_backend:TF{
8428     \stex_if_do_html:T{
8429       \begin{stex_annotate_env}{solution}{}
8430       \str_if_empty:NF \l__problems_solution_type_str {
8431         \par\noindent
8432         \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{}
8433       }
8434       \noindent\textbf{Solution}\tl_if_empty:NF\l__problems_solution_title_tl{~(\l__problem
8435     }
8436   }{
8437     \setbox\l__problems_solution_box\vbox\bgroup
8438     \par\smallskip\hrule\smallskip
8439     \noindent\textbf{Solution}\tl_if_empty:NF\l__problems_solution_title_tl{~(\l__problems
8440   }

```

```

8441 }{
8442   \stex_html_backend:TF{
8443     \stex_if_do_html:T{
8444       \end{stex_annotate_env}
8445     }
8446   }{
8447     \smallskip\hrule
8448     \egroup
8449     \bool_if:NT \c__problems_solutions_bool {
8450       \strut\par\noindent
8451       \box\l__problems_solution_box
8452     }
8453   }
8454 }
8455
8456 \newcommand\startsolutions{
8457   \bool_set_true:N \c__problems_solutions_bool
8458   \solutionstrue
8459   % \specialcomment{solution}{\@startsolution}{
8460   %   \bool_if:NF \c__problems_boxed_bool {
8461   %     \hrule\medskip
8462   %   }
8463   %   \end{small}}%
8464   % }
8465   % \bool_if:NT \c__problems_boxed_bool {
8466   %   \surroundwithmdframed{solution}
8467   % }
8468 }

```

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

### \stopsolutions

```

8469 \newcommand\stopsolutions{\bool_set_false:N \c__problems_solutions_bool \solutionsfalse}%\ex

```

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

### exnote (env.)

```

8470 \bool_if:NTF \c__problems_notes_bool {
8471   \newenvironment{exnote}[1][{}]{
8472     \par\smallskip\hrule\smallskip
8473     \noindent\textbf{\prob@note@kw :~ }\small
8474   }{
8475     \smallskip\hrule
8476   }
8477 }{
8478   \excludacomment{exnote}
8479 }

```

### hint (env.)

```

8480 \bool_if:NTF \c__problems_notes_bool {
8481   \newenvironment{hint}[1][{}]{
8482     \par\smallskip\hrule\smallskip
8483     \noindent\textbf{\prob@hint@kw :~ }\small
8484   }{

```

```

8485     \smallskip\hrule
8486   }
8487   \newenvironment{exhint}[1][ ]{
8488     \par\smallskip\hrule\smallskip
8489     \noindent\textbf{\prob@hint@kw :~ }\small
8490   }{
8491     \smallskip\hrule
8492   }
8493   }{
8494     \excludecomment{hint}
8495     \excludecomment{exhint}
8496   }

```

**gnote** (*env.*)

```

8497   \bool_if:NTF \c__problems_notes_bool {
8498     \newenvironment{gnote}[1][ ]{
8499       \par\smallskip\hrule\smallskip
8500       \noindent\textbf{\prob@gnote@kw :~ }\small
8501     }{
8502       \smallskip\hrule
8503     }
8504   }{
8505     \excludecomment{gnote}
8506   }

```

## 41.3 Markup for Added Value Services

## 41.4 Multiple Choice Blocks

EdN:12

**mcb** (*env.*)<sup>12</sup>

```

8507   \newenvironment{mcb}{
8508     \begin{enumerate}
8509   }{
8510     \end{enumerate}
8511   }

```

we define the keys for the mcb macro

```

8512   \cs_new_protected:Nn \__problems_do_yes_param:Nn {
8513     \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
8514       \bool_set_true:N #1
8515     }{
8516       \bool_set_false:N #1
8517     }
8518   }
8519   \keys_define:nn { problem / mcb }{
8520     id      .str_set_x:N = \l__problems_mcc_id_str ,
8521     feedback .tl_set:N   = \l__problems_mcc_feedback_tl ,
8522     T       .default:n   = { false } ,
8523     T       .bool_set:N  = \l__problems_mcc_t_bool ,
8524     F       .default:n   = { false } ,

```

---

<sup>12</sup>EdNOTE: MK: maybe import something better here from a dedicated MC package

```

8525 F      .bool_set:N    = \l__problems_mcc_f_bool ,
8526 Ttext   .tl_set:N     = \l__problems_mcc_Ttext_tl ,
8527 Ftext   .tl_set:N     = \l__problems_mcc_Ftext_tl
8528 }
8529 \cs_new_protected:Nn \l__problems_mcc_args:n {
8530   \str_clear:N \l__problems_mcc_id_str
8531   \tl_clear:N \l__problems_mcc_feedback_tl
8532   \bool_set_false:N \l__problems_mcc_t_bool
8533   \bool_set_false:N \l__problems_mcc_f_bool
8534   \tl_clear:N \l__problems_mcc_Ttext_tl
8535   \tl_clear:N \l__problems_mcc_Ftext_tl
8536   \str_clear:N \l__problems_mcc_id_str
8537   \keys_set:nn { problem / mcc }{ #1 }
8538 }

```

**\mcc**

```

8539 \def\mccTrueText{\textbf{\prob@correct@kw!~}}
8540 \def\mccFalseText{\textbf{\prob@wrong@kw!~}}
8541 \newcommand\mcc[2][] {
8542   \l__problems_mcc_args:n{ #1 }
8543   \item[{$\Box$}] #2
8544   \bool_if:NT \c__problems_solutions_bool {
8545     \
8546     \bool_if:NT \l__problems_mcc_t_bool {
8547       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccTrueText\l__problems_mcc_Ttext_tl
8548     }
8549     \bool_if:NT \l__problems_mcc_f_bool {
8550       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccFalseText\l__problems_mcc_Ftext_tl
8551     }
8552     \tl_if_empty:NF \l__problems_mcc_feedback_tl {
8553       \emph{\l__problems_mcc_feedback_tl}
8554     }
8555   }
8556 } %solutions

```

(End definition for \mcc. This function is documented on page 69.)

## 41.5 Filling in Concrete Solutions

**\includeproblem** This is embarrassingly simple, but can grow over time.

```

8557 \newcommand\fillinsol[2][] {%
8558   \def\@test{#1}
8559   \quad%
8560   \ifsolutions\textcolor{red}{\@test!}\else%
8561   \fbox{\ifx\@test\empty\phantom{\huge{21}}\else\hspace{#1}\fi}%
8562   \fi}

```

(End definition for \includeproblem. This function is documented on page 71.)

## 41.6 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.

```

8563
8564 \keys_define:nn{ problem / inclproblem }{
8565   id      .str_set_x:N = \l__problems_inclprob_id_str,
8566   pts     .tl_set:N    = \l__problems_inclprob_pts_tl,
8567   min     .tl_set:N    = \l__problems_inclprob_min_tl,
8568   title   .tl_set:N    = \l__problems_inclprob_title_tl,
8569   refnum  .int_set:N    = \l__problems_inclprob_refnum_int,
8570   type    .tl_set:N    = \l__problems_inclprob_type_tl,
8571   mhrepos .str_set_x:N = \l__problems_inclprob_mhrepos_str
8572 }
8573 \cs_new_protected:Nn \l__problems_inclprob_args:n {
8574   \str_clear:N \l__problems_prob_id_str
8575   \tl_clear:N \l__problems_inclprob_pts_tl
8576   \tl_clear:N \l__problems_inclprob_min_tl
8577   \tl_clear:N \l__problems_inclprob_title_tl
8578   \tl_clear:N \l__problems_inclprob_type_tl
8579   \int_zero_new:N \l__problems_inclprob_refnum_int
8580   \str_clear:N \l__problems_inclprob_mhrepos_str
8581   \keys_set:nn { problem / inclproblem }{ #1 }
8582   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
8583     \let\l__problems_inclprob_pts_tl\undefined
8584   }
8585   \tl_if_empty:NT \l__problems_inclprob_min_tl {
8586     \let\l__problems_inclprob_min_tl\undefined
8587   }
8588   \tl_if_empty:NT \l__problems_inclprob_title_tl {
8589     \let\l__problems_inclprob_title_tl\undefined
8590   }
8591   \tl_if_empty:NT \l__problems_inclprob_type_tl {
8592     \let\l__problems_inclprob_type_tl\undefined
8593   }
8594   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
8595     \let\l__problems_inclprob_refnum_int\undefined
8596   }
8597 }
8598
8599 \cs_new_protected:Nn \l__problems_inclprob_clear: {
8600   \let\l__problems_inclprob_id_str\undefined
8601   \let\l__problems_inclprob_pts_tl\undefined
8602   \let\l__problems_inclprob_min_tl\undefined
8603   \let\l__problems_inclprob_title_tl\undefined
8604   \let\l__problems_inclprob_type_tl\undefined
8605   \let\l__problems_inclprob_refnum_int\undefined
8606   \let\l__problems_inclprob_mhrepos_str\undefined
8607 }
8608 \l__problems_inclprob_clear:
8609
8610 \newcommand\includeproblem[2][ ]{
8611   \l__problems_inclprob_args:n{ #1 }

```



```

8612 \exp_args:No \stex_in_repository:nn\l__problems_inclprob_mhrepos_str{
8613   \stex_html_backend:TF {
8614     \str_clear:N \l_tmpa_str
8615     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
8616       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
8617     }
8618     \stex_annotate_invisible:nnn{includeproblem}{
8619       \l_tmpa_str / #2
8620     }{}
8621   }{
8622     \begingroup
8623     \inputreftrue
8624     \tl_if_empty:nTF{ ##1 }{
8625       \input{#2}
8626     }{
8627       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
8628     }
8629     \endgroup
8630   }
8631 }
8632 \__problems_inclprob_clear:
8633 }

```

(End definition for `\includeproblem`. This function is documented on page 71.)

## 41.7 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.

```

8634 \AddToHook{enddocument}{
8635   \bool_if:NT \c__problems_pts_bool {
8636     \message{Total:~\arabic{pts}~points}
8637   }
8638   \bool_if:NT \c__problems_min_bool {
8639     \message{Total:~\arabic{min}~minutes}
8640   }
8641 }

```

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

```

8642 \def\pts#1{
8643   \bool_if:NT \c__problems_pts_bool {
8644     \marginpar{#1~\prob@pt@kw}
8645   }
8646 }
8647 \def\min#1{
8648   \bool_if:NT \c__problems_min_bool {
8649     \marginpar{#1~\prob@min@kw}
8650   }
8651 }

```

`\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.

```

8652 \newcounter{pts}
8653 \def\show@pts{
8654   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
8655     \bool_if:NT \c__problems_pts_bool {
8656       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
8657       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
8658     }
8659   }{
8660     \tl_if_exist:NT \l__problems_prob_pts_tl {
8661       \bool_if:NT \c__problems_pts_bool {
8662         \tl_if_empty:NT\l__problems_prob_pts_tl{
8663           \tl_set:Nn \l__problems_prob_pts_tl {0}
8664         }
8665         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
8666         \addtocounter{pts}{\l__problems_prob_pts_tl}
8667       }
8668     }
8669   }
8670 }

```

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

\show@min

```

8671 \newcounter{min}
8672 \def\show@min{
8673   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
8674     \bool_if:NT \c__problems_min_bool {
8675       \marginpar{\l__problems_inclprob_pts_tl\ min}
8676       \addtocounter{min}{\l__problems_inclprob_min_tl}
8677     }
8678   }{
8679     \tl_if_exist:NT \l__problems_prob_min_tl {
8680       \bool_if:NT \c__problems_min_bool {
8681         \tl_if_empty:NT\l__problems_prob_min_tl{
8682           \tl_set:Nn \l__problems_prob_min_tl {0}
8683         }
8684         \marginpar{\l__problems_prob_min_tl\ min}
8685         \addtocounter{min}{\l__problems_prob_min_tl}
8686       }
8687     }
8688   }
8689 }
8690 \</package>

```

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

## 41.8 Testing and Spacing

\testspace

```

8691 \newcommand\testspace[1]{\bool_if:NT \c__problems_boxed_bool {\vspace*{#1}}}

```

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

`\testnewpage`

```
8692 \newcommand\testnewpage{\bool_if:NT \c__problems_boxed_bool {\newpage}}
```

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

`\testemptypage`

```
8693 \newcommand\testemptypage[1] [] {%
```

```
8694 \bool_if:NT \c__problems_boxed_bool {\begin{center}\hwexam@testemptypage@kw\end{center}\vfil
```

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

`\test*space`

```
8695 \newcommand\testsmallspace{\testspace{1cm}}
```

```
8696 \newcommand\testmedspace{\testspace{2cm}}
```

```
8697 \newcommand\testbigspace{\testspace{3cm}}
```

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

## Chapter 42

# Implementation: The hwexam Package

### 42.1 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.

```
8698 \*package>
8699 \ProvidesExplPackage{hwexam}{2022/09/14}{3.2.0}{homework assignments and exams}
8700 \RequirePackage{13keys2e}
8701
8702 \newif\iftest\testfalse
8703 \DeclareOption{test}{\testtrue\PassOptionsToPackage{\CurrentOption}{problem}}
8704 \newif\ifmultiple\multiplefalse
8705 \DeclareOption{multiple}{\multipletrue}
8706 \DeclareOption{lang}{\PassOptionsToPackage{\CurrentOption}{problem}}
8707 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
8708 \ProcessOptions
```

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

```
8709 \RequirePackage{keyval}[1997/11/10]
8710 \RequirePackage{problem}
```

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

```
8711 \newcommand\hwexam@assignment@kw{Assignment}
8712 \newcommand\hwexam@given@kw{Given}
8713 \newcommand\hwexam@due@kw{Due}
8714 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~blank~for~extra~space}
8715 \newcommand\hwexam@minutes@kw{minutes}
8716 \newcommand\correction@probs@kw{prob.}
8717 \newcommand\correction@pts@kw{total}
8718 \newcommand\correction@reached@kw{reached}
8719 \newcommand\correction@sum@kw{Sum}
8720 \newcommand\correction@grade@kw{grade}
8721 \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

```

8722 \AddToHook{begindocument}{
8723 \ltx@ifpackageloaded{babel}{
8724 \makeatletter
8725 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
8726 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
8727 \input{hwexam-ngerman.ldf}
8728 }
8729 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
8730 \input{hwexam-finnish.ldf}
8731 }
8732 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
8733 \input{hwexam-french.ldf}
8734 }
8735 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
8736 \input{hwexam-russian.ldf}
8737 }
8738 \makeatother
8739 }{}
8740 }
8741

```

## 42.2 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.

```

8742 \newcounter{assignment}
8743 %\numberproblemsin{assignment}

We will prepare the keyval support for the assignment environment.

8744 \keys_define:nn { hwexam / assignment } {
8745 id .str_set:N = \l_@@_assign_id_str,
8746 number .int_set:N = \l_@@_assign_number_int,
8747 title .tl_set:N = \l_@@_assign_title_tl,
8748 type .tl_set:N = \l_@@_assign_type_tl,
8749 given .tl_set:N = \l_@@_assign_given_tl,
8750 due .tl_set:N = \l_@@_assign_due_tl,
8751 loadmodules .code:n = {
8752 \bool_set_true:N \l_@@_assign_loadmodules_bool
8753 }
8754 }
8755 \cs_new_protected:Nn \_@@_assignment_args:n {
8756 \str_clear:N \l_@@_assign_id_str
8757 \int_set:Nn \l_@@_assign_number_int {-1}
8758 \tl_clear:N \l_@@_assign_title_tl
8759 \tl_clear:N \l_@@_assign_type_tl
8760 \tl_clear:N \l_@@_assign_given_tl
8761 \tl_clear:N \l_@@_assign_due_tl
8762 \bool_set_false:N \l_@@_assign_loadmodules_bool
8763 \keys_set:nn { hwexam / assignment }{ #1 }
8764 }

```

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.

```

8765 \newcommand\given@due[2]{
8766 \bool_lazy_all:nF {
8767 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
8768 { \tl_if_empty_p:V \l_@@_assign_given_tl }
8769 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
8770 { \tl_if_empty_p:V \l_@@_assign_due_tl }
8771 }{ #1 }
8772
8773 \tl_if_empty:NTF \l_@@_inclasssign_given_tl {
8774 \tl_if_empty:NF \l_@@_assign_given_tl {
8775 \hwexam@given@kw\xspace\l_@@_assign_given_tl
8776 }
8777 }{
8778 \hwexam@given@kw\xspace\l_@@_inclasssign_given_tl
8779 }
8780
8781 \bool_lazy_or:nnF {
8782 \bool_lazy_and_p:nn {
8783 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
8784 }{
8785 \tl_if_empty_p:V \l_@@_assign_due_tl
8786 }
8787 }{
8788 \bool_lazy_and_p:nn {
8789 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
8790 }{
8791 \tl_if_empty_p:V \l_@@_assign_due_tl
8792 }
8793 }{ ,~ }
8794
8795 \tl_if_empty:NTF \l_@@_inclasssign_due_tl {
8796 \tl_if_empty:NF \l_@@_assign_due_tl {
8797 \hwexam@due@kw\xspace \l_@@_assign_due_tl
8798 }
8799 }{
8800 \hwexam@due@kw\xspace \l_@@_inclasssign_due_tl
8801 }
8802
8803 \bool_lazy_all:nF {
8804 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
8805 { \tl_if_empty_p:V \l_@@_assign_given_tl }
8806 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
8807 { \tl_if_empty_p:V \l_@@_assign_due_tl }
8808 }{ #2 }
8809 }

```

`\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.

```

8810 \newcommand\assignment@title[3]{
8811 \tl_if_empty:NTF \l_@@_inclasssign_title_tl {
8812 \tl_if_empty:NTF \l_@@_assign_title_tl {
8813 #1
8814 }{
8815 #2\l_@@_assign_title_tl#3
8816 }
8817 }{
8818 #2\l_@@_inclasssign_title_tl#3
8819 }
8820 }

```

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

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

```

8821 \newcommand\assignment@number{
8822 \int_compare:nNnTF \l_@@_inclasssign_number_int = {-1} {
8823 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8824 \arabic{assignment}
8825 } {
8826 \int_use:N \l_@@_assign_number_int
8827 }
8828 }{
8829 \int_use:N \l_@@_inclasssign_number_int
8830 }
8831 }

```

(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 (env.)** For the **assignment** environment we delegate the work to the **@assignment** environment that depends on whether **multiple** option is given.

```

8832 \newenvironment{assignment}[1][]{
8833 \_@@_assignment_args:n { #1 }
8834 %\sref@target
8835 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8836 \global\stepcounter{assignment}
8837 }{
8838 \global\setcounter{assignment}{\int_use:N\l_@@_assign_number_int}
8839 }
8840 \setcounter{sproblem}{0}
8841 \renewcommand\prob@label[1]{\assignment@number.##1}
8842 \def\current@section@level{\document@hwexamtype}
8843 %\sref@label{id}{\document@hwexamtype \thesection}
8844 \begin{@assignment}
8845 }{
8846 \end{@assignment}
8847 }

```

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

```

8848 \def\ass@title{
8849 {\protect\document@hwexamtype}\arabic{assignment}
8850 \assignment@title{}\{;\}\; -- \given@due{}\}
8851 }
8852 \ifmultiple
8853 \newenvironment{@assignment}{
8854 \bool_if:NTF \l_@@_assign_loadmodules_bool {
8855 \begin{sfragment}[loadmodules]{\ass@title}
8856 }{
8857 \begin{sfragment}{\ass@title}
8858 }
8859 }{
8860 \end{sfragment}
8861 }

```

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

```

8862 \else
8863 \newenvironment{@assignment}{
8864 \begin{center}\bf
8865 \Large@title\strut\
8866 \document@hwexamtype}\arabic{assignment}\assignment@title{}\{;\}\;{\}\}
8867 \large\given@due{--;\}\{;\}--}
8868 \end{center}
8869 }{}
8870 \fi% multiple

```

## 42.3 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.

```

8871 \keys_define:nn { hwexam / inclassignment } {
8872 %id .str_set_x:N = \l_@@_assign_id_str,
8873 number .int_set:N = \l_@@_inclassign_number_int,
8874 title .tl_set:N = \l_@@_inclassign_title_tl,
8875 type .tl_set:N = \l_@@_inclassign_type_tl,
8876 given .tl_set:N = \l_@@_inclassign_given_tl,
8877 due .tl_set:N = \l_@@_inclassign_due_tl,
8878 mhrepos .str_set_x:N = \l_@@_inclassign_mhrepos_str
8879 }
8880 \cs_new_protected:Nn \l_@@_inclassignment_args:n {
8881 \int_set:Nn \l_@@_inclassign_number_int {-1}
8882 \tl_clear:N \l_@@_inclassign_title_tl
8883 \tl_clear:N \l_@@_inclassign_type_tl
8884 \tl_clear:N \l_@@_inclassign_given_tl
8885 \tl_clear:N \l_@@_inclassign_due_tl
8886 \str_clear:N \l_@@_inclassign_mhrepos_str
8887 \keys_set:nn { hwexam / inclassignment }{ #1 }
8888 }
8889 \l_@@_inclassignment_args:n {}
8890
8891 \newcommand\inputassignment[2][{}]{

```



```

8892 \_@@_inclassassignment_args:n { #1 }
8893 \str_if_empty:NTF \l_@@_inclasssign_mhrepos_str {
8894   \input{#2}
8895 }{
8896   \stex_in_repository:nn{\l_@@_inclasssign_mhrepos_str}{
8897     \input{\mhpath{\l_@@_inclasssign_mhrepos_str}{#2}}
8898   }
8899 }
8900 \_@@_inclassassignment_args:n {}
8901 }
8902 \newcommand\includeassignment[2][]{
8903   \newpage
8904   \inputassignment[#1]{#2}
8905 }

```

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

## 42.4 Typesetting Exams

\quizheading

```

8906 \ExplSyntaxOff
8907 \newcommand\quizheading[1]{%
8908   \def\@tas{#1}%
8909   \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
8910   \ifx\@tas\@empty\else%
8911     \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
8912   \fi%
8913 }
8914 \ExplSyntaxOn

```

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

\testheading

```

8915
8916 \def\hwexamheader{\input{hwexam-default.header}}
8917
8918 \def\hwexamminutes{
8919   \tl_if_empty:NTF \testheading@duration {
8920     {\testheading@min}~\hwexam@minutes@kw
8921   }{
8922     \testheading@duration
8923   }
8924 }
8925
8926 \keys_define:nn { hwexam / testheading } {
8927   min .tl_set:N = \testheading@min,
8928   duration .tl_set:N = \testheading@duration,
8929   reqpts .tl_set:N = \testheading@reqpts,
8930   tools .tl_set:N = \testheading@tools
8931 }
8932 \cs_new_protected:Nn \_@@_testheading_args:n {
8933   \tl_clear:N \testheading@min
8934   \tl_clear:N \testheading@duration

```

```

8935 \tl_clear:N \testheading@reqpts
8936 \tl_clear:N \testheading@tools
8937 \keys_set:nn { hwexam / testheading }{ #1 }
8938 }
8939 \newenvironment{testheading}[1][ ]{
8940 \_@@_testheading_args:n{ #1 }
8941 \newcount\check@time\check@time=\testheading@min
8942 \advance\check@time by -\theassignment@totalmin
8943 \newif\if@bonuspoints
8944 \tl_if_empty:NTF \testheading@reqpts {
8945 \@bonuspointsfalse
8946 }{
8947 \newcount\bonus@pts
8948 \bonus@pts=\theassignment@totalpts
8949 \advance\bonus@pts by -\testheading@reqpts
8950 \edef\bonus@pts{\the\bonus@pts}
8951 \@bonuspointstrue
8952 }
8953 \edef\check@time{\the\check@time}
8954
8955 \makeatletter\hwexamheader\makeatother
8956 }{
8957 \newpage
8958 }

```

(End definition for \testheading. 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.

```

8959 <@@=problems>
8960 \renewcommand\@problem[3]{
8961 \stepcounter{assignment@probs}
8962 \def\__problemspts{#2}
8963 \ifx\__problemspts\empty\else
8964 \addtocounter{assignment@totalpts}{#2}
8965 \fi
8966 \def\__problemsmin{#3}\ifx\__problemsmin\empty\else\addtocounter{assignment@totalmin}{#3}\fi
8967 \xdef\correction@probs{\correction@probs & #1}%
8968 \xdef\correction@pts{\correction@pts & #2}
8969 \xdef\correction@reached{\correction@reached &}
8970 }
8971 <@@=hwexam>

```

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

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

```

8972 \newcounter{assignment@probs}
8973 \newcounter{assignment@totalpts}
8974 \newcounter{assignment@totalmin}
8975 \def\correction@probs{\correction@probs@kw}
8976 \def\correction@pts{\correction@pts@kw}
8977 \def\correction@reached{\correction@reached@kw}
8978 \stepcounter{assignment@probs}
8979 \newcommand\correction@table{

```

```

8980 \resizebox{\textwidth}{!}{%
8981 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
8982 &\multicolumn{\theassignment@probs}{c|}{%|
8983 {\footnotesize\correction@forgrading@kw} &\\ \hline
8984 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
8985 \correction@pts & \theassignment@totalpts & \\ \hline
8986 \correction@reached & & \[.7cm]\hline
8987 \end{tabular}}
8988 \end{package}

```

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

## 42.5 Leftovers

at some point, we may want to reactivate the logos font, then we use

```

here we define the logos that characterize the assignment
\font\bierfont=../assignments/bierglas
\font\denkerfont=../assignments/denker
\font\uhrfont=../assignments/uhr
\font\warnschildfont=../assignments/achtung

\newcommand\bierglas{{\bierfont\char65}}
\newcommand\denker{{\denkerfont\char65}}
\newcommand\uhr{{\uhrfont\char65}}
\newcommand\warnschild{{\warnschildfont\char 65}}
\newcommand\hardA{\warnschild}
\newcommand\longA{\uhr}
\newcommand\thinkA{\denker}
\newcommand\discussA{\bierglas}

```

# Chapter 43

## References

EdN:13

13

- [Bus+04] Stephen Buswell et al. *The Open Math Standard, Version 2.0*. Tech. rep. The OpenMath Society, 2004. URL: <http://www.openmath.org/standard/om20>.
- [CR99] David Carlisle and Sebastian Rathz. *The graphicx package*. Part of the T<sub>E</sub>X distribution. The Comprehensive T<sub>E</sub>X Archive Network. 1999. URL: <https://www.tug.org/texlive/devsrc/Master/texmf-dist/doc/latex/graphics/graphics.pdf>.
- [DCM03] The DCMI Usage Board. *DCMI Metadata Terms*. DCMI Recommendation. Dublin Core Metadata Initiative, 2003. URL: <http://dublincore.org/documents/dcmi-terms/>.
- [Koh06] Michael Kohlhase. *OMDoc – An open markup format for mathematical documents [Version 1.2]*. LNAI 4180. Springer Verlag, Aug. 2006. URL: <http://omdoc.org/pubs/omdoc1.2.pdf>.
- [LMH] *LMH Scripts*. URL: <https://github.com/sLaTeX/lmhtools>.
- [MMT] *MMT – Language and System for the Uniform Representation of Knowledge*. Project web site. URL: <https://uniformal.github.io/> (visited on 01/15/2019).
- [MRK18] Dennis Müller, Florian Rabe, and Michael Kohlhase. “Theories as Types”. In: *9th International Joint Conference on Automated Reasoning*. Ed. by Didier Galmiche, Stephan Schulz, and Roberto Sebastiani. Springer Verlag, 2018. URL: <https://kwarc.info/kohlhase/papers/ijcar18-records.pdf>.
- [Rab15] Florian Rabe. “The Future of Logic: Foundation-Independence”. In: *Logica Universalis* 10.1 (2015). 10.1007/s11787-015-0132-x; Winner of the Contest “The Future of Logic” at the World Congress on Universal Logic, pp. 1–20.
- [RK13] Florian Rabe and Michael Kohlhase. “A Scalable Module System”. In: *Information & Computation* 0.230 (2013), pp. 1–54. URL: <https://kwarc.info/frabe/Research/mmt.pdf>.
- [RT] *sLaTeX/RusTeX*. URL: <https://github.com/sLaTeX/RusTeX> (visited on 04/22/2022).

---

<sup>13</sup>EdNOTE: we need an un-numbered version sfragment\*

- [ST] *sTeX – An Infrastructure for Semantic Preloading of LaTeX Documents*. URL: <https://ctan.org/pkg/stex> (visited on 04/22/2022).
- [sTeX] *sTeX: A semantic Extension of TeX/LaTeX*. URL: <https://github.com/sLaTeX/sTeX> (visited on 05/11/2020).
- [Tana] Till Tantau. *beamer – A LaTeX class for producing presentations and slides*. URL: <http://ctan.org/pkg/beamer> (visited on 01/07/2014).
- [Tanb] Till Tantau. *User Guide to the Beamer Class*. URL: <http://ctan.org/macros/latex/contrib/beamer/doc/beameruserguide.pdf>.
- [TL] *TeX Live*. URL: <http://www.tug.org/texlive/> (visited on 12/11/2012).